Unmanned Maritime Autonomy Architecture (UMAA)
Situational Awareness (SA)
Interface Control Document (ICD)
(UMAA-SPEC-SAICD)

MDE Version 5.0.1

UMAA Spec Commit 63fb9f9

Version 5.2.1
20 April 2022

Contents

List of Figures

List of Tables

1 Scope

1.1 Identification

This document defines a set of services and interfaces as part of the Unmanned Maritime Autonomy Architecture (UMAA). The services and corresponding interfaces covered in this ICD encompass the functionality to provide situational awareness for an Unmanned Maritime Vehicle (UMV) (surface or undersea). As such, it provides services that focuses on providing decision-level information, which includes self (ownship) state information (such as navigation data and self-health); environmental information (both a-priori and sensed) such as map/chart data, bathymetry, currents, and weather; and world model data, including contact information and classification information (from single or multiple data-fused sources). This document is generated automatically from data models that define its services and interfaces as part of the Unmanned Systems (UxS) Control Segment (UCS) Architecture as extended by UMAA to provide autonomy services for unmanned vehicles.

To put each ICD in context of the UMAA Architecture Design Description (ADD), the UMAA functional decomposition mapping to UMAA ICDs is shown in Figure 1.

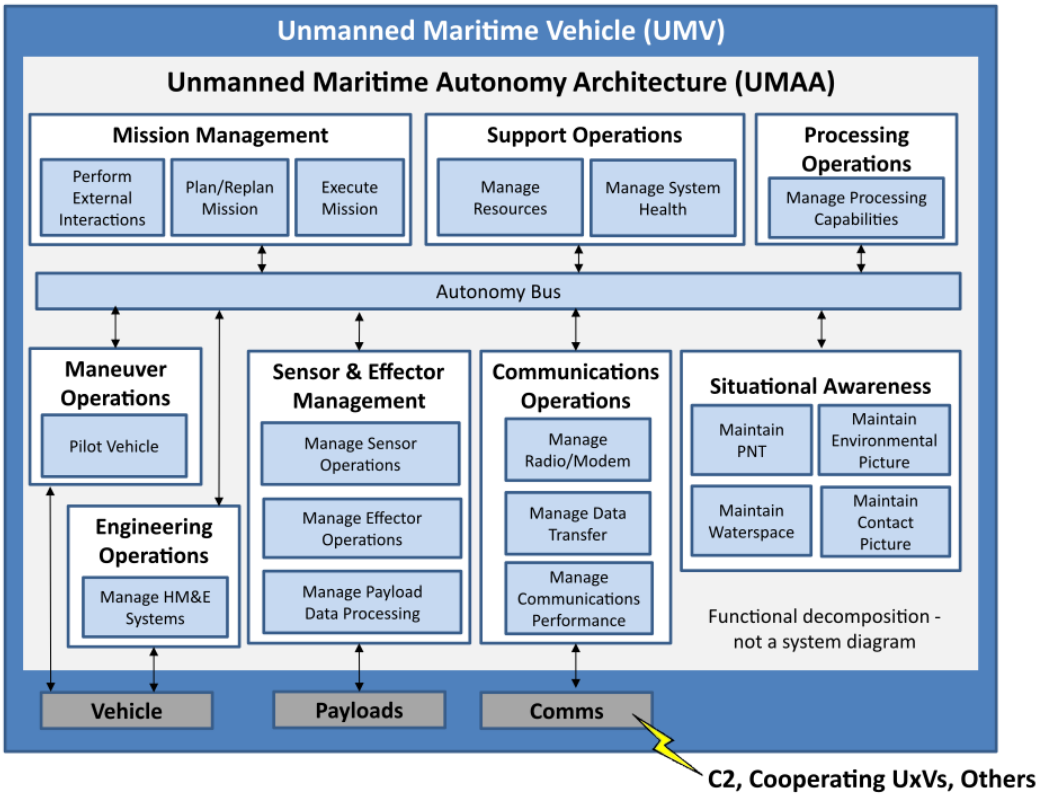

Figure 1: UMAA Functional Organization.

1.2 Overview

The fundamental purpose of UMAA is to promote the development of common, modular, and scalable software for unmanned vehicles that is independent of a particular autonomy implementation. Unmanned Maritime Systems (UMSs) consist of Command and Control (C2), one or more unmanned vehicles, and support equipment and software (e.g. recovery system, Post Mission Analysis applications). The scope of UMAA is focused on the autonomy that resides on-board the unmanned vehicle. This includes the autonomy for all classes of unmanned vehicles and must support varying levels of communication in mission (i.e., constant, intermittent, or none) with external systems. To enable modular development and upgrade of the functional capabilities of the on-board autonomy, UMAA defines eight high-level functions. These core functions include: Communications Operations, Engineering Operations, Maneuver Operations, Mission Management, Processing Operations, Sensor and Effector Operations, Situational Awareness, and Support Operations. In each of these areas, it is anticipated that new capabilities will be required to satisfy evolving Navy missions over time. UMAA seeks to define standard interfaces for these functions so that individual programs can leverage capabilities developed to these standard interfaces across programs that meet the standard interface specifications. Individual programs may group services and interfaces into components in

different ways to serve their particular vehicle's needs. However, the entire interface defined by UMAA will be required as defined in the ICDs for all services that are included in a component. This requirement is what enables autonomy software to be ported between heterogeneous UMAA-compliant vehicles with their disparate vendor-defined vehicle control interfaces without recoding to a vehicle-specific interface.

Situational Awareness includes the services and interfaces required to enable autonomous decision making by providing information gleaned from the sensing system (e.g. radar or sonar), from a-priori data (e.g. bathymetry), and from any externally provided data (contact data from a collaborating UMV). Situational Awareness (often also referred to as world model) also includes such things as data fusion from multiple data sources (e.g. visual and radar), inferencing of higher level knowledge from raw data (e.g. contact reported by a radar is a merchant), registration of different data sources to common spatial references (e.g. contact is 570 yards away on bearing 27 degrees relative to ownship and is at 34.5 degrees latitude and 76.3048 degrees longitude). Standardization of these services enables them to be used across platforms and across programs based on available sensing capabilities on each individual platform.

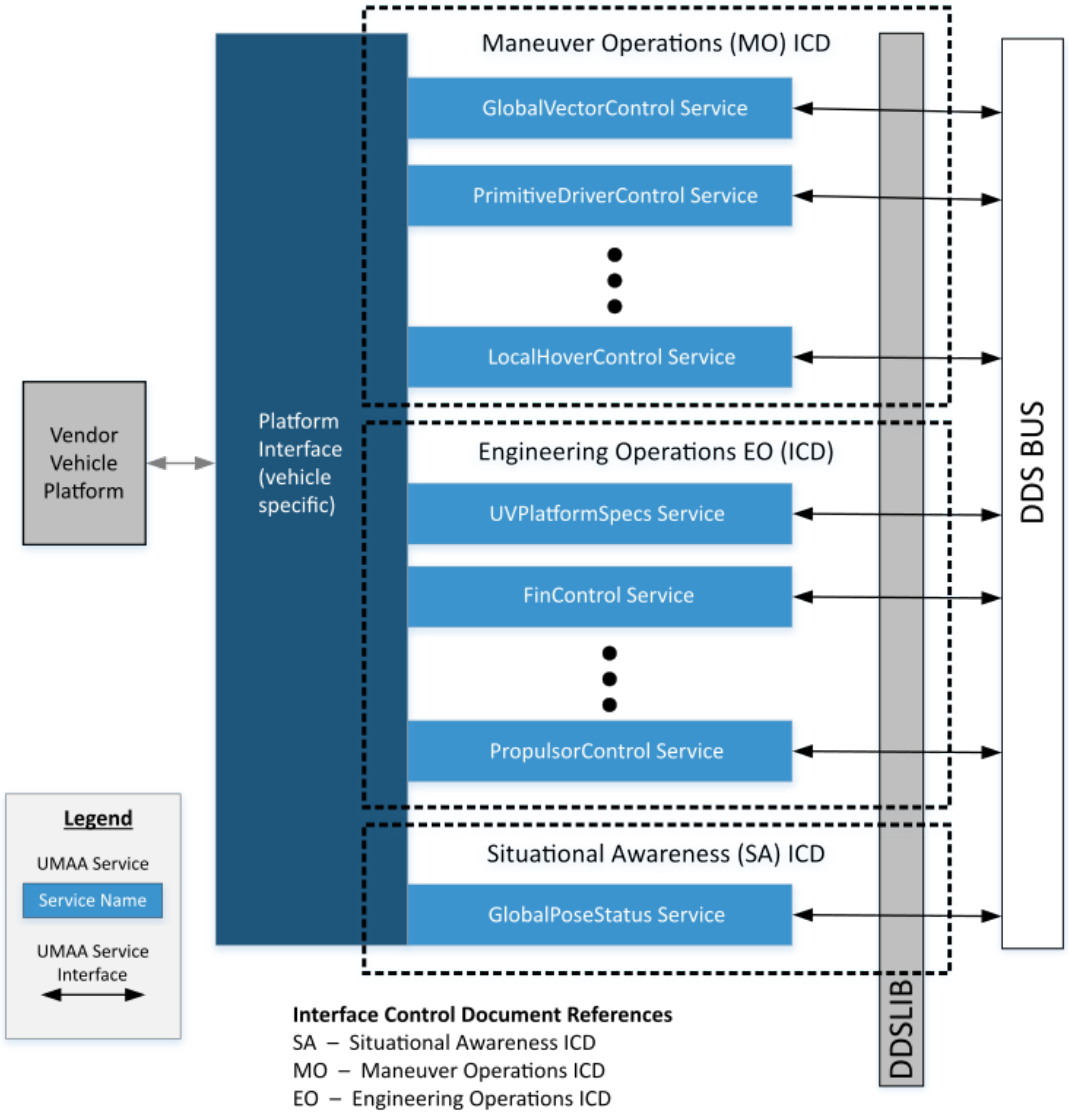

Figure 2: UMAA Services and Interfaces Example.

1.3 Document Organization

This interface control document is organized as follows:

Section 1 – Scope: A brief purview of this document

Section 2 – Referenced Documents: A listing of associated of government and non-government documents and standards

Section 3 – Introduction to Data Model, Services, and Interfaces: A description of the common data model across all services and interfaces

Section 4 – Introduction to Coordinate Reference Frames and Position Model: An overview of the reference frame model used by UMAA

Section 5 – Flow Control: A description of different flow control patterns used throughout UMAA

Section 6 – Situational Awareness (SA) Services and Interfaces: A description of specific services and interfaces for this ICD

2 Referenced Documents

The documents in the following table were used in the creation of the UMAA interface design documents. Not all references may be applicable to this particular document.

Table 1: Standards Documents

Title	Release Date
A Universally Unique IDentifier (UUID) URN Namespace	July 2005
Data Distribution Service for Real-Time Systems Specification, Version 1.4	March 2015
Data Distribution Service Interoperability Wire Protocol (DDSI-RTPS), Version 2.3	April 2019
Object Management Group Interface Definition Language Specification (IDL)	March 2018
Extensible and Dynamic Topic Types for DDS, Version 1.3	February 2020
UAS Control Segment (UCS) Architecture, Architecture Description, Version 2.4	27 March 2015
UCS Architecture, Conformance Specification, Version 2.2	27 September 2014
UCS-SPEC-MODEL v3.4 Enterprise Architect Model	27 March 2015
UCS Architecture, Architecture Technical Governance, Version 2.5	27 March 2015
System Modeling Language Specification, Version 1.5	May 2017
Unified Modeling Language Specification, Version 2.5.1	December 2017
Interface Definition Language (IDL), Version 4.2	March 2018
U.S. Department Of Homeland Security, United States Coast Guard "Navigation Rules International-Inland" COMDTINST M16672.2D	March 1999
IEEE 1003.1-2017 - IEEE Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7	December 2017
Guard, U. C. (2018). Navigation Rules and Regulations Handbook: International—Inland. Simon and Schuster.	June 2018
Department of Defense Interface Standard: Joint Military Symbology (MIL-STD-2525D Appendix A)	10 June 2014
DOD Dictionary of Military and Associated Terms	August 2018

Table 2: Government Documents

Title	Release Date
Unmanned Maritime Autonomy Architecture (UMAA) Architecture Design Description (ADD), Version 1.0	January 2019
Manual for the Submission of Oceanographic Data Collected by Unmanned Undersea Vehicles (UUVs)	October 2018

3 Introduction to Data Model, Services, and Interfaces

3.1 Data Model

A common data model is at the heart of UMAA. The common data model describes the entities that represent system state data, the attributes of those entities and relationships between those entities. This is a "data at rest" view of system-level information. It also contains data classes that define types of messages that will be produced by components, or a "data in motion" view of system-level information.

The common data model and coordinated service interfaces are described in a Unified Modeling Language (UML™) modeling tool and are represented as UML™ class diagrams. Interface definition source code for messages/topics and other interface definition products and documentation will be automatically generated from the common data model so that they are consistent with the data model and to ensure that delivered software matches its interface specification.

The data model is maintained as a Multi-Domain Extension (MDE) to the UCS Architecture and will be maintained under configuration control by the UMAA Board as UCSMDE and will be incrementally integrated into the core UCS standard. Section 6 content is automatically generated from this data model, as are other automated products such as IDL that are used for automated code generation.

3.2 Definitions

UMAA ICDs follow the UCS terminology definitions found in the UCS Architecture Description v2.4. The normative (required) implementation to satisfy the requirements of a UMAA ICD is to provide service and interface specification compliance. Components may group services and required interfaces in any manner so long as every service meets its interface specifications. Figure 3 shows a particular grouping of services into components. The interfaces are represented by the blue and green lines and may equate to one or more independent input and output interfaces for each service. The implementation of the service into software components is left up to the individual system development. Given this context, section 6 correspondingly defines services with their interfaces and not components.

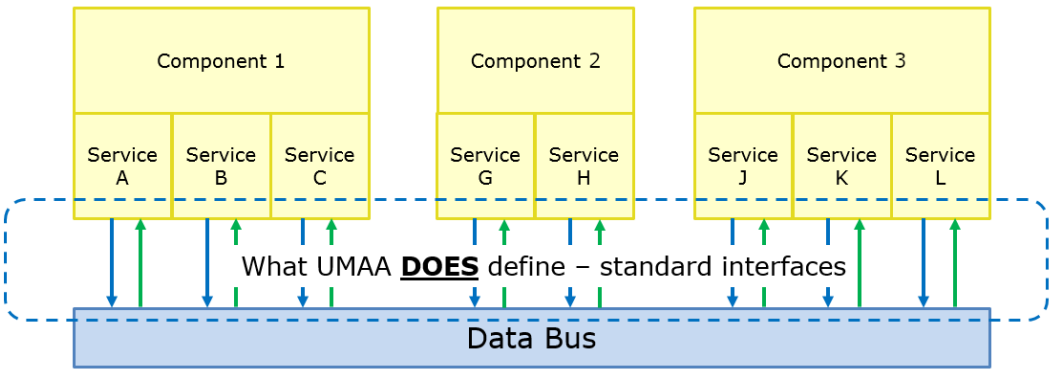

Figure 3: Services and Interfaces Exposed on the UMAA Data Bus.

Services may use other services within this ICD, or in other UMAA defined ICDs, to provide their capability. Additionally, components for acquisition and development may span multiple ICDs. An example of this would be a commercial radar that provides both status and control of the unit via the radar's software Application Programming Interface (API).

3.3 Data Distribution Service (DDS™)

The data bus supporting autonomy messaging (as seen in Figure 3) is implemented via DDS™. DDS is a middleware protocol and API standard for data-centric connectivity from the Object Management Group (OMG). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture. In a distributed system, middleware is the software layer that lies between the operating system and applications. It enables the various system components to more easily communicate and share data. It simplifies the development of distributed systems by letting software developers focus on the specific purpose of their applications rather than the mechanics of passing information between applications and systems. The DDS specification is fully described in free reference material on the OMG website and there are both open source and commercially available implementations.

3.4 Naming Conventions

UMAA services are modeled within the UCS Architecture under the Multi-Domain Extension (MDE). The UCS Architecture uses SoaML concepts of participant, serviceInterface, service port, and request port to describe the interfaces that make up a service and show how the service is used. Each service defines the capability it provides as well as required interfaces. Each interface consists of an operation that accepts a single message (A SoaML MessageType). In SoaML, a MessageType is defined as a unit of information exchanged between participant Request and Service ports via ServiceInterfaces. Instances of a MessageType are passed as parameters in ServiceInterface operations. (Reference: UCS Architecture, Architecture Technical Governance)

To promote commonality across service definitions, a common way of naming services and their sets of operations and messages has been adopted for defining services within UCS-MDE. The convention uses the Service Base Name <SBN> and an optional Function Name [FN] to derive all service names and their associated operations and messages. As this is meant to be a guide, services might not include all of the defined operations and messages and their names might not follow the convention where a more appropriate name adds clarity.

Furthermore, services in UMAA are not required to be defined as indicated in Table 3 when all parts of the service capabilities are required for the service to be meaningful (such as ResourceAllocation).

Additionally, note that for UMAA not all operations defined in UCS-MDE result in a message being published to the DDS bus, e.g., since DDS uses publish/subscribe, most query operations result in a subscription to a topic and do not actually publish the associated request message. In the case of cancel commands, there is no associated implementation of the cancel<SBN>[FN]CommandStatus as it is just the intrinsic response of the DDS dispose function; so, it is essentially a NOOP (no operation) in implementation. The conventions used to define UCS-MDE services are as follows:

Service Name
- <SBN>[FN]Config
- <SBN>[FN]Control
- <SBN>[FN]Specs
- <SBN>[FN]Status OR Report

where the SBN should be descriptive of the task or information provided by the service. Note that the FN is optional and only included if needed to clarify the function of the service. The suffixes Status and Report are interchangeable. If a "Report" is a more appropriate description of the service, it can be used in lieu of "Status".

Table 3: Service Requests and Associated Responses

	Service Requests (Inputs)	**Service Responses (Outputs)**
Config	set<SBN>[FN]Config	report<SBN>[FN]ConfigCommandStatus
	query<SBN>[FN]ConfigAck	report<SBN>[FN]ConfigAck
	query<SBN>[FN]Config	report<SBN>[FN]Config
	cancel<SBN>[FN]Config	report<SBN>[FN]CancelConfigCommandStatus
	query<SBN>[FN]ConfigExecutionStatus	report<SBN>[FN]ConfigExecutionStatus
Control	set<SBN>[FN]	report<SBN>[FN]CommandStatus
	query<SBN>[FN]CommandAck	report<SBN>[FN]CommandAck
	cancel<SBN>[FN]Command	report<SBN>[FN]CancelCommandStatus
	query<SBN>[FN]ExecutionStatus	report<SBN>[FN]ExecutionStatus
Specs	query<SBN>[FN]Specs	report<SBN>[FN]Specs
Status OR Report	query<SBN>[FN]	report<SBN>[FN]

Service Requests (operation:message)
- set<SBN>[FN]Config:<SBN>[FN]ConfigCommandType

query<SBN>[FN]Config:<SBN>[FN]ConfigRequestType[1]
set<SBN>[FN]:<SBN>[FN]CommandType
query<SBN>[FN]CommandAck:<SBN>[FN]CommandAckRequestType[1]
cancel<SBN>[FN]Command:<SBN>[FN]CancelCommandType[1]
cancel<SBN>[FN]Config:<SBN>[FN]CancelConfigType[1]
query<SBN>[FN]ExecutionStatus:<SBN>[FN]ExecutionStatusRequestType[1]
query<SBN>[FN]ConfigExecutionStatus:<SBN>[FN]ConfigExecutionStatusRequestType[1]
query<SBN>[FN]ConfigAck:<SBN>[FN]ConfigAckRequestType[1]
query<SBN>[FN]Specs:<SBN>[FN]SpecsRequestType[1]
query<SBN>[FN]:<SBN>[FN]RequestType [1] [2]

Service Responses (operation:message)
report<SBN>[FN]ConfigCommandStatus:<SBN>[FN]ConfigCommandStatusType
report<SBN>[FN]Config:<SBN>[FN]ConfigReportType
report<SBN>[FN]ConfigAck:<SBN>[FN]ConfigAckReportType
report<SBN>[FN]CommandStatus:<SBN>[FN]CommandStatusType
report<SBN>[FN]CommandAck:<SBN>[FN]CommandAckReportType
report<SBN>[FN]CancelCommandStatus:<SBN>[FN]CancelCommandStatusType[1]
report<SBN>[FN]CancelConfigCommandStatus:<SBN>[FN]CancelConfigCommandStatusType[1]
report<SBN>[FN]ExecutionStatus:<SBN>[FN]ExecutionStatusReportType
report<SBN>[FN]ConfigExecutionStatus:<SBN>[FN]ConfigExecutionStatusReportType
report<SBN>[FN]Specs:<SBN>[FN]SpecsReportType
report<SBN>[FN]:<SBN>[FN]ReportType

where,

- Config (Configuration) Command/Report – This is the setup of a resource for operation of a particular task. Attributes may be static or variable. Examples include: maximum RPM allowed, operational sonar frequency range allowed, and maximum allowable radio transmit power.
- Command Status – This is the current state of a particular command (either control or configuration).
- Command – This is the ability to influence or direct the behavior of a resource during operation of a particular task. Attributes are variable. Examples include a vehicle's speed, engine RPM, antenna raising/lowering, and controlling a light or gong.
- Command Ack (Acknowledgement) Report – This is the command currently being executed.
- Cancel – This is the ability to cancel a particular command that has been issued.
- Execution Status Report – This is the status related to executing a particular command. Examples associated with a waypoint command include cross track error, time to achieve, and distance remaining.
- Specs (Specifications) Report – Provides a detailed description of a resource and/or its capabilities and constraints. Attributes are static. Examples include: maximum RPM of a motor, minimum frequency of a passive sonar sensor, length of the unmanned vehicle, and cycle time of a radar.
- Report – This is the current information being provided by a resource. Examples include vehicle speed, rudder angle, current waypoint, and contact bearing.

3.5 Namespace Conventions

Each UMAA service and the messages under the service can be accessed through their appropriate UMAA namespace. The namespace reflects the mapping of a specific service to its parent ICD, and the parent ICD's mapping to the overall UMAA Design Description. For example:

Access the Primitive Driver Control service under Maneuver Operations:
UMAA::MO::PrimitiveDriverControl
Access the ContactReport Service under Situational Awareness:

[1]These message types are required for UCS model rules of construction, but are not implemented as messages in the UMAA specification.
[2]At this time, there are no Requests in the specification. This will be the message format when Requests have been added.

UMAA::SA::ContactReport

The UMAA model uses common data types that are re-used through the model to define service interface topics, interface topics, and other common data topics. These data types are not intended to be directly utilized but, for reference, they can be accessed in the same manner:

Access the common UMAA Status Message Fields:
UMAA::UMAAStatus
Access the common UMAA GeoPosition2D (i.e., latitude and longitude) structure:
UMAA::Common::Measurement::GeoPosition2D

3.6 Cybersecurity

The UMAA standard addressed in this ICD is independent from defining specific measures to achieve Cybersecurity compliance. This UMAA ICD does not preclude the incorporation of security measures, nor does it imply or guarantee any level of Cybersecurity within a system. Cybersecurity compliance will be performed on a program-specific basis and compliance testing is outside the scope of UMAA.

3.7 GUID algorithm

The UMAA standard utilizes the Globally Unique IDentifier (GUID), conforming to the variant defined in RFC 4122 (variant value of 2). Generators of GUIDs may generate GUIDs of any valid, RFC 4122-defined version that is appropriate for their specific use case and requirements. (Reference: A Universally Unique IDentifier (UUID) URN Namespace)

3.8 Large Collections

The UMAA standard defines Large Collections, which are collections of decoupled but related data. Large Collections provide the ability to update one or more elements of the collection without republishing the entire collection to the DDS bus. This avoids two problems related to using an unbounded sequence type in a DDS message: 1) resource consumption growing as the collection is appended to or updated, and 2) DDS implementation-specific limitations on unbounded sequences. There are two implementations of a Large Collection: the Large Set (unordered) and the Large List (ordered).

In both Large Collection implementations, there are two important abstractions: the collection metadata and collection element type. Because Large Collections are specific to the UMAA PSM, the type definitions for the collection metadata and collection element are not part of MDE, and the IDL definitions of these types are generated separately. A particular UMAA message that has a Large Collection attribute will reference the metadata type (LargeSetMetadata or LargeListMetadata). The collection element type is defined under the same namespace as the message that uses it, and follows the naming pattern <parent message name><attribute name><collection type>Element. Each element of the collection is published as a separate message on the DDS bus, and can be tracked back to their related collection using the setID or listID. Users can also trace an element in a set to the source attribute (a NumericGUID) of the Service Provider that generated the report with this set using the collection metadata.

3.8.1 Necessary QoS

To achieve the Large Collection consistency in the update process described below, ordering of samples on the collection element type topic is necessary. Therefore, publishers and subscribers to the collection element type topic must use the `PRESENTATION` QoS policy with an `access_scope` of `DDS_TOPIC_PRESENTATION_QOS` and `ordered_access`.

3.8.2 Updating Large Collections

When elements of the collection are updated, the metadata must be updated as well to signal a change in the set. The `updateElementID` is updated to match the elementID of the element whose reception signals the end of the atomic update of the collection. Because of the requirement of an ordered topic described above, this will be the element that is updated last chronologically. The metadata `updateElementTimestamp` must be updated to the timestamp of the same element that signals the end of the update.

The set can be updated as a batch (multiple elements in a single "update cycle," as determined by the provider). This allows for a coarse synchronization: data elements that do not match the metadata `updateElementID` and `updateElementTimestamp` can be assumed to be part of an in-progress update cycle. Consumers can choose to immediately act on those data individually

or wait until the matching element is received to signal that the complete update cycle has finished and consider the set as a whole. Note that the coarseness of synchronization is service-dependent: in some cases an intermediate view of a collection update may be logically incorrect to act upon.

3.8.3 Specifying an Empty Large Collection

A particular Large Collection can be empty during initial creation. This is indicated by publishing metadata with a `size` of zero and an `updateElementID` set to the Nil UUID. As specified in section 4.1.7 of the referenced document "A Universally Unique IDentifier (UUID) URN Namespace", this is a "special form of UUID that is specified to have all 128 bits set to zero".

3.8.4 Large Set Types

The following details the LargeSetMetadata structure:

Table 4: LargeSetMetadata Structure Definition

Attribute Name	Attribute Type	Attribute Description
setID	NumericGUID	Identifies the Large Set instance this metadata relates to.
updateElementID	NumericGUID	This field references the element ID of the set element whose reception signals the end of an atomic update to this set. This elementID must be used in conjunction with the updateElementTimestamp below to fully identify when the atomic update has completed and the set is stable.
updateElementTimestamp†	DateTime	This field identifies the elementTimestamp of the element, referenced above by updateElementID, that signals the end of an atomic update to this set. This field will be empty in the event that the element update results from a DDS dispose.
size	LargeCollectionSize	Indicates the number of elements associated with this set after the atomic update is complete.

An example element type is shown below, where a `FooReportType` message has a Large Set attribute called "items" whose type is `BarType`

Table 5: Example FooReportTypeItemsSetElement Structure Definition

Attribute Name	Attribute Type	Attribute Description
element	BarType	The value of the set element.
setID	NumericGUID	Identifies the Large Set instance this element relates to.
elementID*	NumericGUID	Uniquely identifies this element within the set and across all large collection elements that currently exist on the DDS bus.
elementTimestamp	DateTime	The timestamp of this element.

3.8.5 Large List Types

The following details the LargeListMetadata structure:

Table 6: LargeListMetadata Structure Definition

Attribute Name	Attribute Type	Attribute Description
listID	NumericGUID	Identifies the Large List instance this metadata relates to.
updateElementID	NumericGUID	This field references the element ID of the list element whose reception signals the end of an atomic update to this list. This elementID must be used in conjunction with the updateElementTimestamp below to fully identify when the atomic update has completed and the list is stable.
updateElementTimestamp†	DateTime	This field identifies the elementTimestamp of the element, referenced above by updateElementID, that signals the end of an atomic update to this list. This field will be empty in the event that the element update results from a DDS dispose.
startingElementID	NumericGUID	This field identifies the list element, tying to its elementID, that is sequentially first in the list. This is provided for convenience when iterating through the linked list using the nextElementID field.
size	LargeCollectionSize	Indicates the number of elements associated with this set after the atomic update is complete.

An example element type is shown below, where a `FooReportType` message has a Large List attribute called "items" whose type is `BarType`

Table 7: Example FooReportTypeItemsListElement Structure Definition

Attribute Name	Attribute Type	Attribute Description
element	BarType	The value of the list element.
listID	NumericGUID	Identifies the Large List instance this element relates to.
elementID*	NumericGUID	Uniquely identifies this element within the list and across all large collection elements that currently exist on the DDS bus.
elementTimestamp	DateTime	The timestamp of this element.
nextElementID†	NumericGUID	This field references to the elementID of the element that logically follows this element in the linked list. This is empty if this element is sequentially last.

4 Introduction to Coordinate Reference Frames and Position Model

4.1 Vehicle Reference Frame

In the following Service Definitions, we use the parameters yaw, pitch, and roll to define the vehicle orientation with respect to the specified reference frame. Each parameter is described as a rotation around a given axis: Yaw about the Z axis. Pitch about the Y axis. Roll about the X axis. A UUV is shown in the diagrams because it has more degrees for freedom for its pose and motion, however, the terminology equally applies to both USVs and UUVs.

The axes are defined as:

- X - Positive in the forward direction, negative in the aft.
- Y - Positive in the starboard direction, negative in the port.
- Z - Positive in the down direction, negative in the up.

Additionally, rotations about all axes follow the right-hand rule.

4.2 Earth-Centered Earth-Fixed Frame

The Earth-Centered Earth-Fixed (ECEF) frame is a global reference frame with its origin at the center of the ellipsoid modeling the Earth's surface (Figure 4). The Z-axis points along the Earth's axis of rotation through the North Pole. The X-axis points from the origin to the intersection of the equator with the prime meridian, which defines 0° longitude. The Y-axis completes the right-handed orthogonal system, intersecting the equator at the 90° east meridian.

4.3 North-East-Down Frame

The North-East-Down (NED) frame is defined with an origin at the object described by the navigation solution. The Down axis is defined as normal to the surface of the reference ellipsoid in the direction pointing towards the interior of the Earth. The North axis is the projection of the line from the object to the north pole onto the plane orthogonal to the Down axis. The East axis completes the right-handed orthogonal system and points in the East direction.

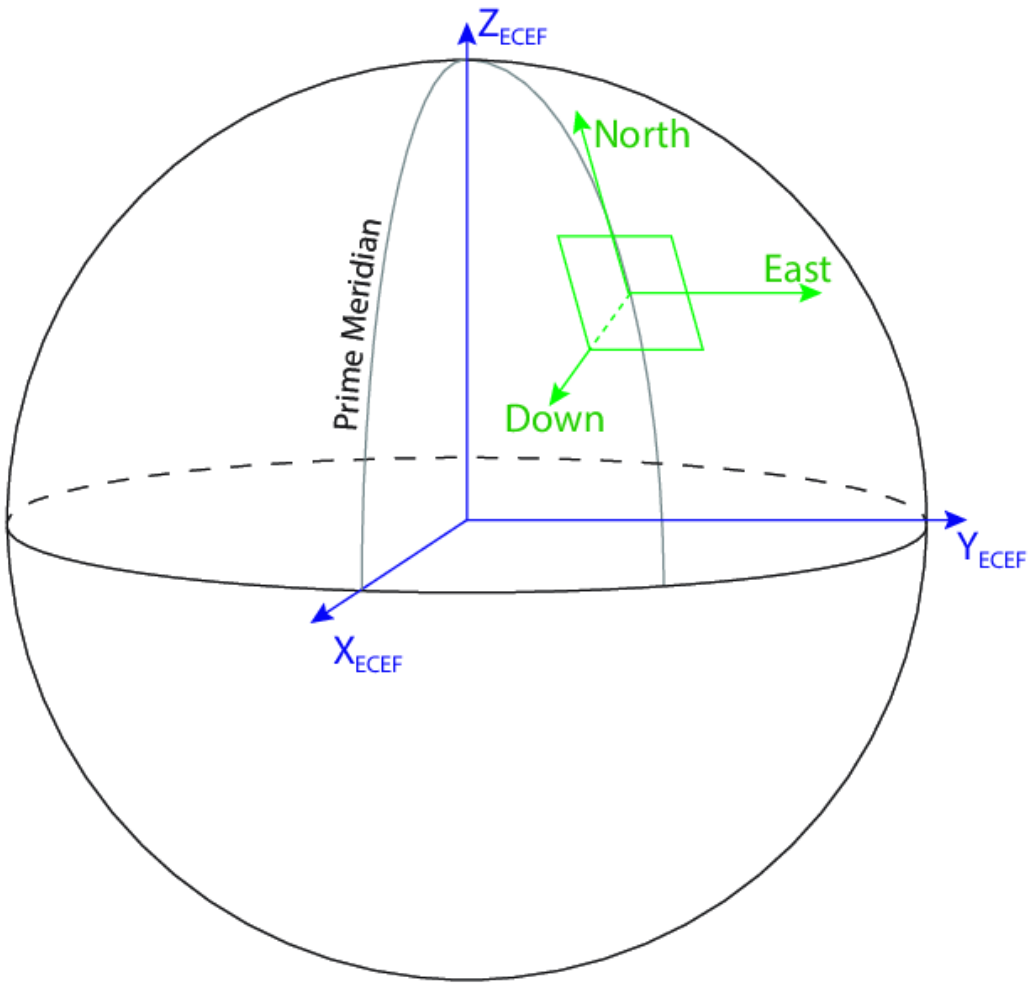

Figure 4: Origins and axes of the Earth-Centered Earth-Fixed (ECEF) and North-East-Down (NED) frames.

4.4 WGS 84

The World Geodetic System (WGS) 1984 defines a standard coordinate system for the Earth. It represents the Earth as an oblate spheroid, and defines the mapping between latitude-longitude-altitude (LLA) coordinates and Cartesian ECEF coordinates. GPS reports positions in WGS 84 LLA coordinates. It has become the standard datum for navigation.

While the UMAA services typically make use of the coordinate systems defined by WGS 84, it also defines an Earth Gravity Model (EGM) and a World Magnetic Model (WMM) which are updated regularly.

4.5 Vehicle Orientation

Determining the orientation of the vehicle (Figure 5) with respect to any reference frame is carried out via the following procedure (Figure 6).

1. Align the vehicle's longitudinal or X axis with the reference frame X axis. In the global reference frame, this is the north direction.
2. Align the vehicle's down or Z axis with the reference frame's Z axis. In the global reference frame, this is the gravity direction.
3. Ensure that the vehicle's transverse or Y axis is aligned with the reference frame's Y axis. In the global reference frame, this is the east direction.
4. Rotate the vehicle about the vehicle's Z axis by the yaw angle (Figure 7).
5. Rotate the vehicle about the vehicle's newly oriented Y axis by the pitch angle (Figure 8).

6. Rotate the vehicle about the vehicle's newly oriented X axis by the roll angle (Figure 9).

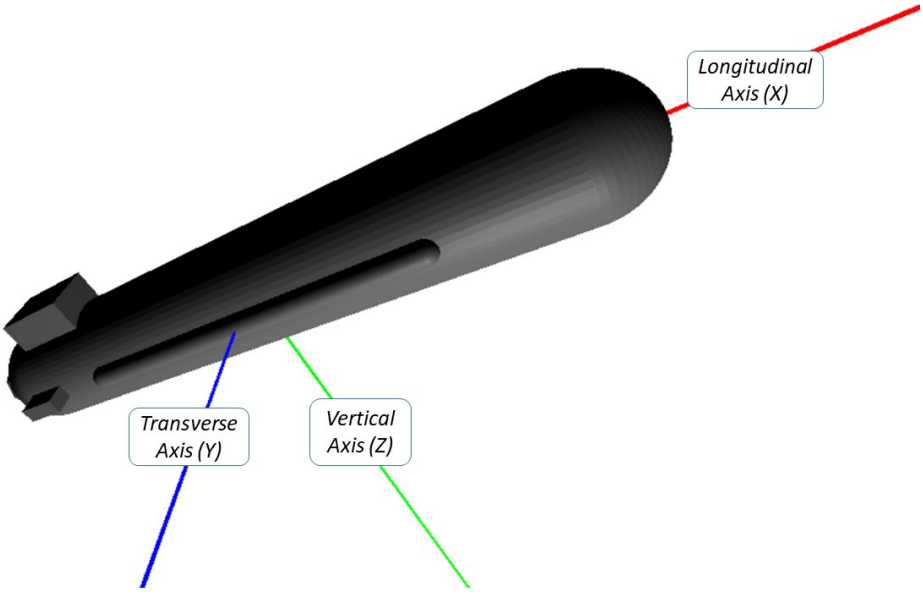

Figure 5: Define the Vehicle Coordinate System

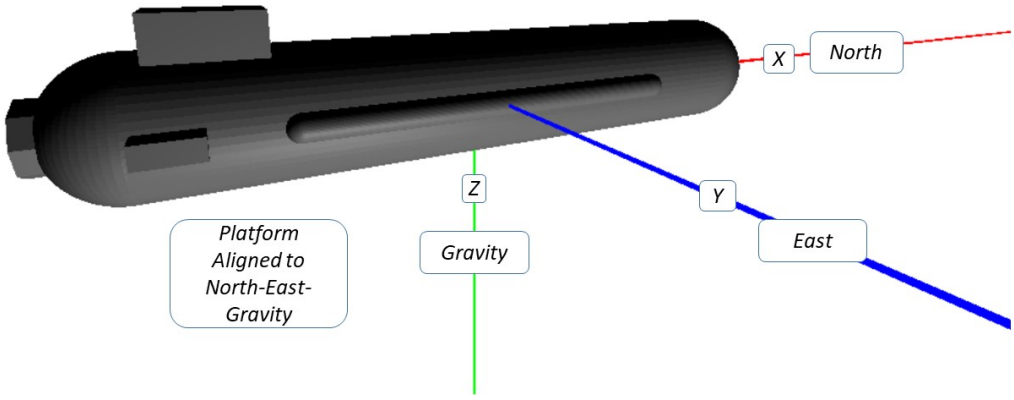

Figure 6: Align the Vehicle with the Reference Frame Axes.

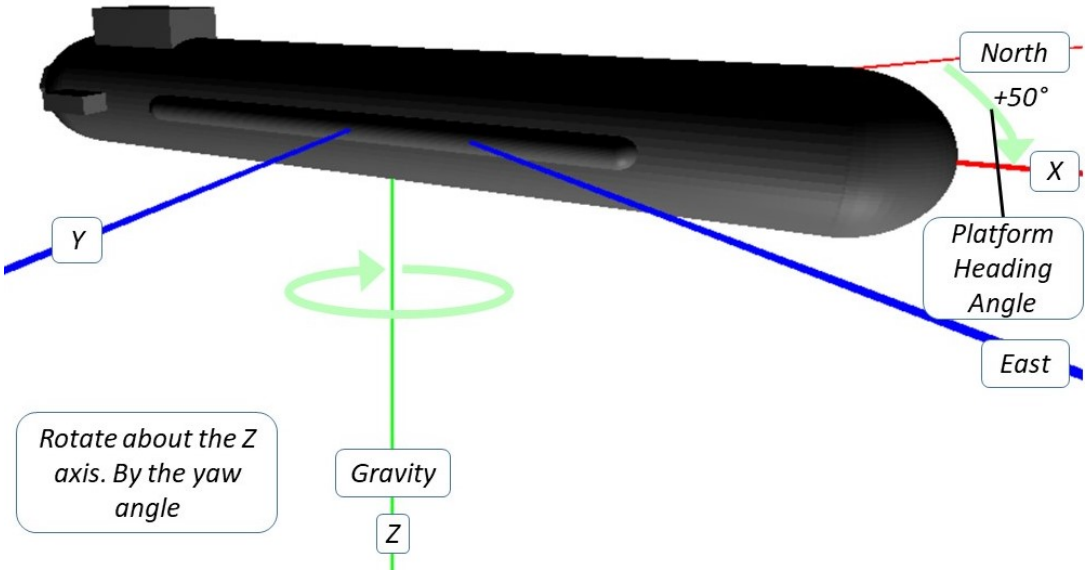

Figure 7: Rotate the Vehicle by the Yaw Angle.

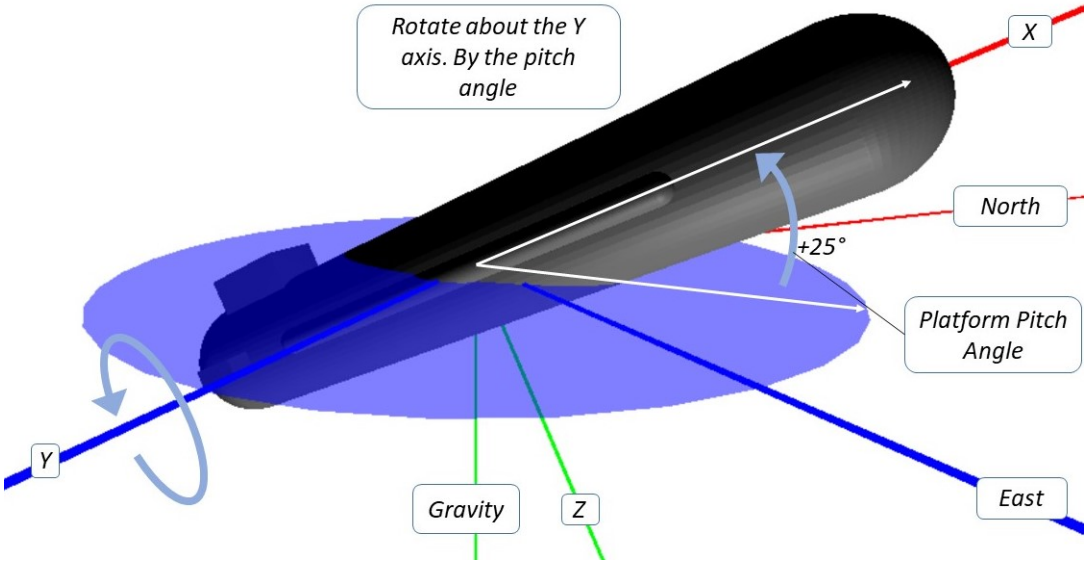

Figure 8: Rotate the Vehicle by the Pitch Angle.

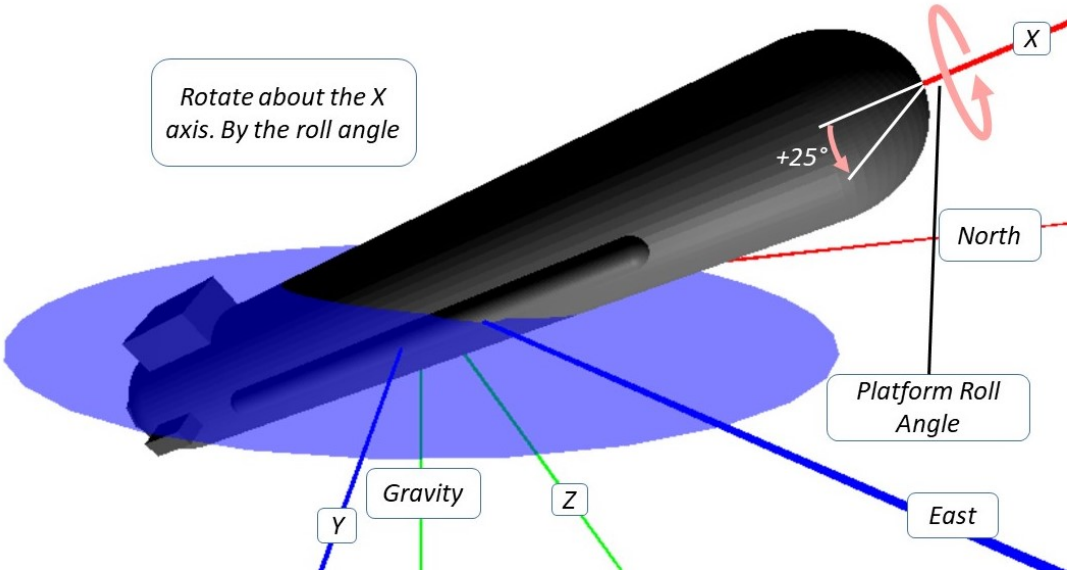

Figure 9: Rotate the Vehicle by the Roll Angle.

4.6 Vehicle Coordinate Reference Frame Origin

UMAA does not specify a required origin for the vehicle coordinate reference frame. However, certain applications may benefit from defining a specific origin such as the registration of multiple sensors with associated offsets for data fusion. Possible origins include the keel/transom intersection, bow/waterline intersection, center of gravity, center of buoyancy and location of INS. A few examples follow.

Definitions

- Keel Transom Intersection
  - Beam at Waterline (BWL) - The maximum distance of the vehicle at the waterline, the distance along the Y axis of the widest point of the hull where it meets the waterline.
  - Design Waterline (DWL) - The line representing the waterline on the vehicle at designed load in summer temperature.
  - Keel - The principal fore-and-aft component of a ship's framing, located along the centerline of the bottom and connected to the stem and stern frames.
  - Length at Waterline (LWL) - The measured distance of the vehicle at the level where it sits in the water, measured along the X axis.
  - Transom - The aftermost transverse flat or shaped plating enclosing the hull.

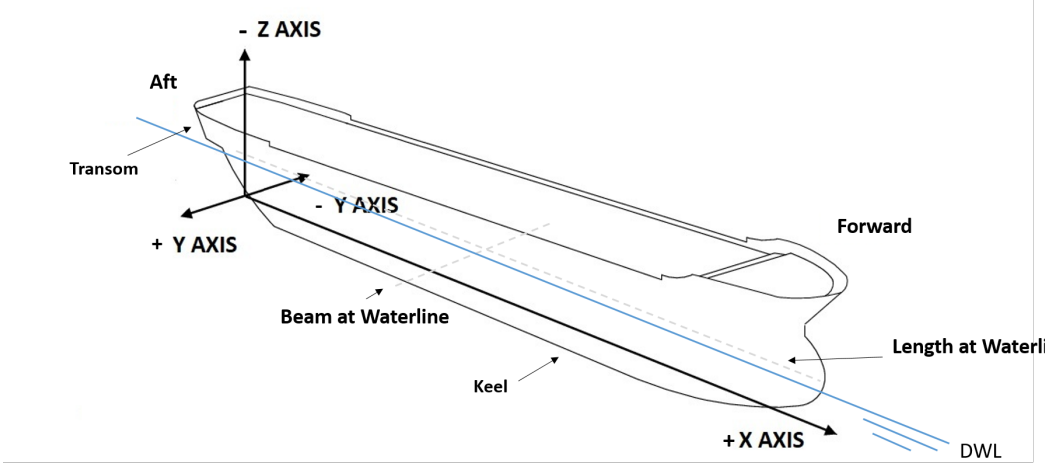

Figure 10: Keel Transom Intersection Origin Location on a USV as Example

- Center of Buoyancy
  - X - The Longitudinal Center of Buoyancy (LCB) when fully submerged.
  - Y - The symmetrical centerline.
  - Z - The Vertical Center of Buoyancy (VCB) when fully submerged.

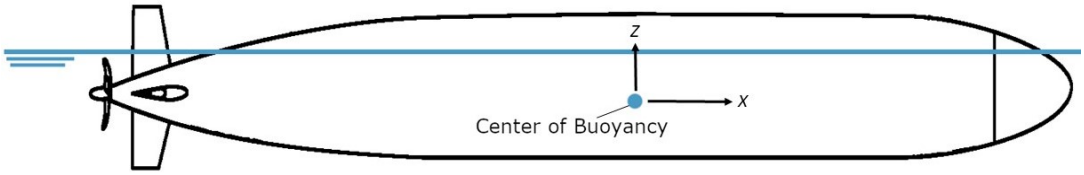

Figure 11: Center of Buoyancy Origin Location on a UUV as Example.

5 Flow Control

5.1 Command / Response

This section defines the flow of control for command/response over the DDS bus. A command/response controls a specific service. While the exact names and processes will depend on the specific service and command being executed, all command/responses in UMAA follow a similar pattern. A notional "Function" command `FunctionCommand` is used in the following examples. As will be described in subsequent paragraphs, DDS publish/subscribe methods are used in implementations to issue commands and responses.

To direct a `FunctionCommand` at a specific Service Provider, UMAA includes a `destination` GUID in all commands. A Service Provider is required to respond to all `FunctionCommand`s where the `destination` is the same as the Service Provider's ID. The Service Consumer will also create a `sessionID` for the command when commanded. The `sessionID` is used to track the command execution as a key into other command-related messages. The `sessionID` must be unique across all `FunctionCommand` instances that are active (i.e. currently on the DDS bus), otherwise the Service Provider will consider the `FunctionCommand` to be a command update (see Section 5.1.4.2). Once a `FunctionCommand` is removed from the DDS bus as part of the Command Cleanup process (see Section 5.1.5), its `sessionID` may be reused for future commands without triggering a command update; therefore it is not necessary for a Service Provider to maintain a complete history of `sessionID`s.

Service Provider and Service Consumer terminology in the following sections is adopted from the OMG Service-oriented architecture Modeling Language (SoaML).

To initialize, a Service Provider (controllable resource) subscribes to the `FunctionCommand` DDS topic. At startup or right before issuing a command, the Service Consumer (controlling resource) subscribes to the `FunctionCommandStatus` DDS topic. Optionally, the Service Consumer may also subscribe to the `FunctionCommandAckReport` to monitor which command is currently being executed, and the `FunctionExecutionStatusReport` (if defined for the Function service) that provides reporting on function-specific data status.

Both Service Providers and Service Consumers are required to recover or clean up any previous persisted commands on the bus during initialization.

To execute a command, the Service Consumer publishes a `FunctionCommandType` to the DDS bus. The Service Provider will be notified and will begin processing the request. During each phase of processing, the Service Provider will provide updates to the Service Consumer via published updates to a related `FunctionCommandStatus` topic. Command responses are correlated to their originating command via the `sessionID`. If a command with a duplicate `sessionID` is received, the Service Provider will regard this as a command update, and follow the flow control detailed in Section 5.1.4.2. Command status updates are provided in the command responses via the `commandStatus` field with additional details included in the `commandStatusReason` field. The Service Provider will also publish the current executing command to the `FunctionCommandAckReport` topic. When defined for the Function service, the Service Provider must also publish the `FunctionExecutionStatusReport` topic and update it as appropriate throughout the execution of the command.

The required state transitions for the `commandStatus` field are shown in Figure 12. Commands may complete normally, or they may terminate early due to failure (Section 5.1.4.4) or cancellation (Section 5.1.4.5). The state machine for a command can also be reset to `ISSUED` via a command update (Section 5.1.4.2). If there is not a self-transition indicated in the diagram, you cannot republish that state in a message. Every command must transition through the states as defined. For example, it is a violation to transition from `ISSUED` to `EXECUTING` without transitioning through `COMMANDED`. Even in the case where there is no logic executing between the `ISSUED` and `EXECUTING` states, the Service Provider is required to transition through `COMMANDED`. This ensures consistent behavior across different Service Providers, including those that do require the `COMMANDED` state.

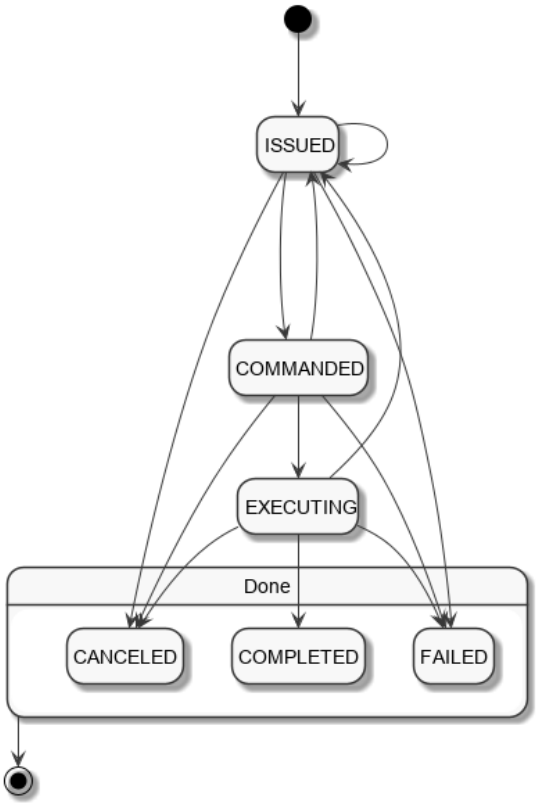

Figure 12: State transitions of the `commandStatus` as commands are processed.

As described above, each time a command transitions to a new state, a `FunctionCommandStatus` message is published containing the updated `commandStatus` and a `commandStatusReason` that indicates why the state transition happened. The table below shows all valid `commandStatusReason` values for each `commandStatus` transition.

	Ending State					
Starting State	ISSUED	COMMANDED	EXECUTING	COMPLETED	FAILED	CANCELED
Initial State	SUCCEEDED	—	—	—	—	—
ISSUED	UPDATED	SUCCEEDED	—	—	VALIDATION_FAILED RESOURCE_FAILED INTERRUPTED TIMEOUT SERVICE_FAILED	CANCELED
COMMANDED	UPDATED	—	SUCCEEDED	—	RESOURCE_REJECTED INTERRUPTED TIMEOUT SERVICE_FAILED	CANCELED
EXECUTING	UPDATED	—	—	SUCCEEDED	OBJECTIVE_FAILED RESOURCE_FAILED INTERRUPTED TIMEOUT SERVICE_FAILED	CANCELED
COMPLETED	—	—	—	—	—	—
FAILED	—	—	—	—	—	—
CANCELED	—	—	—	—	—	—

Figure 13: Valid `commandStatusReason` values for each `commandStatus` state transition. Entries marked with a (—) indicate that the state transition is invalid.

In the following sections, the sequence diagrams demonstrate different exchanges between a Service Consumer and Service

Provider. Within the diagrams, the dashed arrows represent implementation-specific communications that are outside of UMAA's scope. These sequence diagrams are just an example of one possible implementation. Other implementations may have different communication patterns between the Service Provider and the Resource or be implemented completely within the Service Provider process itself (no dependency on an external Resource). Likewise, the interactions between the User and Service Consumer may follow similar or different patterns. However, the UMAA-defined exchanges with the DDS bus between the Service Consumer and Service Provider must happen in the order shown within the sequence diagrams.

5.1.1 High-Level Flow

The high-level flow of a command sequence is shown in Figure 14 and can be described as follows:

1. The Command Startup Sequence is performed.
2. For each command to be executed:
   (a) The Command Start Sequence is performed.
   (b) The command is executed (sequence depends on the execution path, i.e., success, failure, or cancel).
   (c) The Command Cleanup Sequence is performed.
3. The Command Shutdown Sequence is performed.

The `ref` blocks will be defined in later sequence diagrams. Note that the duration of the system execution for any particular `FunctionCommandType` is defined by the combination of the Service Provider(s) and Service Consumer(s) in the system and may not be identical to the overall system execution duration. For example, providers may only be available to execute certain commands during specific mission phases or when certain hardware is in specific configurations. This Command Startup Sequence is not required to happen during a system startup phase. The only requirement is that it must be completed by at least one Service Provider and one Service Consumer before any `FunctionCommandType` commands can be fully executed. Likewise, the Command Shutdown sequence may occur at any time the `FunctionCommandType` will no longer be supported. There is no requirement stating that the Command Shutdown Sequence only be performed during a system shutdown phase.

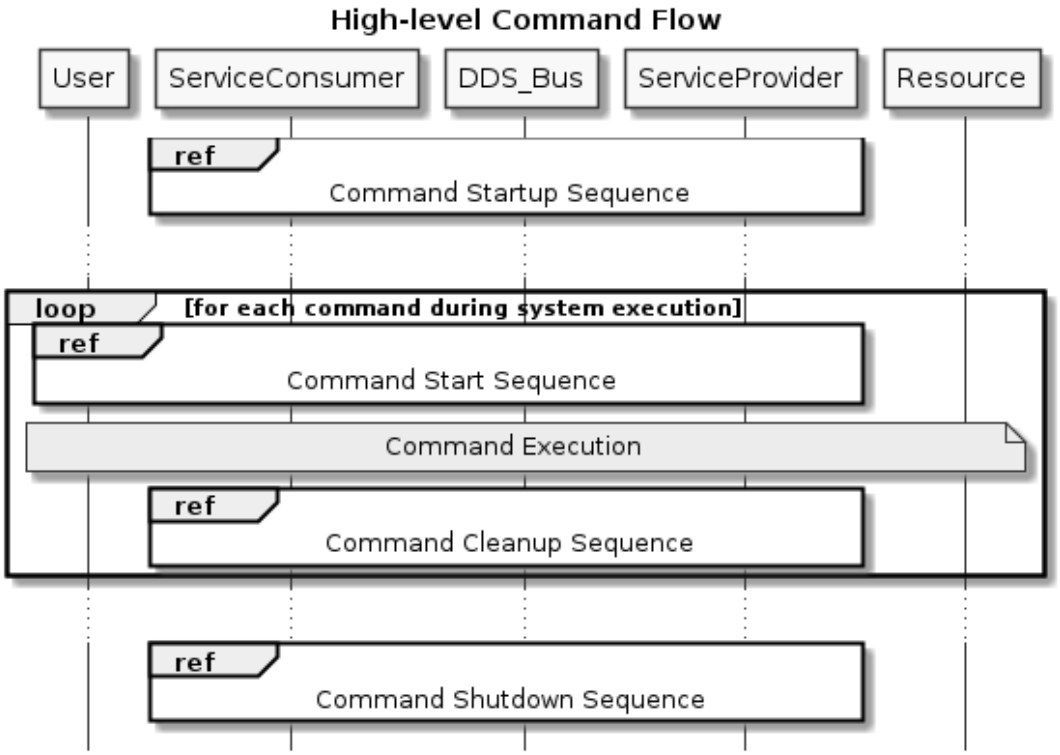

Figure 14: Sequence Diagram for the High-Level Description of a Command Execution.

5.1.2 Command Startup Sequence

As part of initialization both the Service Provider and Service Consumer are required to perform a startup sequence. This startup prepares the Service Provider to execute commands and the Service Consumer to request commands and monitor the progress of those requested commands.

The Service Provider and Service Consumer can initialize in any order. Commands will not be completely executed until both have completed their initialization. The sequence diagram is shown in Figure 15.

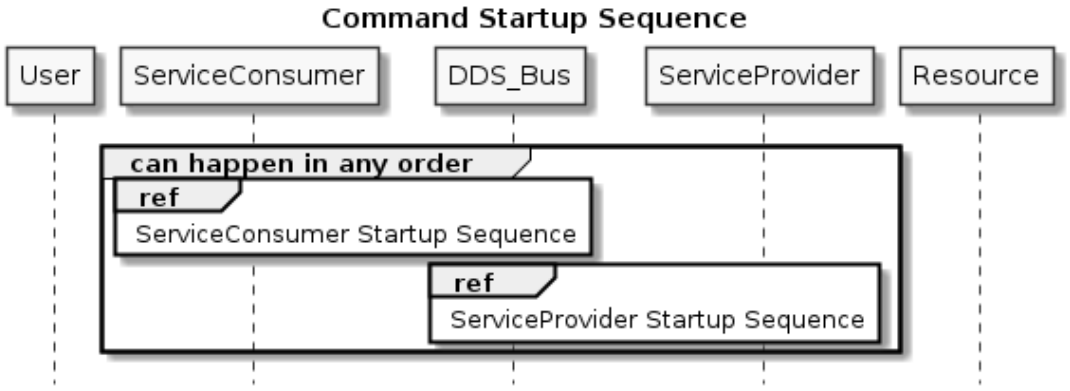

Figure 15: Sequence Diagram for Command Startup.

5.1.2.1 Service Provider Startup Sequence During startup, the Service Provider is required to register as a publisher to the `FunctionCommandStatus`, `FunctionCommandAckReport`, and (if defined for the Function service) the `FunctionExecutionStatusReport` topics.

The Service Provider is also required to subscribe to the `FunctionCommand` topic to be notified when new commands are published.

Finally, the Service Provider is required to handle any existing `FunctionCommandType` commands persisted on the DDS bus with the Service Provider's ID. For each command, if the Service Provider can and wishes to recover, it can continue to execute the command. To obtain the last published state of the command, the Service Provider must subscribe to the `FunctionCommandStatusType`. The Service Provider will continue following the normal status update sequence, picking up from the last status on the bus. If the Service Provider cannot or chooses not to continue processing the command, it must fail the command by publishing a `FunctionCommandStatus` with a `commandStatus` of `FAILED` and a `reason` of `SERVICE_FAILED`.

The Service Provider Startup sequence is shown in Figure 16.

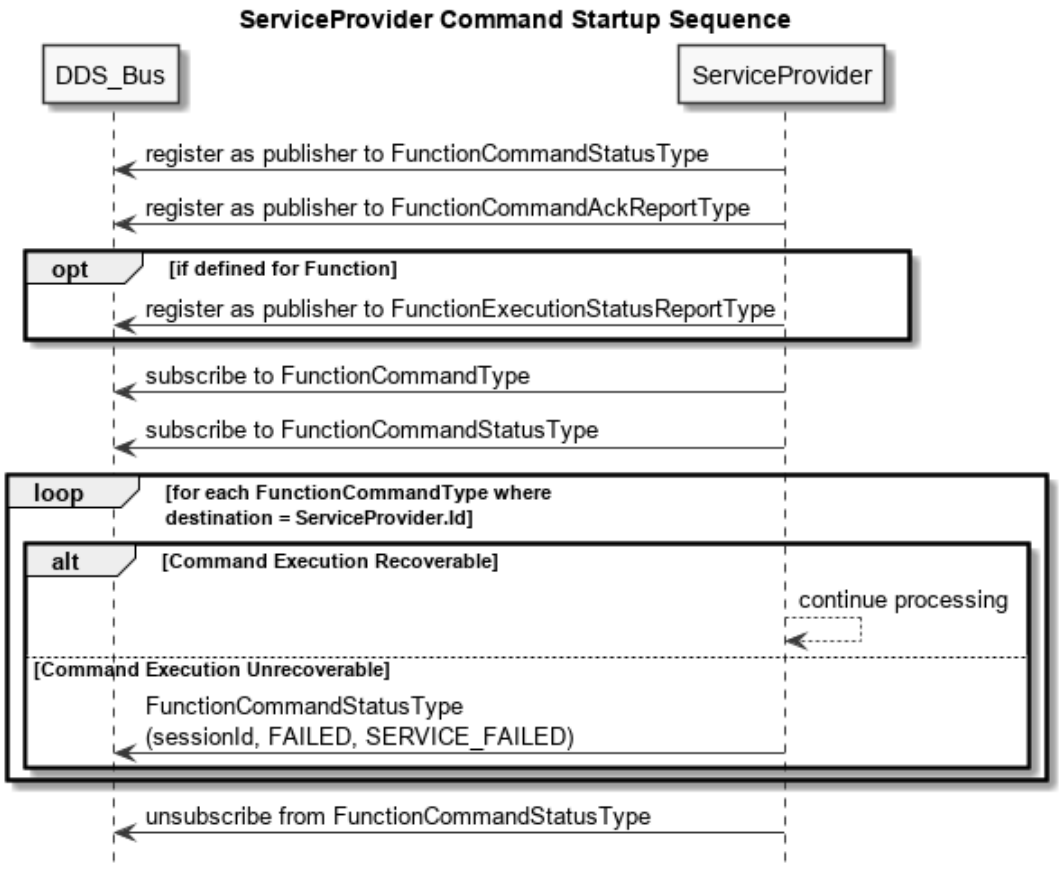

Figure 16: Sequence Diagram for Command Startup for Service Providers.

5.1.2.2 Service Consumer Startup Sequence During startup, the Service Consumer is required to register as a publisher of the `FunctionCommandType`.

The Service Consumer is also required to subscribe to the `FunctionCommandStatusType` to monitor the execution of any published commands. The Service Consumer can optionally register for the `FunctionCommandAckReportType` and, if defined for the Function service, the `FunctionExecutionStatusReportType` if it desires to track additional status of the execution of commands.

Finally, the Service Consumer is required to handle any existing `FunctionCommandType` commands persisted on the DDS bus with this Service Consumer's ID. To find existing `FunctionCommandType`s on the bus, it must first subscribe to the topic. If the Service Consumer can and wishes to recover, it can continue to monitor the execution of the command. If the Service Consumer cannot or chooses not to continue the execution of the command, it must cancel the command via the normal command cancel method.

The Service Consumer Startup sequence is shown in Figure 17.

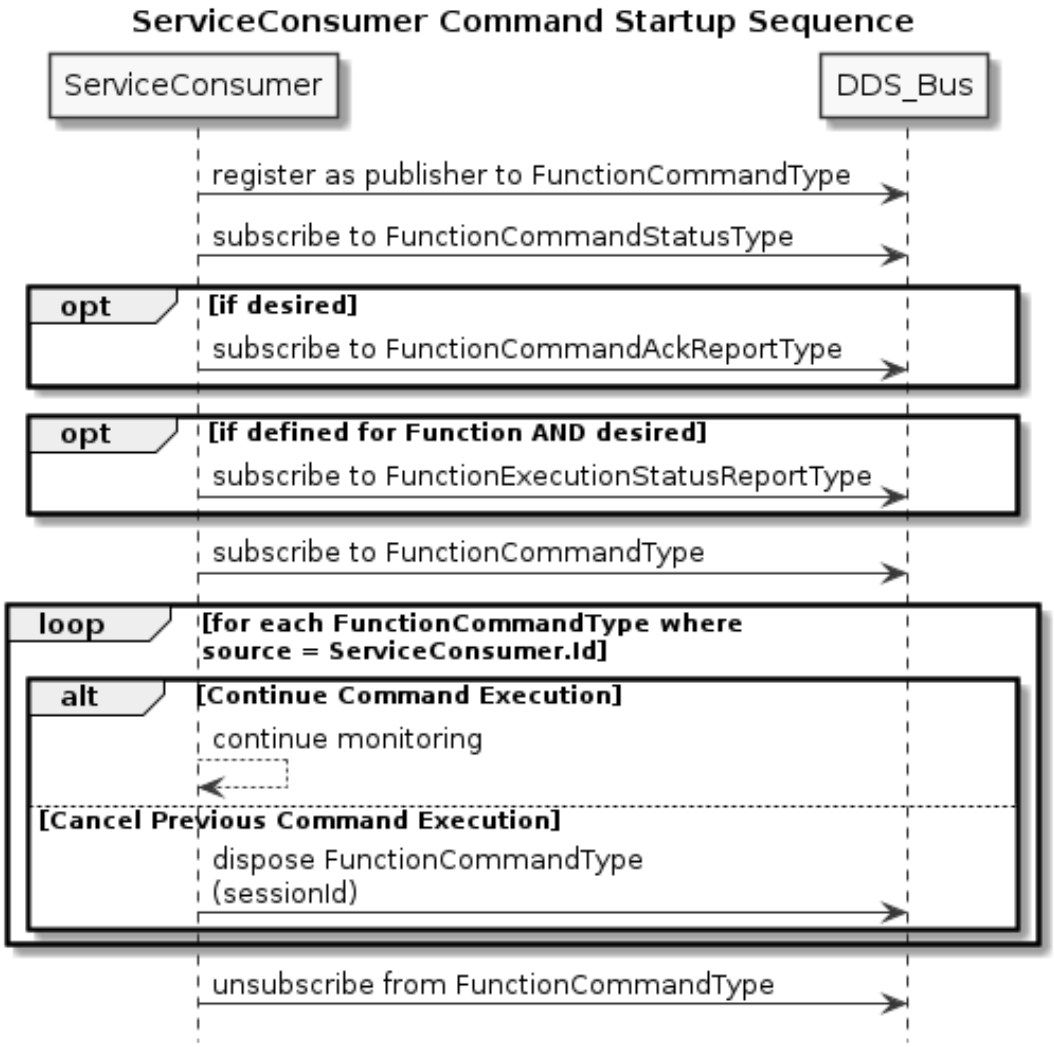

Figure 17: Sequence Diagram for Command Startup for Service Consumers.

5.1.3 Command Execution Sequences

Once both the Service Provider and Service Consumer have performed the startup sequence, the system is ready to begin issuing and executing commands.

5.1.4 Command Start Sequence

The initial start sequence to execute a single new command follows this pattern:

1. The User of the Service Consumer issues a request for a command to be executed.
2. The Service Consumer publishes the `FunctionCommandType` with a unique session ID, the source ID of the Service Consumer, and the destination ID of the desired Service Provider.
3. The Service Provider, upon notification of the new `FunctionCommandType`, publishes a new `FunctionCommandStatusType` with (1) the same session ID as the new `FunctionCommandType`, (2) the status of `ISSUED` and (3) the reason of `SUCCEEDED` to notify the Service Consumer it has received the new command.

The Command Start Sequence for a new command is shown in Figure 18. This pattern will be repeated each time a new command is requested. Note that the Command Start Sequence differs if the `FunctionCommandType` has a `sessionID` that matches another `FunctionCommandType` that currently exists on the DDS bus. This is considered a command update and detailed in Section 5.1.4.2.

After the Command Start Sequence, the sequence can take different paths depending on the actual execution of the command,

detailed from Section 5.1.4.1 to Section 5.1.4.5, but they do not enumerate all of the possible execution paths. Other paths (e.g., an objective failing) will follow a similar pattern to other failures; all are required to follow the state diagram shown in Figure 12 and eventually end with the Command Cleanup Sequence (shown in Figure 25).

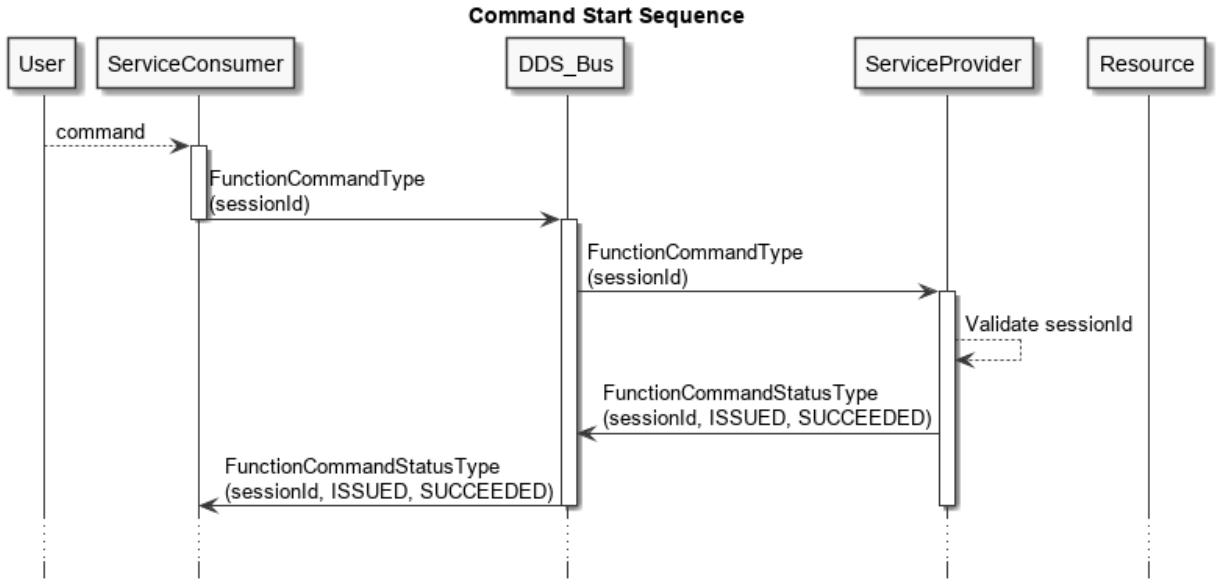

Figure 18: Sequence Diagram for the Start of a Command Execution.

5.1.4.1 Command Execution Once a Service Provider starts to process a command, the Command Execution sequence is:

1. The Service Provider publishes a `FunctionCommandAckReportType` with matching session ID and parameters as the `FunctionCommandType` it is starting to process.
2. The Service Provider performs any validation and negotiation with backing resources as necessary. Once the command is ready to be executed, the Service Provider publishes a `FunctionCommandStatusType` with a status `COMMANDED` and reason `SUCCEEDED` to notify the Service Consumer that the command has been validated and commanded to start execution.
3. Once the command has begun executing, the Service Provider publishes a `FunctionCommandStatusType` with a status `EXECUTING` and reason `SUCCEEDED` to notify the Service Consumer that the command has been validated and commanded to start.
4. If the Function has a defined `FunctionExecutionStatusReportType`, the Service Provider must publish a new instance with matching session ID as the associated `FunctionCommandType`. The `FunctionExecutionStatusReportType` must be updated by the Service Provider throughout the execution as dictated by the definitions of the command-specific attributes in the execution status report.

The command execution sequence is shown in Figure 19. This sequence holds until the command completes execution.

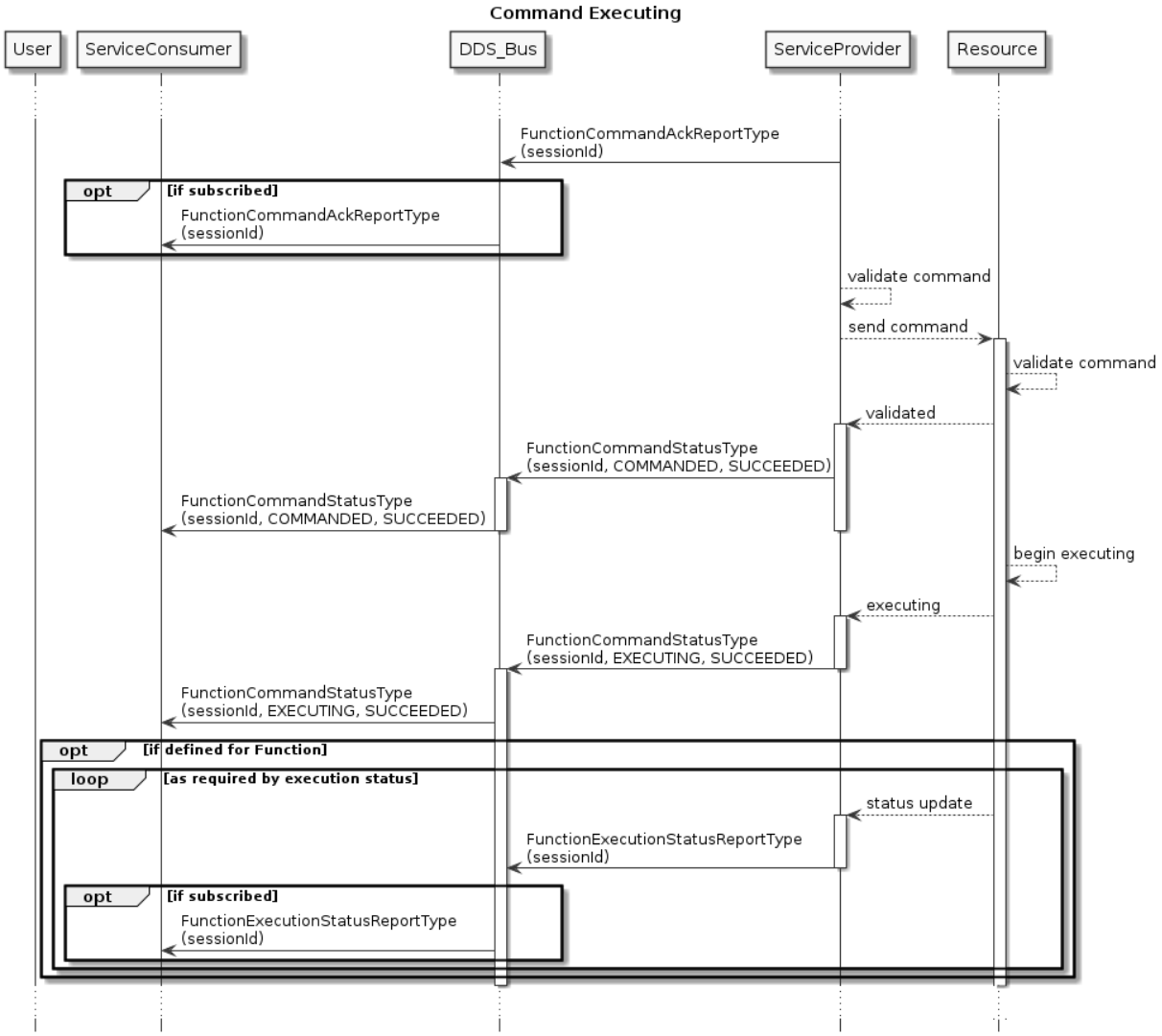

Figure 19: Beginning Sequence Diagram for a Command Execution.

The normal successful conclusion of a command being executed in some cases is initiated by the Service Consumer (an endless GlobalVector command concluded by canceling it) and in other cases is initiated by the Service Provider (a GlobalWaypoint commanded concluded by reaching the last waypoint). Unless otherwise explicitly stated, it is assumed the Service Provider will be able to identify the successful conclusion of a command. In the cases where commands are defined to be indeterminate the Service Consumer must cancel the command when the Service Consumer no longer desires the command to be executed.

5.1.4.2 Updating a Command An updated command is defined as a command with a source ID and session ID identical to the current command being processed by the Service Provider, but whose timestamp is newer than the current command. Only a command that is in a non-terminal state may be updated - otherwise, the Service Consumer must follow the normal command cleanup process and issue a new command with an updated unique session ID. When the Service Provider receives an updated command, it is required to take one of two possible actions:

1. If the current command is in a non-terminal state (`ISSUED`, `COMMANDED`, or `EXECUTING`), then the Service Provider publishes a `FunctionCommandStatusType` with a status `ISSUED` and reason `UPDATED`. The state machine then restarts and proceeds through the normal command flow detailed in 5.1.4. The Service Provider must consider the updated command as an entirely new command, resetting any internal state related to the command (e.g. a timer that keeps track of command timeout).
2. If the current command is in a terminal state (`COMPLETED`, `CANCELED`, or `FAILED`), then the updated command cannot be processed, and the Service Provider must publish a `FunctionCommandStatusType` with a status `FAILED` and follow the normal command cleanup process.

The flow control for command update is detailed below:

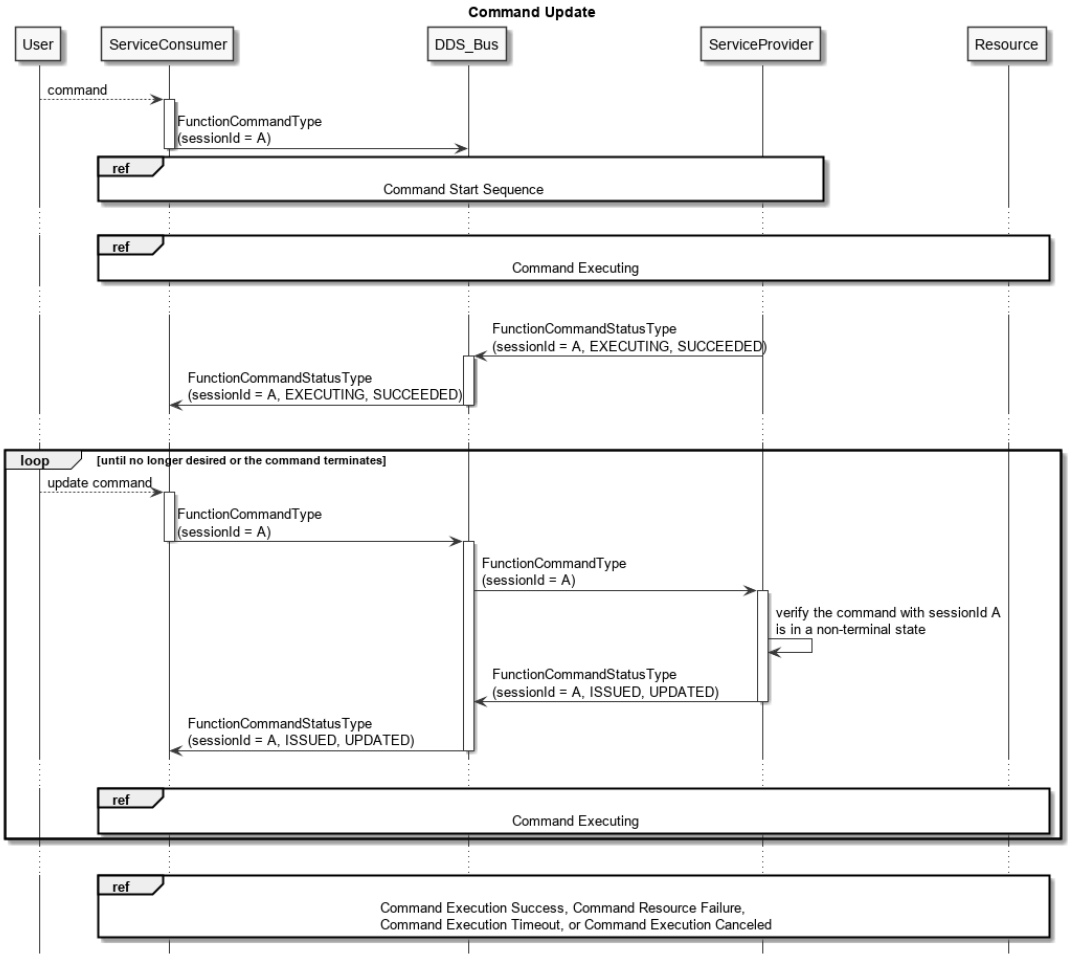

Figure 20: Sequence Diagram for Command Update.

5.1.4.3 Command Execution Success When the Service Provider determines a command has successfully completed, it must update the associated `FunctionCommandStatusType` with as status of `COMPLETED` and reason of `SUCCEEDED`. This signals to the Service Consumer that the command has completed successfully.

The Command Execution Success sequence is shown in Figure 21.

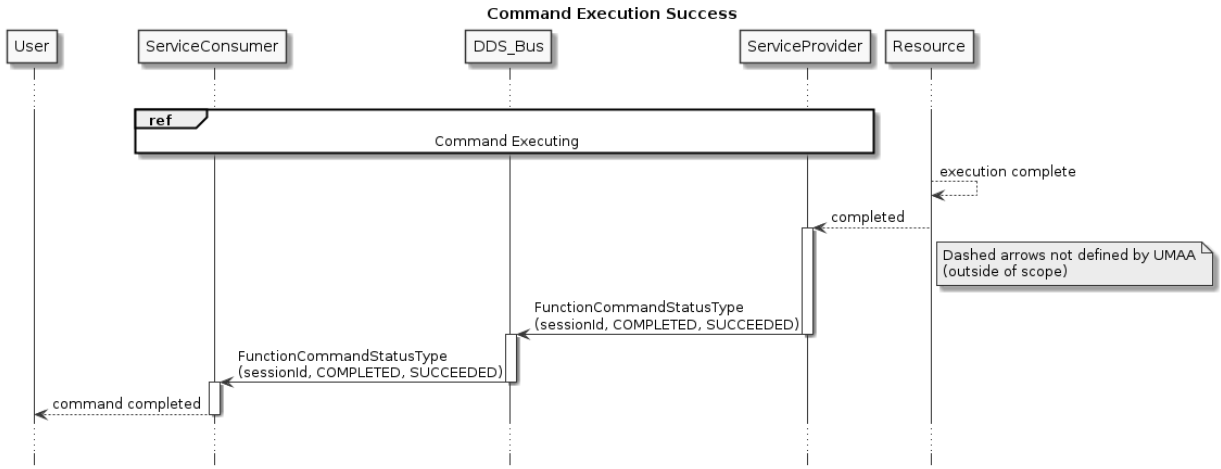

Figure 21: Sequence Diagram for a Command That Completes Successfully.

5.1.4.4 Command Execution Failure The command may fail to complete for any number of reasons including software errors, hardware failures, or unfavorable environmental conditions. The Service Provider may also reject a command for a number of reasons including inability to perform the task, malformed or out of range requests, or a command being interrupted by a higher priority process. In all cases, the Service Provider must publish a `FunctionCommandStatusType` with an identical `sessionID` as the originating `FunctionCommandType` with a status of `FAILED` and the reason that reflects the cause of the failure (`VALIDATION_FAILED, SERVICE_FAILED, OBJECTIVE_FAILED,` etc).

Figure 22 and Figure 23 provide examples where a command has failed.

In the first example, the backing Resource failed and the Service Provider is unable to communicate with it. In this case, the Service Provider will report a `FunctionCommandStatusType` with a status of `FAILED` and a reason of `RESOURCE_FAILED`. This is shown in Figure 22.

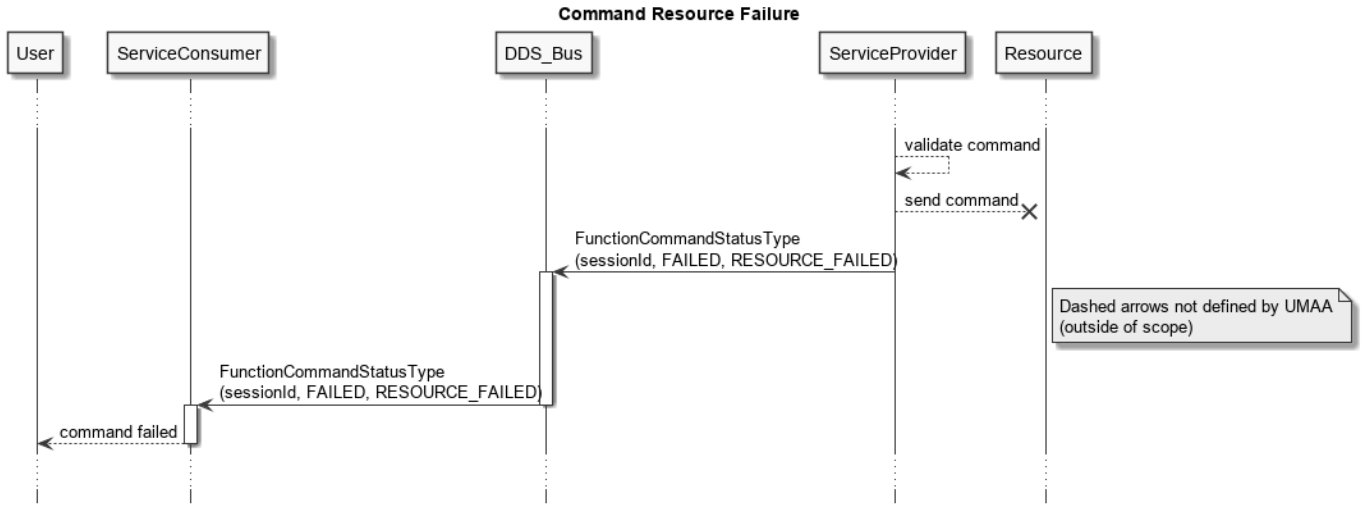

Figure 22: Sequence Diagram for a Command That Fails due to Resource Failure.

In the second example, the Resource takes too long to respond, so the Service Provider cancels the request and reports a `FunctionCommandStatusType` with a status of `FAILED` and a reason of `TIMEOUT`. This is shown in Figure 23.

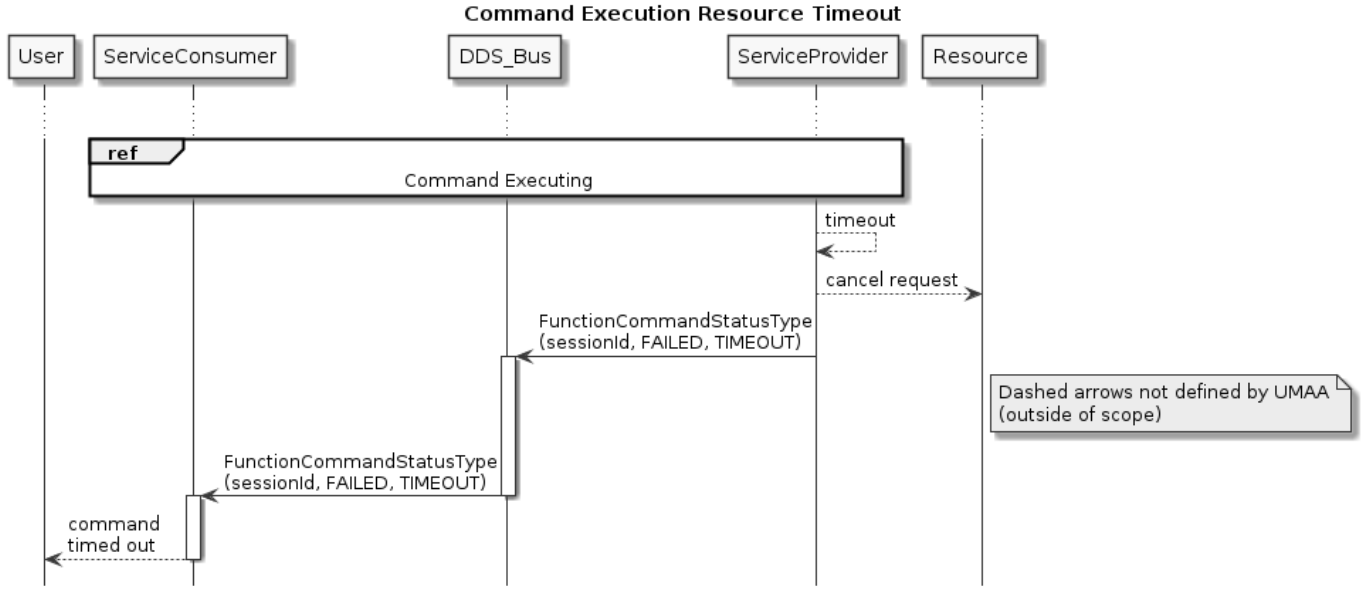

Figure 23: Sequence Diagram for a Command That Times Out Before Completing.

Other failure conditions will follow a similar pattern: when the failure is recognized, the Service Provider will publish a

`FunctionCommandStatusType` with a status of `FAILED` and a reason that reflect the cause of the failure.

5.1.4.5 Command Canceled The Service Consumer may decide to cancel the command before processing is finished. To signal a desire to cancel a command, the Service Consumer disposes of the existing `FunctionCommandType` from the DDS bus before the execution is complete. When notified of the command disposal, and if the Service Provider is able to cancel the command, it should respond to the Service Consumer with a `FunctionCommandStatusType` with both the status and reason as `CANCELED`. At this point, the DDS bus should dispose of the `FunctionCommandStatusType`, the `FunctionCommandAckReportType` and, (if defined for the Function service) the `FunctionExecutionStatusReportType`. This is shown in Figure 24. If the command cannot be canceled, then the Service Provider can continue to update the command status until the execution is completed. Reporting will include `FunctionCommandStatusType` with a status of `COMPLETED` and a reason of `SUCCEEDED`. Then, the DDS bus should dispose of the `FunctionCommandStatusType`, the `FunctionCommandAckReportType`, and (if defined for the Function service) the `FunctionExecutionStatusReportType`.

There is no new, unique, or specific status message response to a cancel command from the Service Provider. The cancel command status can be inferred through the corresponding `FunctionCommandStatusType` status and reason updates.

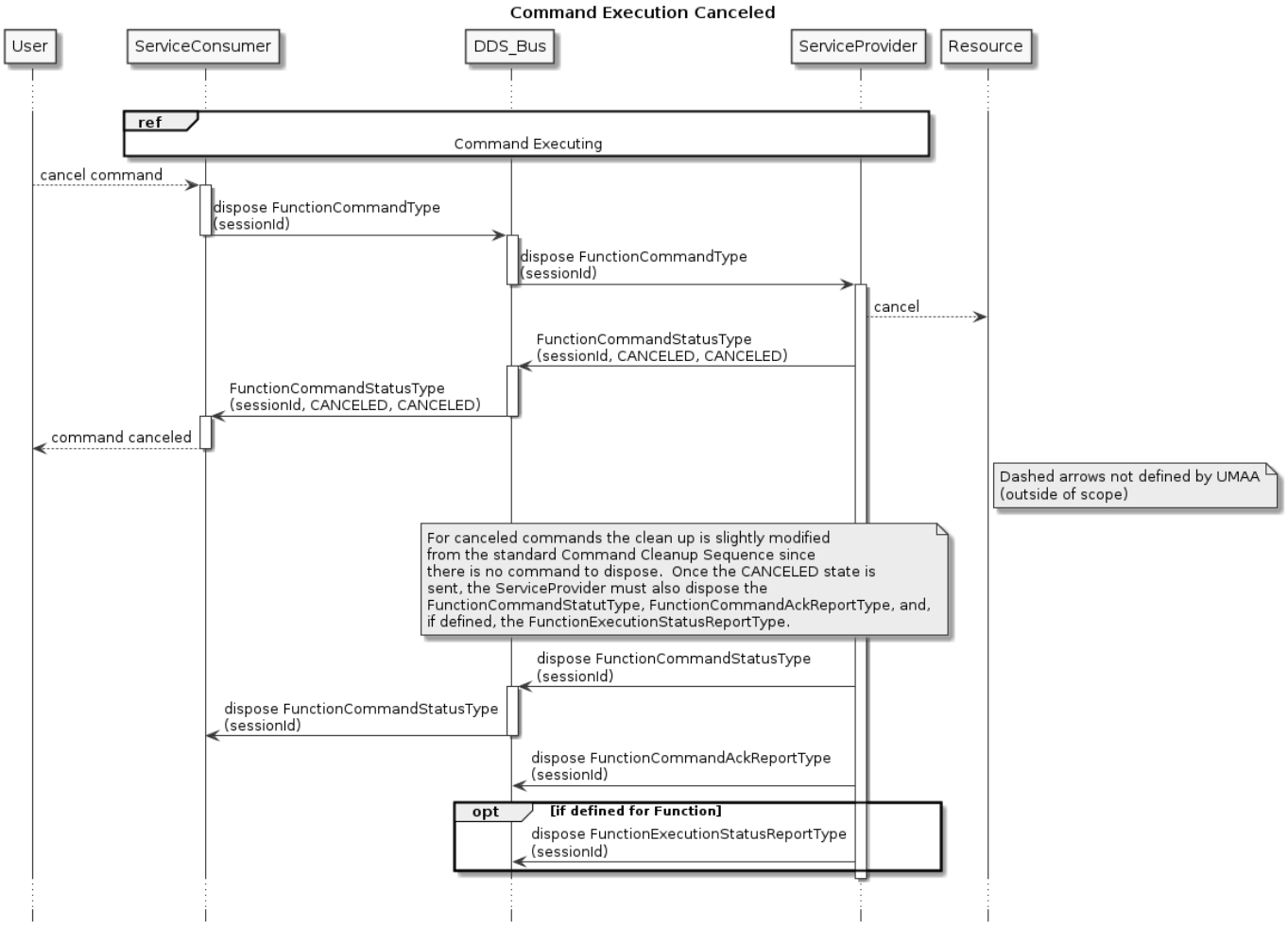

Figure 24: Sequence Diagram for a Command That is Canceled by the Service Consumer Before the Service Provider can Complete It.

5.1.5 Command Cleanup

The Service Consumer and Service Provider are responsible for disposing of corresponding data that is published to the DDS bus when the command is no longer active. With the exception of a canceled command, the signal that a `FunctionCommandType` can be disposed is when the `FunctionCommandStatusType` reports a terminal state (`COMPLETED` or `FAILED`)[3]. In turn, the

[3]While `CANCELED` is also a terminal state, the `CANCELED` command cleanup is handled specially as part of the cancelling sequence and, as such, does not need to be handled here.

signal that a `FunctionCommandStatusType`, `FunctionCommandAckReportType`, and (if defined for the Function service) the `FunctionExecutionStatusReportType` can be disposed is when the corresponding `FunctionCommandType` has been disposed. This is shown in Figure 25.

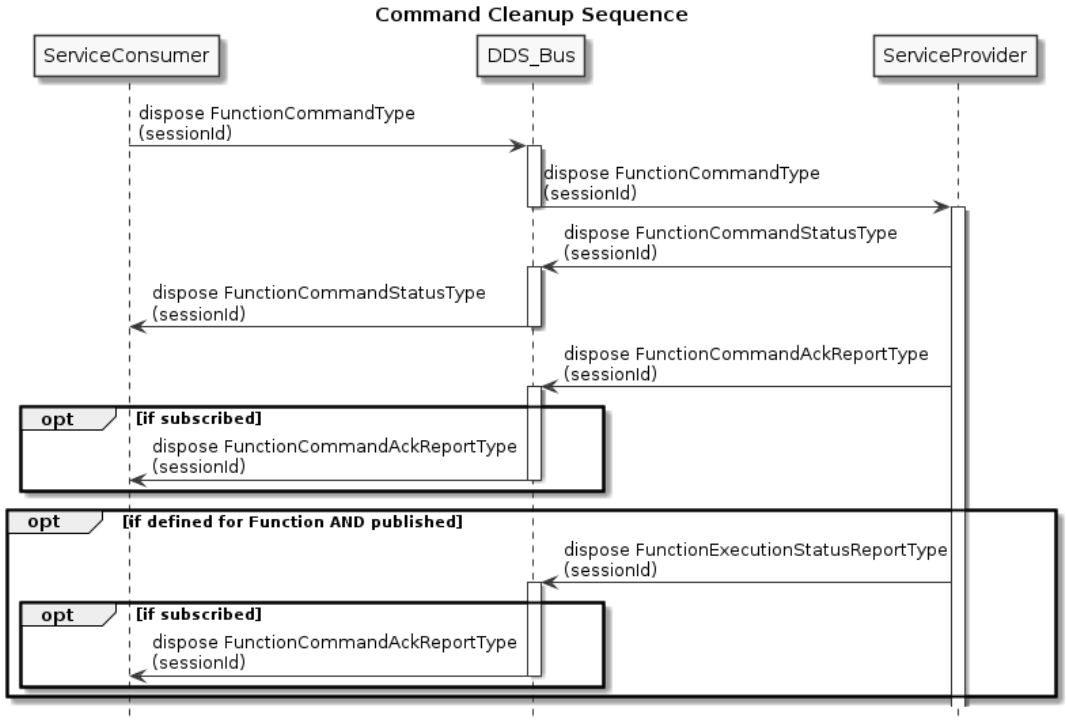

Figure 25: Sequence Diagram Showing Cleanup of the Bus When a Command Has Been Completed and the Service Consumer No Longer Wishes to Maintain the Commanded State.

5.1.6 Command Shutdown Sequence

As part of shutdown, both the Service Provider and Service Consumer are required to perform a shutdown sequence. This shutdown cleans up resources on the DDS bus and informs the system that the Service Provider and Service Consumer are no longer available.

The Service Provider and Service Consumer can shut down in any order. The sequence diagram is shown in Figure 26.

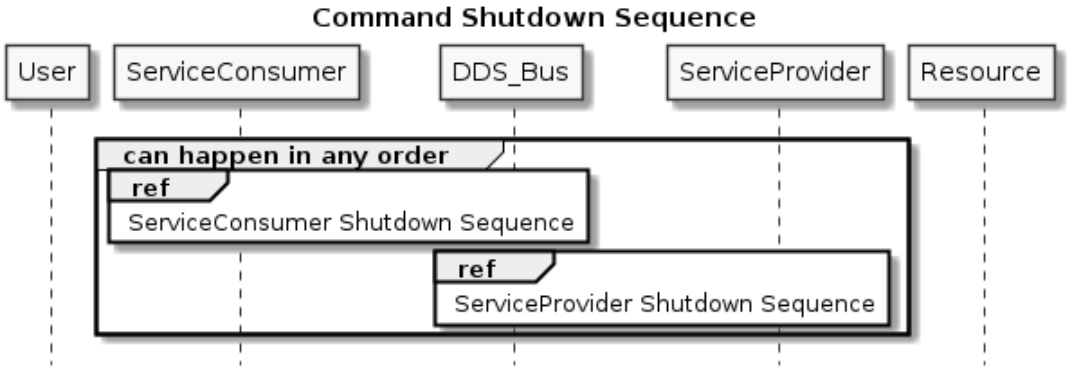

Figure 26: Sequence Diagram for Command Shutdown.

5.1.6.1 Service Provider Shutdown Sequence During shutdown, the Service Provider is required to fail any incomplete requests and then unregisters as a publisher of the `FunctionCommandStatusType`, `FunctionCommandAckReportType`, and (if defined for the Function service) the `FunctionExecutionStatusReportType`.

The Service Provider is also required to unsubscribe from the `FunctionCommandType`.

The Service Provider Shutdown sequence is shown in Figure 27.

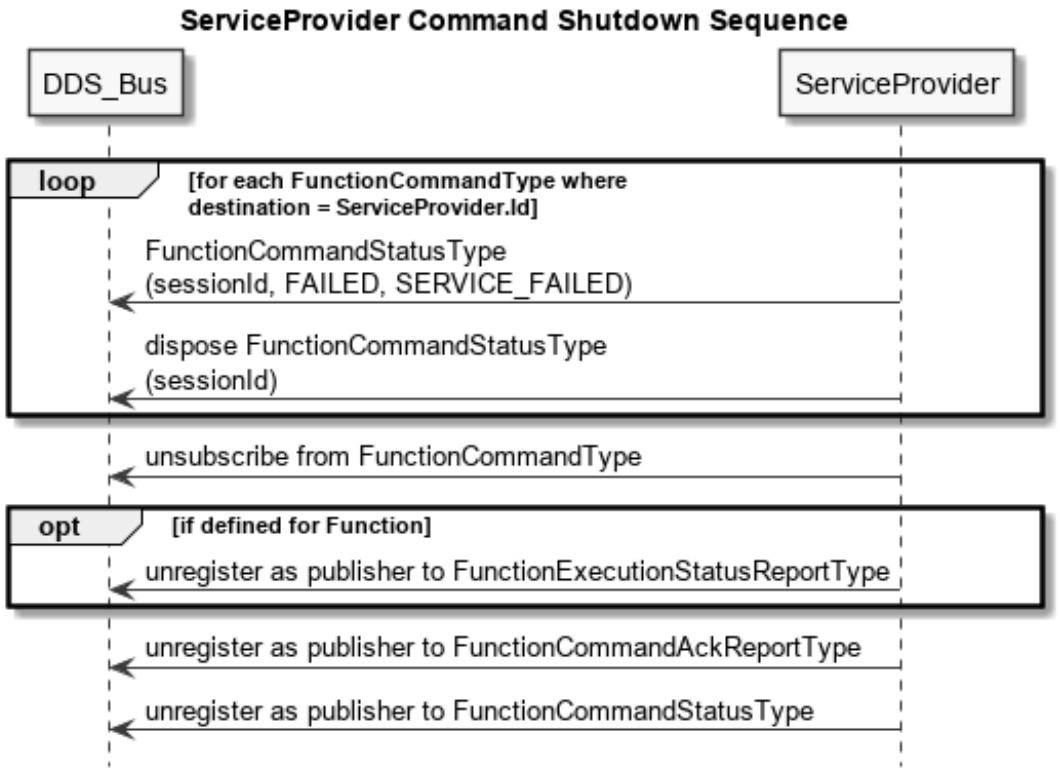

Figure 27: Sequence Diagram for Command Shutdown for Service Providers.

5.1.6.2 Service Consumer Shutdown Sequence During shutdown, the Service Consumer is required to cancel any incomplete requests and then unregister as a publisher of the `FunctionCommandType`.

The Service Consumer is also required to unsubscribe from the `FunctionCommandStatusType`, the `FunctionCommandAckReportType` if subscribed, and the `FunctionExecutionStatusReportType` if defined for the Function service and subscribed.

The Service Consumer Shutdown sequence is shown in Figure 28.

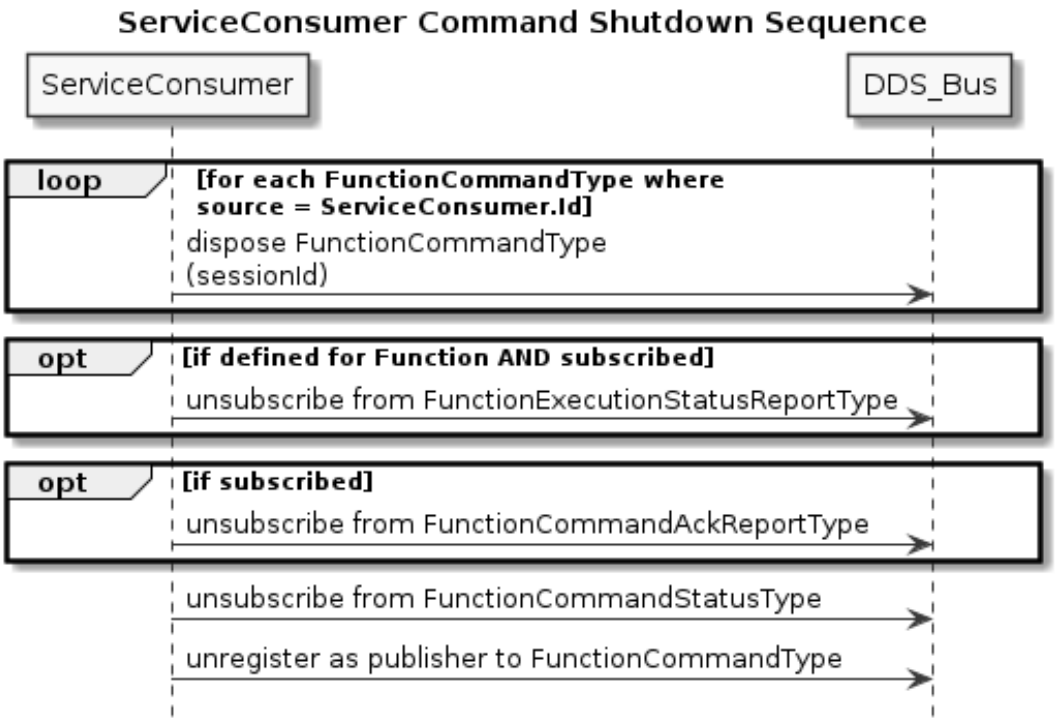

Figure 28: Sequence Diagram for Command Shutdown for Service Consumers.

5.2 Request / Reply

This section defines the flow of control for request/reply over the DDS bus. A request/reply is used to obtain data or status from a specific Service Provider.

A Service Provider is required to reply to all requests it receives. In the case of requests with no query data, this is accomplished via a DDS subscribe. In the case of a request with associated query data, a message with the query data must be published by the requester. To direct a request at a specific Service Provider or set of services, UMAA defines a `destination` GUID as part of requests.

The sequence diagrams in Sections 29 through 33 demonstrate different exchanges between a Service Consumer and Service Provider. Within the diagrams, the dashed arrows represent implementation-specific communications that are outside of UMAA's scope. Additionally, these sequence diagrams are examples of one possible implementation. Other implementations may have different communication patterns between the Service Provider and the Resource, or be implemented completely within the Service Provider process itself (no external Resource). However, in all implementations, UMAA-defined exchanges with the DDS bus between the Service Consumer and Service Provider must happen in the order shown within the sequence diagrams.

5.2.1 Request/Reply without Query Data

Figure 29 shows the sequence of exchanges in the case where there is no specific query data (i.e., the service is always just providing the current data to the bus).

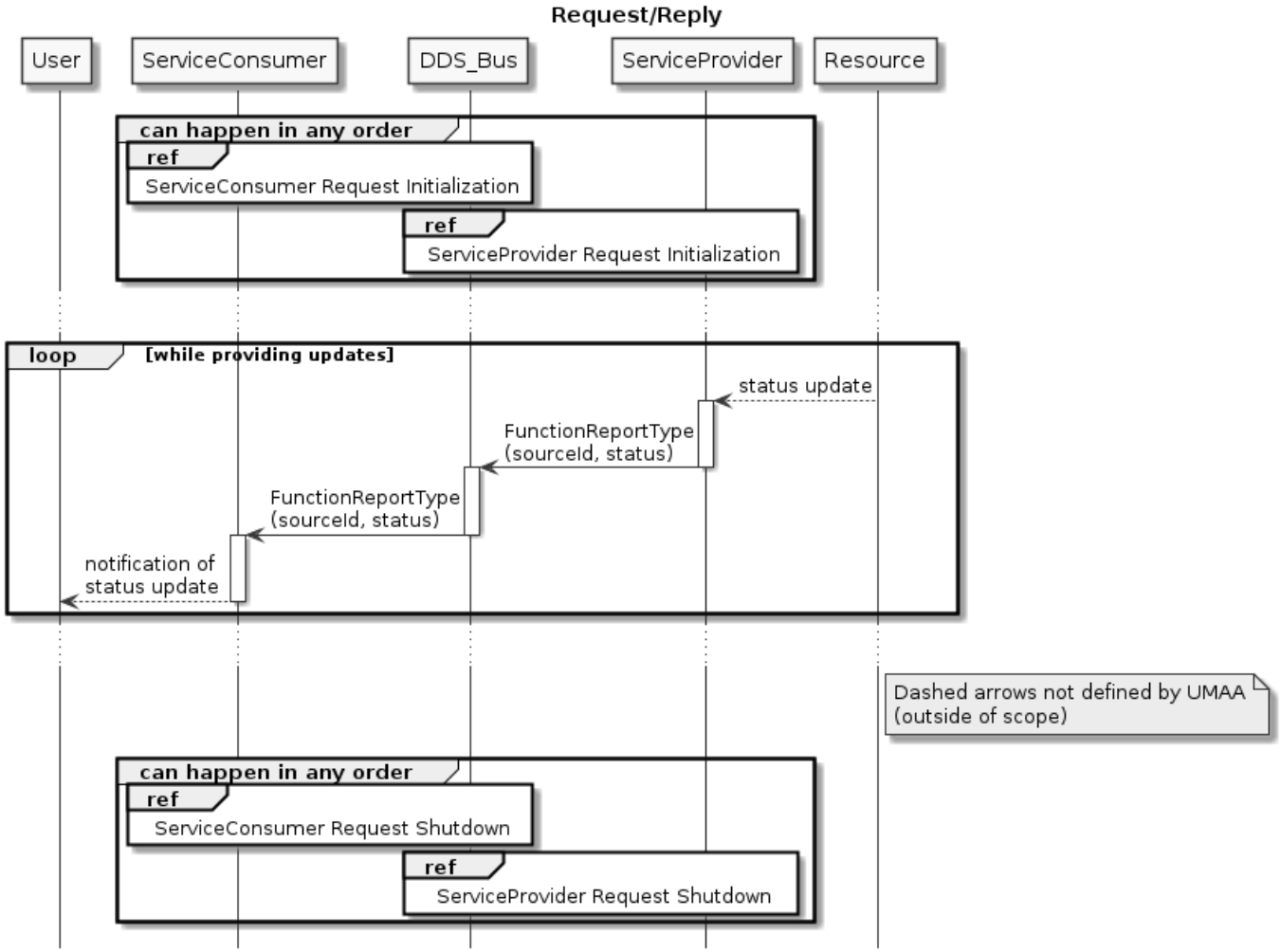

Figure 29: Sequence Diagram for a Request/Reply for Report Data That Does Not Require any Specific Query Data.

5.2.1.1 Service Provider Startup Sequence The Service Provider registers as a publisher of `FunctionReportTypes` to be able to respond to requests. The Service Provider must also handle reports that exist on the bus from a previous instantiation, either by providing an immediate update or, if the status is unrecoverable, disposing of the old `FunctionReportType`. This is shown in Figure 30.

As `FunctionReportType` updates are required (either through event-driven changes or periodic updates), the Service Provider publishes the updated data. The DDS bus will deliver the updates to the Service Consumer.

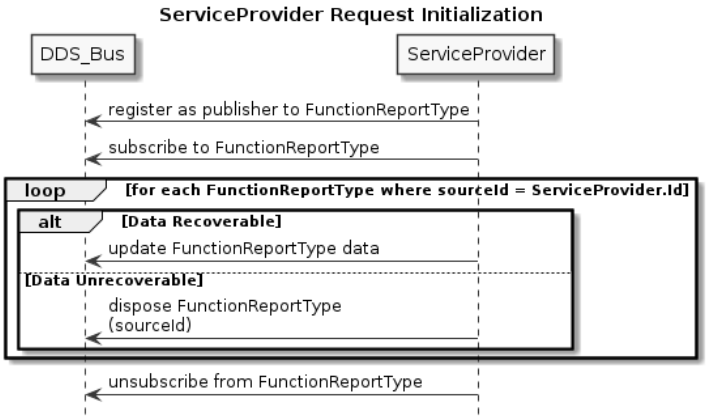

Figure 30: Sequence Diagram for Initialization of a Service Provider to Provide `FunctionReportType`s.

5.2.1.2 Service Consumer Startup Sequence The Service Consumer subscribes to the `FunctionReportType` to signal an outstanding request for updates. This is shown in Figure 31.

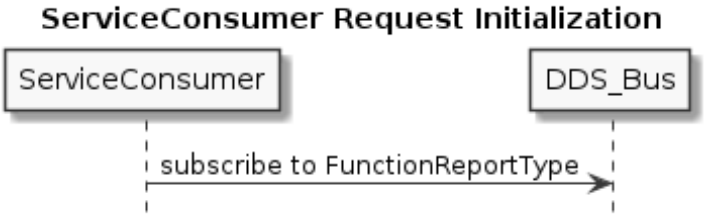

Figure 31: Sequence Diagram for Initialization of a Service Consumer to Request `FunctionReportType`s.

5.2.1.3 Service Provider Shutdown To no longer provide `FunctionReportType`s, the Service Provider disposes of the `FunctionReportType` and unregisters as a publisher of the data (shown in Figure 32).

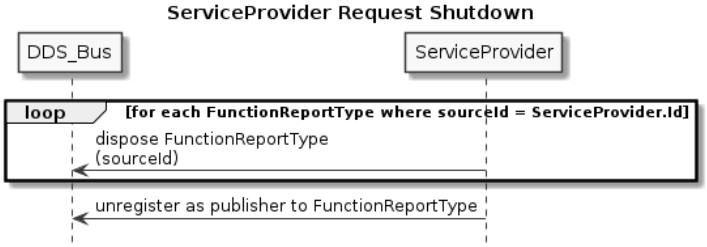

Figure 32: Sequence Diagram for Shutdown of a Service Provider.

5.2.1.4 Service Consumer Shutdown To no longer request `FunctionReportType`s, the Service Consumer unsubscribes from `FunctionReportType` (shown in Figure 33).

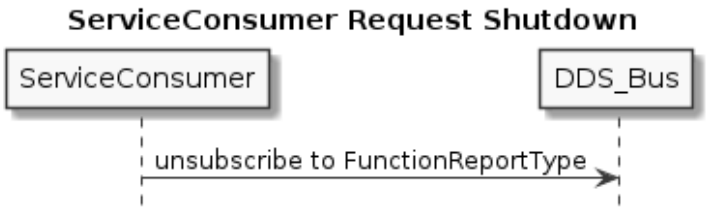

Figure 33: Sequence Diagram for Shutdown of a Service Consumer.

5.2.2 Request/Reply with Query Data

Currently, UMAA does not define any request/reply interactions with query data, but it is expected that some will be defined. When defined, this section will be expanded to describe how they must be used.

6 Situational Awareness (SA) Services and Interfaces

6.1 Services and Interfaces

The interfaces in the following subsections describe how each UCS-UMAA topic is defined by listing the name, namespace, and member attributes. The "name" corresponds with the message name of a given service interface. The "namespace" defines the scope of the "name" where similar commands are grouped together. The "member attributes" are fields that can be populated with differing data types, e.g. a generic "depth" attribute could be populated with a double data value. Note that using a UCS-UMAA "Topic Name" requires using the fully-qualified namespace plus the topic name.

Each interface topic is referenced by a UMAA service and is defined as either an input or output interface.

Attributes ending in one or more asterisk(s) denote the following:
* = Key (annotated with @key in IDL file; vendors may use different notation to indicate a key field)
† = Optional (annotated with @optional in IDL file; vendors may use different notation to indicate an optional field)

Optional fields should be handled as described in the UMAA Compliance Specification.

Commands issued on the DDS bus must be treated as if they are immutable in UMAA and, therefore, if updated (treated incorrectly as mutable), the resulting service actions are indeterminate and flow control protocols are no longer guaranteed.

Operations without DDS Topics

The following operations are all handled directly by DDS. They are marked in the operations tables with a ⊕.

query<...> - All query operations are used to retrieve the correlated report message. For UMAA, this operation is accomplished through subscribing to the appropriate DDS topic.

cancel<...> - All cancel operations are used to nullify the current command. For UMAA, this operation is accomplished through the DDS dispose action on the publisher.

report<...>CancelCommandStatus - All cancel reports are included here to show completeness of the MDE model mapping to UMAA. For UMAA, this operation is not used. Instead, the cancel status is inferred from the associated command status. If the cancel command is successful, the corresponding command will fail with a command status and reason of CANCELED. If the corresponding command status reports COMPLETED, then this cancel command has failed.

6.1.1 ContactCOLREGSClassificationStatus

The purpose of this service is to provide the vehicles and/or the operator with the COLREGS classification to existing contacts (manmade objects) for situational awareness in the operational area. If a contact is dropped from the contact service, it should be removed from this service as well.

Table 8: ContactCOLREGSClassificationStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryContactCOLREGSClassification⊕	reportContactCOLREGSClassification

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.1.1 reportContactCOLREGSClassification

Description: This operation is a response to retrieve the current contact classification data.

Namespace: UMAA::SA::ContactCOLREGSClassificationStatus

Topic: ContactCOLREGSClassificationReport

Data Type: ContactCOLREGSClassificationReportType

Table 9: ContactCOLREGSClassificationReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
confidence	Percent	The confidence in the COLREGS classification of the contact.
colregsClassification*	COLREGSClassificationEnumType	The COLREGS classification of the contact.
contactID*	NumericGUID	An identifier of the contact.

6.1.2 ContactReport

The purpose of this service is to provide contact (object) related data to the vehicles and/or the operator for situational awareness in the operational area.

Table 10: ContactReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryContact⊕	reportContact

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.2.1 reportContact

Description: This operation is a response to retrieve the current contact data.

Namespace: UMAA::SA::ContactReport

Topic: ContactReport

Data Type: ContactReportType

Table 11: ContactReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
contacts→setID	LargeSet<ContactType>	A list of object structures. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::SA::ContactReport::ContactReportContactsSetElement.

6.1.3 ContactVisualClassificationStatus

The purpose of this service is to provide the vehicles and/or the operator with the visual classification to existing contacts (manmade objects) for situational awareness in the operational area. If a contact is dropped from the contact service, it should be removed from this service as well.

Table 12: ContactVisualClassificationStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryContactVisualClassification⊕	reportContactVisualClassification

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.3.1 reportContactVisualClassification

Description: This operation is a response to retrieve the current contact classification data.

Namespace: UMAA::SA::ContactVisualClassificationStatus

Topic: ContactVisualClassificationReport

Data Type: ContactVisualClassificationReportType

Table 13: ContactVisualClassificationReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
confidence	Percent	The confidence in the visual classification of the contact.
contactID*	NumericGUID	An identifier of the contact.
visualClassification*	VisualClassificationEnumType	The visual classification of the contact.

6.1.4 DateTimeStatus

The purpose of this service is to allow participants to access the current vehicle time.

Table 14: DateTimeStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryDateTime⊕	reportDateTime

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.4.1 reportDateTime

Description: This operation is used to report the current vehicle time. The timestamp of the message provides the current time.

Namespace: UMAA::SA::DateTimeStatus

Topic: DateTimeReport

Data Type: DateTimeReportType

Table 15: DateTimeReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		

6.1.5 ECEFPoseStatus

The purpose of this service is to report the current position and orientation of the vehicle in the earth-centered, earth-fixed reference frame.

Table 16: ECEFPoseStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryECEFPose⊕	reportECEFPose

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.5.1 reportECEFPose

Description: This operation is used to report the current position and orientation of the vehicle in the earth-centered, earth-fixed reference frame.

Namespace: UMAA::SA::ECEFPoseStatus

Topic: ECEFPoseReport

Data Type: ECEFPoseReportType

Table 17: ECEFPoseReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
attitude	Orientation3DNEDType	The orientation (roll, pitch, yaw) of the vehicle.
attitudeCovariance†	CovarOrientationType	The current error covariance value of the attitude data.
positionCovariance†	CovariancePositionECEFType	The current error covariance value of the position data.
xPosition	Distance	The x position of the vehicle.
yPosition	Distance	The y position of the vehicle.
zPosition	Distance	The z position of the vehicle.

6.1.6 GlobalPoseStatus

The purpose of this service is to report the current position and orientation of the vehicle in the global coordinate system.

Table 18: GlobalPoseStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryGlobalPose⊕	reportGlobalPose

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.6.1 reportGlobalPose

Description: This operation is used to report the current position and orientation of the vehicle in the global coordinate system.

Namespace: UMAA::SA::GlobalPoseStatus

Topic: GlobalPoseReport

Data Type: GlobalPoseReportType

Table 19: GlobalPoseReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
altitude†	MSLAltitude	The current orthometric height above the Geoid (Mean Sea Level) of the vehicle.
altitudeAGL†	DistanceAGL	The current height above ground level of the vehicle.
altitudeASF†	DistanceASF	The current height above the sea floor of the vehicle.
altitudeGeodetic†	GeodeticAltitude	The current geodetic height above the ellipsoid of the vehicle.
attitude	Orientation3DNEDType	The current orientation (roll, pitch, yaw) of the vehicle.
attitudeCovariance†	CovarOrientationType	The current error covariance value of the attitude data.
course	CourseTrueNorth	The current course angle used for the vehicle.
depth†	DistanceBSL	The current depth of the maritime vehicle.
navigationSolution	NavigationSolutionEnumType	The desired navigation solution (estimated or measured).
position	GeoPosition2D	The current position of the vehicle.
positionCovariance†	CovariancePositionNEDType	The current error covariance value of the position data.

6.1.7 MagneticVariationSpecs

The purpose of this service is to query and report the magnetic deviation in the global coordinate system.

Table 20: MagneticVariationSpecs Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMagneticVariationSpecs⊕	reportMagneticVariationSpecs

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.7.1 reportMagneticVariationSpecs

Description: This operation is used to report the geomagnetic deviation being used by the vehicle.

Namespace: UMAA::SA::MagneticVariationSpecs

Topic: MagneticVariationSpecsReport

Data Type: MagneticVariationSpecsReportType

Table 21: MagneticVariationSpecsReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
magneticDeviation	sequence<MagneticDeviationType> max size = 360	The location specific magnetic variation.

6.1.8 MagneticVariationStatus

The purpose of this service is to query and report the magnetic declination in the global coordinate system.

Table 22: MagneticVariationStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMagneticVariation⊕	reportMagneticVariation

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.8.1 reportMagneticVariation

Description: This operation is used to report the geomagnetic declination being used by the vehicle.

Namespace: UMAA::SA::MagneticVariationStatus

Topic: MagneticVariationReport

Data Type: MagneticVariationReportType

Table 23: MagneticVariationReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
magneticDeclination	MagneticVariation	The difference in angle between magnetic north and true north on the horizontal plane.

6.1.9 SeaStateReport

The purpose of this service is to report the current sea state at the operational area.

Table 24: SeaStateReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
querySeaState⊕	reportSeaState

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.9.1 reportSeaState

Description: This operation is used to report the current sea state from the sea state sensor on board of the vehicle.

Namespace: UMAA::SA::SeaStateReport

Topic: SeaStateReport

Data Type: SeaStateReportType

Table 25: SeaStateReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
state	SeaStateEnumType	The WMO sea state.

6.1.10 SpeedStatus

This service has the responsibility of reporting the instantaneous speed of the vehicle.

Table 26: SpeedStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
querySpeed⊕	reportSpeed

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.10.1 reportSpeed

Description: This operation is used to report the data parameters of the VehicleSpeed service. This message is used to provide the receiver the speed of the vehicle.

Namespace: UMAA::SA::SpeedStatus

Topic: SpeedReport

Data Type: SpeedReportType

Table 27: SpeedReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
engineRPM†	EngineSpeed	The engine RPM of the vehicle.
mode†	VehicleSpeedModeEnumType	Describes the speed mode.
speedOverGround†	GroundSpeed	The value of speed over the ground of the vehicle.
speedThroughAir†	IndicatedAirspeed	The value of speed through air of the vehicle.
speedThroughWater†	SpeedLocalWaterMass	The value of speed through water of the vehicle.

6.1.11 TranslationalShipMotionStatus

This service has the responsibility of reporting the translational motion of the vehicle.

Table 28: TranslationalShipMotionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryTranslationalShipMotion⊕	reportTranslationalShipMotion

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.11.1 reportTranslationalShipMotion

Description: This operation is used to report the the current translational motion of the vehicle.

Namespace: UMAA::SA::TranslationalShipMotionStatus

Topic: TranslationalShipMotionReport

Data Type: TranslationalShipMotionReportType

Table 29: TranslationalShipMotionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
heave†	Down	The linear vertical motion of the ship. Positive heave is down (toward the water bottom); a vehicle increasing draft is moving in the positive heave direction.
surge†	Forward	The linear longitudinal motion of the ship. Positive surge is forward of the vehicle.
sway†	Left	The linear transverse motion of the ship. Positive sway is to the port side of the vehicle.

6.1.12 VelocityStatus

The purpose of this service is to report the linear velocity and rotational rate of the vehicle. The velocity state of a rigid body is defined by six parameters that represent the velocity components of a point in the body that is coincident with the origin of the fixed reference and the instantaneous angular velocity components. The reference system is defined as a fixed coordinate system that at that instant, aligns with the vehicle or system coordinate system.

Table 30: VelocityStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryVelocity⊕	reportVelocity

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.12.1 reportVelocity

Description: This operation is used to report the current linear velocity and rotational rate of the vehicle.

Namespace: UMAA::SA::VelocityStatus

Topic: VelocityReport

Data Type: VelocityReportType

Table 31: VelocityReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
attitudeRate	OrientationVel3D	The current rotational rate of the vehicle.
attitudeRateCovariance†	CovarAttitudeRateType	The current error covariance value of the attitude rate of change data.
velocity	Velocity3DPlatformNEDType	The current linear velocity (in the NED reference frame) of the vehicle.
velocityCovariance†	CovarianceVelocityType	The current error covariance value of the velocity data.

6.1.13 WaterCurrentStatus

The purpose of this service is to provide information about the state of the water current.

Table 32: WaterCurrentStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryWaterCurrent⊕	reportWaterCurrent

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.13.1 reportWaterCurrent

Description: This operation is used to report the status of the water current.

Namespace: UMAA::SA::WaterCurrentStatus

Topic: WaterCurrentReport

Data Type: WaterCurrentReportType

Table 33: WaterCurrentReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
currentDrift	Speed	The speed of the water current.
currentSet	HeadingTrueNorthAngle	The heading of the water current relative to true north.

6.1.14 WaterspaceStatus

Defines the current active waterspace zones.

Table 34: WaterspaceStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryWaterspace⊕	reportWaterspace

See Section 6.1 for an explanation of the inputs and outputs marked with a ⊕.

6.1.14.1 reportWaterspace

Description: This operation is used to report the current waterspace zones.

Namespace: UMAA::SA::WaterspaceStatus

Topic: WaterspaceReport

Data Type: WaterspaceReportType

Table 35: WaterspaceReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
zones→setID	LargeSet<ZoneType>	The zones described by the waterspace. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::SA::WaterspaceStatus::WaterspaceReportZonesSetElement.

6.2 Common Data Types

Common data types define DDS types that are referenced throughout the UMAA model. These DDS types are considered common because they can be re-used as the data type for many attributes defined in service interface topics, interface topics, and other common data types. These data types are not intended to be directly published to/subscribed as DDS topics.

6.2.1 UCSMDEInterfaceSet

Namespace: UMAA::UCSMDEInterfaceSet

Description: Defines the common UCSMDE Interface Set Message Fields.

Table 36: UCSMDEInterfaceSet Structure Definition

Attribute Name	Attribute Type	Attribute Description
timeStamp	DateTime	The time at which the data is valid.

6.2.2 UMAACommand

Namespace: UMAA::UMAACommand

Description: Defines the common UMAA Command Message Fields.

Table 37: UMAACommand Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UCSMDEInterfaceSet		
source*	NumericGUID	The unique identifier of the originating source of the command interface.
destination*	NumericGUID	The unique identifier of the destination of the command interface.
sessionID*	NumericGUID	The identifier of the session.

6.2.3 UMAAStatus

Namespace: UMAA::UMAAStatus

Description: Defines the common UMAA Status Message Fields.

Table 38: UMAAStatus Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UCSMDEInterfaceSet		
source*	NumericGUID	The unique identifier of the originating source of the status interface.

6.2.4 UMAACommandStatusBase

Namespace: UMAA::UMAACommandStatusBase

Description: Defines the common UMAA Command Status Base Message Fields.

Table 39: UMAACommandStatusBase Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UCSMDEInterfaceSet		
source*	NumericGUID	The unique identifier of the originating source of the command status interface.
sessionID*	NumericGUID	The identifier of the session.

6.2.5 UMAACommandStatus

Namespace: UMAA::UMAACommandStatus

Description: Defines the common UMAA Command Status Message Fields.

Table 40: UMAACommandStatus Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
commandStatus	CommandStatusEnumType	The status of the command.
commandStatusReason	CommandStatusReasonEnumType	The reason for the status of the command.
logMessage	StringLongDescription	Human-readable description related to response. Systems should not parse or use any information from this for processing purposes.

6.2.6 DateTime

Namespace: UMAA::Measurement::DateTime

Description: Describes an absolute time. Conforms with POSIX time standard (IEEE Std 1003.1-2017) epoch reference point of January 1st, 1970 00:00:00 UTC.

Table 41: DateTime Structure Definition

Attribute Name	Attribute Type	Attribute Description
seconds	DateTimeSeconds	The number of seconds offset from the standard POSIX (IEEE Std 1003.1-2017) epoch reference point of January 1st, 1970 00:00:00 UTC.
nanoseconds	DateTimeNanoSeconds	The number of nanoseconds elapsed within the current DateTimeSecond.

6.2.7 AltitudeAGLType

Namespace: UMAA::Common::Measurement::AltitudeAGLType

Description: The height above ground level.

Table 42: AltitudeAGLType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeAGL	DistanceAGL	Specifies the distance above ground level.

6.2.8 AltitudeASFType

Namespace: UMAA::Common::Measurement::AltitudeASFType

Description: The height above sea floor.

Table 43: AltitudeASFType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeASF	DistanceASF	The height above the sea floor.

6.2.9 AltitudeGeodeticType

Namespace: UMAA::Common::Measurement::AltitudeGeodeticType

Description: The geodetic height above the ellipsoid.

Table 44: AltitudeGeodeticType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeGeodetic	GeodeticAltitude	The altitude above the reference ellipsoid.

6.2.10 AltitudeMSLType

Namespace: UMAA::Common::Measurement::AltitudeMSLType

Description: The orthometric height above the Geoid (Mean Sea Level).

Table 45: AltitudeMSLType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeMSL	MSLAltitude	The orthometric height above the Geoid (Mean Sea Level).

6.2.11 ContactType

Namespace: UMAA::SA::ContactReport::ContactType

Description: This structure is used to report the contact characteristics.

Table 46: ContactType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeAGL†	DistanceAGL	The height above ground level of the entity.
altitudeASF†	DistanceASF	The height above the sea floor of the entity.
altitudeGeodetic†	GeodeticAltitude	The geodetic height above the ellipsoid of the entity.
altitudeMSL†	MSLAltitude	The orthometric height above the Geoid (Mean Sea Level) of the entity.
callSign†	StringShortDescription	Designator assigned by licensing authority, or name of vehicle.
confidence†	Percent	An indication of how confident the source is in the report.
contactName	StringShortDescription	The name of the contact.
course†	CourseTrueNorth	The course of the contact.
depth†	DistanceBSL	The distance below sea level of the entity.
heading†	HeadingTrueNorthAngle	The heading of the contact.
height†	Distance	Height of the contact, intended to be the largest dimension orthogonal to length in the vertical plane.
length†	Distance	The length of the contact, intended to be the longest dimension of the shape.
MMSINumber†	MMSI	Nine digit numbers used by maritime digital selective calling (DSC), automatic identification systems (AIS) and certain other equipment to uniquely identify a ship or a coast radio station.
position	GeoPosition2D	The position of the object.
positionCovariance†	CovariancePositionNEDType	Current error covariance value of the object position.
quality†	Percent	The quality of the relevant track. May represent a moving or fading average of the growth rate of the track and is useful for determining the track confidence or when to drop the track entirely.
SIDC†	StringShortDescription	A numeric code based on a hierarchical structure that provides the elements required to construct the basic symbol (defined in Appendix A of MIL-STD-2525D).
source	sequence<NumericGUID> max size = 32	A list of source contacts.
sourceIndicator	SourceIndicatorEnumType	Describes the data source indicator.
specialManeuverIndicator	SpecialManeuverIndicatorEnumType	The special maneuver indicator.
speedOverGround	GroundSpeed	The speed over the ground of the contact.
timeFirstAcquired	DateTime	The time the contact was acquired.
timeLost†	DateTime	The time the contact was lost. For active tracks with continued detections, this value will not be provided. In the event detections are lost but the track is coasted, this value will identify the start time of coasting.

Attribute Name	Attribute Type	Attribute Description
width†	Distance	Width of the contact, intended to be the largest dimension orthogonal to length in the horizontal plane.
featureID*	NumericGUID	An identifier of the object.

6.2.12 CovarAttitudeRateType

Namespace: UMAA::Common::Measurement::CovarAttitudeRateType

Description: Contains variances and covariances for random variables representing estimates of Observable values. Given an Observable with true value x the estimated value $\hat{x}$ is taken to be a random variable with sample space and distribution determined by the random processes which contribute to the calculation of the sampled Observable estimation. The ObservableError for the Observable is defined to be the Mean Squared Error, $\text{MSE} = E[(\hat{x} - x)^2]$. The MSE is equal to the sum of the variance of $\hat{x}$ and the square of the estimation bias: $\text{MSE} = \text{Var}[\hat{x}] + \text{Bias}[\hat{x}]^2 = E[(\hat{x} - E[\hat{x}])^2] - E[\hat{x} - x]^2$. The variance of $\hat{x}$ is equal to the MSE, and provides an estimate of the ObservableError, if and only if the underlying estimation process is unbiased.

Table 47: CovarAttitudeRateType Structure Definition

Attribute Name	Attribute Type	Attribute Description
rpRp	CovarOrientationRate	Pitch-Pitch attitude rate error covariance with units of radians squared per second squared.
rpRy†	CovarOrientationRate	Pitch-Yaw attitude rate error covariance with units of radians squared per second squared.
rrRp†	CovarOrientationRate	Roll-Pitch attitude rate error covariance with units of radians squared per second squared.
rrRr	CovarOrientationRate	Roll-Roll attitude rate error covariance with units of radians squared per second squared.
rrRy†	CovarOrientationRate	Roll-Yaw attitude rate error covariance with units of radians squared per second squared.
ryRy	CovarOrientationRate	Yaw-Yaw attitude rate error covariance with units of radians squared per second squared.

6.2.13 CovarOrientationType

Namespace: UMAA::Common::Measurement::CovarOrientationType

Description: Contains variances and covariances for random variables representing estimates of Observable values. Given an Observable with true value x the estimated value $\hat{x}$ is taken to be a random variable with sample space and distribution determined by the random processes which contribute to the calculation of the sampled Observable estimation. The ObservableError for the Observable is defined to be the Mean Squared Error, $\text{MSE} = E[(\hat{x} - x)^2]$. The MSE is equal to the sum of the variance of $\hat{x}$ and the square of the estimation bias: $\text{MSE} = \text{Var}[\hat{x}] + \text{Bias}[\hat{x}]^2 = E[(\hat{x} - E[\hat{x}])^2] - E[\hat{x} - x]^2$. The variance of $\hat{x}$ is equal to the MSE, and provides an estimate of the ObservableError, if and only if the underlying estimation process is unbiased.

Table 48: CovarOrientationType Structure Definition

Attribute Name	Attribute Type	Attribute Description
rpRp	CovarOrientation	Pitch-Pitch angle-angle error covariance with units of radians squared.
rpRy†	CovarOrientation	Pitch-Yaw angle-angle error covariance with units of radians squared.
rrRp†	CovarOrientation	Roll-Pitch angle-angle error covariance with units of radians squared.
rrRr	CovarOrientation	Roll-Roll angle-angle error covariance with units of radians squared.
rrRy†	CovarOrientation	Roll-Yaw angle-angle error covariance with units of radians squared.
ryRy	CovarOrientation	Yaw-Yaw angle-angle error covariance with units of radians squared.

6.2.14 CovariancePositionECEFType

Namespace: UMAA::Common::Measurement::CovariancePositionECEFType

Description: Contains variances and covariances for random variables representing estimates of Observable values. Given an Observable with true value x the estimated value $\hat{x}$ is taken to be a random variable with sample space and distribution determined by the random processes which contribute to the calculation of the sampled Observable estimation. The ObservableError for the Observable is defined to be the Mean Squared Error, $\text{MSE} = E[(\hat{x} - x)^2]$. The MSE is equal to the sum of the variance of $\hat{x}$ and the square of the estimation bias: $\text{MSE} = \text{Var}[\hat{x}] + \text{Bias}[\hat{x}]^2 = E[(\hat{x} - E[\hat{x}])^2] - E[\hat{x} - x]^2$. The variance of $\hat{x}$ is equal to the MSE, and provides an estimate of the ObservableError, if and only if the underlying estimation process is unbiased.

Table 49: CovariancePositionECEFType Structure Definition

Attribute Name	Attribute Type	Attribute Description
pxPx	CovarPosPosECEF	x-x position error covariance with units of meters squared.
pxPy†	CovarPosPosECEF	x-y position error covariance with units of meters squared.
pxPz†	CovarPosPosECEF	x-z position error covariance with units of meters squared.
pyPy	CovarPosPosECEF	y-y position error covariance with units of meters squared.
pyPz†	CovarPosPosECEF	y-z position error covariance with units of meters squared.
pzPz	CovarPosPosECEF	z-z position error covariance with units of meters squared.

6.2.15 CovariancePositionNEDType

Namespace: UMAA::Common::Measurement::CovariancePositionNEDType

Description: Contains variances and covariances for random variables representing estimates of Observable values. Given an Observable with true value x the estimated value $\hat{x}$ is taken to be a random variable with sample space and distribution determined by the random processes which contribute to the calculation of the sampled Observable estimation. The ObservableError for the Observable is defined to be the Mean Squared Error, $\text{MSE} = E[(\hat{x} - x)^2]$. The MSE is equal to the sum of the variance of $\hat{x}$ and the square of the estimation bias: $\text{MSE} = \text{Var}[\hat{x}] + \text{Bias}[\hat{x}]^2 = E[(\hat{x} - E[\hat{x}])^2] - E[\hat{x} - x]^2$. The variance of $\hat{x}$ is equal to the MSE, and provides an estimate of the ObservableError, if and only if the underlying estimation process is unbiased.

Table 50: CovariancePositionNEDType Structure Definition

Attribute Name	Attribute Type	Attribute Description
pdPd†	CovarPosPosNED	Down-Down position error covariance with units of meters squared.
pePd†	CovarPosPosNED	East-Down position error covariance with units of meters squared.
pePe	CovarPosPosNED	East-East position error covariance with units of meters squared.
pnPd†	CovarPosPosNED	North-Down position error covariance with units of meters squared.
pnPe†	CovarPosPosNED	North-East position error covariance with units of meters squared.
pnPn	CovarPosPosNED	North-North position error covariance with units of meters squared.

6.2.16 CovarianceVelocityType

Namespace: UMAA::Common::Measurement::CovarianceVelocityType

Description: Contains variances and covariances for random variables representing estimates of Observable values. Given an Observable with true value x the estimated value $\hat{x}$ is taken to be a random variable with sample space and distribution determined by the random processes which contribute to the calculation of the sampled Observable estimation. The ObservableError for the Observable is defined to be the Mean Squared Error, $\text{MSE} = E[(\hat{x} - x)^2]$. The MSE is equal to the sum of the variance of $\hat{x}$ and the square of the estimation bias: $\text{MSE} = \text{Var}[\hat{x}] + \text{Bias}[\hat{x}]^2 = E[(\hat{x} - E[\hat{x}])^2] - E[\hat{x} - x]^2$. The variance of $\hat{x}$ is equal to the MSE, and provides an estimate of the ObservableError, if and only if the underlying estimation process is unbiased.

Table 51: CovarianceVelocityType Structure Definition

Attribute Name	Attribute Type	Attribute Description
vdVd	CovarVelVel	Down-Down velocity-velocity error covariance with units of meters squared per second squared.
veVd†	CovarVelVel	East-Down velocity-velocity error covariance with units of meters squared per second squared.
veVe	CovarVelVel	East-East velocity-velocity error covariance with units of meters squared per second squared.
vnVd†	CovarVelVel	North-Down velocity-velocity error covariance with units of meters squared per second squared.
vnVe†	CovarVelVel	North-East velocity-velocity error covariance with units of meters squared per second squared.
vnVn	CovarVelVel	North-North velocity-velocity error covariance with units of meters squared per second squared.

6.2.17 DepthType

Namespace: UMAA::Common::Measurement::DepthType

Description: Defines the depth below sea level.

Table 52: DepthType Structure Definition

Attribute Name	Attribute Type	Attribute Description
depth	DistanceBSL	The depth below sea level.

6.2.18 ElevationType

Namespace: UMAA::Common::Measurement::ElevationType

Description: Union Type. Elevation in either altitude from sea floor or depth from surface (other altitude options support above ground and sea level for potential hybrid vehicles).

Table 53: ElevationType Union(s)

Type Name	Type Description
AltitudeAGLType	The height above ground level.
AltitudeASFType	The height above sea floor.
AltitudeGeodeticType	The geodetic height above the ellipsoid.
AltitudeMSLType	The orthometric height above the Geoid (Mean Sea Level).
DepthType	Defines the depth below sea level.

6.2.19 GeoPosition2D

Namespace: UMAA::Common::Measurement::GeoPosition2D

Description: Specifies a location on the surface of the Earth.

Table 54: GeoPosition2D Structure Definition

Attribute Name	Attribute Type	Attribute Description
geodeticLatitude	GeodeticLatitude	Specifies the north-south coordinate of the position.
geodeticLongitude	GeodeticLongitude	Specifies the east-west coordinate of the position.

6.2.20 MagneticDeviationType

Namespace: UMAA::Common::Orientation::MagneticDeviationType

Description: This structure is used to describe the magnetic deviation at a specific heading for the vehicle.

Table 55: MagneticDeviationType Structure Definition

Attribute Name	Attribute Type	Attribute Description
heading	HeadingTrueNorthAngle	The vehicle heading with respect to true north.

Attribute Name	Attribute Type	Attribute Description
magneticDeviation	MagneticVariation	The difference in angle between measured magnetic north and actual magnetic north on the horizontal plane as a result of the vehicle's material properties.

6.2.21 Orientation3DNEDType

Namespace: UMAA::Common::Orientation::Orientation3DNEDType

Description: A requirement that specifies an orientation relative to a North-East-Down coordinate system centered on the platform.

Table 56: Orientation3DNEDType Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitch	PitchYNEDType	Defines a pitch relative to a North-East-Down coordinate system centered on the platform.
roll	RollXNEDType	Defines a roll relative to a North-East-Down coordinate system centered on the platform.
yaw	YawZNEDType	Defines a yaw relative to a North-East-Down coordinate system centered on the platform.

6.2.22 OrientationVel3D

Namespace: UMAA::Common::Measurement::OrientationVel3D

Description: Specifies the rate of change for each axis of an Orientation.

Table 57: OrientationVel3D Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitchRate	PitchRate	Specifies the rate of change of the platform's pitch angle relative to a NED frame centered at the platform location.
rollRate	RollRate	Specifies the rate of change of the platform's roll angle relative to a NED frame centered at the platform location.
yawRate	YawRate	Specifies the rate of change of the platform's yaw angle relative to a NED frame centered at the platform location.

6.2.23 PitchYNEDType

Namespace: UMAA::Common::Orientation::PitchYNEDType

Description: A requirement that specifies a pitch relative to the NED coordinate system.

Table 58: PitchYNEDType Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitch	PitchHalfAngle	Defines a pitch relative to the NED coordinate system.

6.2.24 Polygon

Namespace: UMAA::Common::Measurement::Polygon

Description: Specifies an area defined by a polygon.

Table 59: Polygon Structure Definition

Attribute Name	Attribute Type	Attribute Description
lineKind	LineSegmentEnumType	Indicates the type of lines that form the polygon.
referencePoint	sequence<GeoPosition2D> max size = 128	Specifies the geospatial points defining the vertices of a polygon. Three or more points are needed to define a polygon.

6.2.25 Quaternion

Namespace: BasicTypes::Quaternion

Description: Defines a four-element vector that can be used to encode any rotation in a 3D coordinate system.

Table 60: Quaternion Structure Definition

Attribute Name	Attribute Type	Attribute Description
a		Real number a.
b		Real number b.
c		Real number c.
d		Real number d.

6.2.26 RollXNEDType

Namespace: UMAA::Common::Orientation::RollXNEDType

Description: A requirement that specifies a roll relative to the NED coordinate system.

Table 61: RollXNEDType Structure Definition

Attribute Name	Attribute Type	Attribute Description
roll	RollAngle	Defines a roll relative to the NED coordinate system.

6.2.27 Velocity3DPlatformNEDType

Namespace: UMAA::Common::Measurement::Velocity3DPlatformNEDType

Description: Specifies the velocity of the platform with vector components expressed in a North-East-Down coordinate system centered on the platform.

Table 62: Velocity3DPlatformNEDType Structure Definition

Attribute Name	Attribute Type	Attribute Description
downSpeed	DownSpeed	Specifies the down velocity vector in a North-East-Down coordinate system centered on the platform.
eastSpeed	EastSpeed	Specifies the easting velocity vector in a North-East-Down coordinate system centered on the platform.
northSpeed	NorthSpeed	Specifies the northing velocity vector in a North-East-Down coordinate system centered on the platform.

6.2.28 WaterspaceVolumeType

Namespace: UMAA::MM::BaseType::WaterspaceVolumeType

Description: This structure is used to describe the volume of a region enclosed by a right simple-polygonal prism with bases perpendicular to gravity.

Table 63: WaterspaceVolumeType Structure Definition

Attribute Name	Attribute Type	Attribute Description
area	Polygon	Describes the area enclosed by the simple polygon.
ceiling	ElevationType	Describes the plane relative to the mean sea level that intersects the highest point or plane of the polygon.
floor	ElevationType	Describes the plane relative to the mean sea level that intersects the lowest point or plane of the polygon.

6.2.29 YawZNEDType

Namespace: UMAA::Common::Orientation::YawZNEDType

Description: Specifies a yaw relative to the NED coordinate system.

Table 64: YawZNEDType Structure Definition

Attribute Name	Attribute Type	Attribute Description
yaw	YawPosAngle	Defines a yaw relative to the NED coordinate system.

6.2.30 ZoneType

Namespace: UMAA::MM::BaseType::ZoneType

Description: This structure is used to describe the operational parameters of a zone.

Table 65: ZoneType Structure Definition

Attribute Name	Attribute Type	Attribute Description
zone	WaterspaceVolumeType	Defines the zone.
zoneKind	ZoneKindEnumType	Defines the type of zone, i.e., keep in, keep out, etc.
zoneID*	NumericGUID	Defines an identifier associated with each defined zone. This zoneID does not have to match the source zoneID from the WaterspacePlan.

6.3 Enumerations

Enumerations are used extensively throughout UMAA. This section lists the values associated with each enumeration defined in UCS-UMAA.

6.3.1 COLREGSClassificationEnumType

Namespace: UMAA::Common::MaritimeEnumeration::COLREGSClassificationEnumType

Description: This enumeration is designed to classify a vehicle with accordance to COLREGS definitions.

Table 66: COLREGSClassificationEnumType Enumeration

Enumeration Value	Description
ANCHORED	The vehicle is currently anchored.
CONSTRAINED_BY_DRAUGHT	The vehicle is limited in its navigational capabilities by its draught.
FISHING	The term vehicle engaged in fishing means any vehicle fishing with nets, lines, trawls, or other fishing apparatus which restricts maneuverability, but does not include a vehicle fishing with trolling lines or other fishing apparatus which do not restrict maneuverability.
NON_VESSEL	This value refers to contacts that are determined to be not vehicles (buoy, land, etc) where COLREGS does not apply.
NOT_UNDER_COMMAND	The term vehicle not under command means a vehicle which, through some exceptional circumstance, is unable to maneuver as required by these Rules and is therefore unable to keep out of the way of another vehicle.
POWER_DRIVEN_UNDERWAY	The term power-driven vehicle means any vehicle propelled by machinery.
PUSHING	The vehicle is engaged in a pushing operation such as severely restricts the pushing vehicle and her push in their ability to deviate from their course.
RESTRICTED_IN_ABILITY_TO_-MANUEVER	The term vehicle restricted in her ability to maneuver means a vehicle which, from the nature of her work, is restricted in her ability to maneuver as required by these Rules and is therefore unable to keep out of the way of another vehicle.
SAILING	The term sailing vehicle means any vehicle under sail provided that propelling machinery, if fitted, is not being used.
TOWING	The vehicle is engaged in a towing operation such as severely restricts the towing vehicle and her tow in their ability to deviate from their course.

6.3.2 CommandStatusReasonEnumType

Namespace: UMAA::Common::MaritimeEnumeration::CommandStatusReasonEnumType

Description: Defines a mutually exclusive set of reasons why a command status state transition has occurred.

Table 67: CommandStatusReasonEnumType Enumeration

Enumeration Value	Description
CANCELED	Indicates a transition to the CANCELED state when the command is canceled successfully.
INTERRUPTED	Indicates a transition to the FAILED state when the command has been interrupted by a higher priority process.
OBJECTIVE_FAILED	Indicates a transition to the FAILED state when the commanded resource is unable to achieve the command's objective due to external factors.

Enumeration Value	Description
RESOURCE_FAILED	Indicates a transition to the FAILED state when the commanded resource is unable to achieve the command's objective due to resource or platform failure.
RESOURCE_REJECTED	Indicates a transition to the FAILED state when the commanded resource rejects the command for some reason.
SERVICE_FAILED	Indicates a transition to the FAILED state when the commanded resource is unable to achieve the command's objective due to processing failure.
SUCCEEDED	Indicates the conditions to proceed to this state have been met and a normal state transition has occurred.
TIMEOUT	Indicates a transition to the FAILED state when the command is not acknowledged within some defined time bound.
UPDATED	Indicates a transition back to the ISSUED state from a non-terminal state when the command has been updated.
VALIDATION_FAILED	Indicates a transition to the FAILED state when the command contains missing, out-of-bounds, or otherwise invalid parameters.

6.3.3 LineSegmentEnumType

Namespace: UMAA::Common::Enumeration::LineSegmentEnumType

Description: A mutually exclusive set of values that defines the line segment types used for navigation.

Table 68: LineSegmentEnumType Enumeration

Enumeration Value	Description
GREAT_CIRCLE	The line segment should be traversed as one following a great circle. A great circle is the shortest distance between two points on the surface of a sphere, measured along the surface of the sphere.
RHUMB	The line segment should be traversed as one following a rhumb line. A rhumb line represents an arc cross all meridians of longitude at the same angle (i.e. a path with constant bearing).

6.3.4 CommandStatusEnumType

Namespace: UMAA::Common::MaritimeEnumeration::CommandStatusEnumType

Description: Defines a mutually exclusive set of values that defines the states of a command as it progresses towards completion.

Table 69: CommandStatusEnumType Enumeration

Enumeration Value	Description
CANCELED	The command was canceled by the requestor before the command completed successfully.
COMMANDED	The command has been placed in the resource's command queue but has not yet been accepted.
COMPLETED	The command has been completed successfully.

Enumeration Value	Description
EXECUTING	The command is being performed by the resource and has not yet been completed.
FAILED	The command has been attempted, but was not successful.
ISSUED	The command has been issued to the resource (typically a sensor or streaming device), but processing has not yet commenced.

6.3.5 NavigationSolutionEnumType

Namespace: UMAA::Common::MaritimeEnumeration::NavigationSolutionEnumType

Description: A mutually exclusive set of values that defines the navigation solution.

Table 70: NavigationSolutionEnumType Enumeration

Enumeration Value	Description
ESTIMATED	Estimated
GROUND_TRUTH	Ground Truth
MEASURED	Measured

6.3.6 SeaStateEnumType

Namespace: UMAA::Common::OrderedEnumeration::SeaStateEnumType

Description: A mutually exclusive set of values that defines sea state conditions according to the World Meteorological Organization (WMO).

Table 71: SeaStateEnumType Enumeration

Enumeration Value	Description
CALM_GLOSSY	The sea state is calm glassy, which is characterized by wave height of 0 meters and a World Meteorological Organization (WMO) sea state code of 0.
CALM_RIPPLED	The sea state is calm rippled, which is characterized by wave heights of 0 to 0.1 meters and a World Meteorological Organization (WMO) sea state code of 1.
HIGH	The sea state is high, which is characterized by wave heights of 6 to 9 meters and a World Meteorological Organization (WMO) sea state code of 7.
MODERATE	The sea state is moderate, which is characterized by wave heights of 1.25 to 2.5 meters and a World Meteorological Organization (WMO) sea state code of 4.
PHENOMENAL	The sea state is phenomenal, which is characterized by wave heights of over 14 meters and a World Meteorological Organization (WMO) sea state code of 9.
ROUGH	The sea state is rough, which is characterized by wave heights of 2.5 to 4 meters and a World Meteorological Organization (WMO) sea state code of 5.
SLIGHT	The sea state is slight, which is characterized by wave heights of 0.5 to 1.25 meters and a World Meteorological Organization (WMO) sea state code of 3.
SMOOTH	The sea state is smooth, which is characterized by wave heights of 0.1 to 0.5 meters and a World Meteorological Organization (WMO) sea state code of 2.

Enumeration Value	Description
VERY_HIGH	The sea state is very high, which is characterized by wave heights of 9 to 14 meters and a World Meteorological Organization (WMO) sea state code of 8.
VERY_ROUGH	The sea state is very rough, which is characterized by wave heights of 4 to 6 meters and a World Meteorological Organization (WMO) sea state code of 6.

6.3.7 SourceIndicatorEnumType

Namespace: UMAA::Common::MaritimeEnumeration::SourceIndicatorEnumType

Description: Defines a mutually exclusive set of values for the data source.

Table 72: SourceIndicatorEnumType Enumeration

Enumeration Value	Description
ACTUAL	Source of data is actual sensor data.
GROUND_TRUTH	Source of data is ground truth.
SIMULATED	Source of data is simulated sensor data.
TENTATIVE	Source of data is tentative sensor data.

6.3.8 SpecialManeuverIndicatorEnumType

Namespace: UMAA::Common::MaritimeEnumeration::SpecialManeuverIndicatorEnumType

Description: A mutually exclusive set of values that defines the special maneuver indicator.

Table 73: SpecialManeuverIndicatorEnumType Enumeration

Enumeration Value	Description
ENGAGED	The vessel is engaged in a special maneuver.
NOT_AVAILABLE	The contact source is explicitly reporting that the special maneuver indicator is not available.
NOT_ENGAGED	The vessel is not engaged in a special maneuver.
NOT_PROVIDED	The contact source is not able to determine this information.

6.3.9 VehicleSpeedModeEnumType

Namespace: UMAA::Common::MaritimeEnumeration::VehicleSpeedModeEnumType

Description: A mutually exclusive set of values that defines the type of performance speed of the vehicle.

Table 74: VehicleSpeedModeEnumType Enumeration

Enumeration Value	Description
LRC	Long Range Cruise. A speed that optimizes time, distance and fuel consumption for a vehicle (definition of "optimized" is subjective. Example: for a planing hull, this is usually the minimum planing speed, even though lower speeds can achieve longer endurance or range.)
MEC	Maximum Endurance Cruise. The speed that maximizes the time a vehicle can travel.
MRC	Maximum Range Cruise. The speed that maximizes the distance a vehicle can travel.
SLOW	Slow speed. Minimum speed at which the vehicle can operate (definition of "operate" is subjective. Example: minimum speed to achieve maneuverability, engine idle speed/gear clutched in "idle ahead", etc.)
VEHICLE_SPECIFIC	Preset speed for the vehicle, that is in the range of speeds for the subject vehicle

6.3.10 VisualClassificationEnumType

Namespace: UMAA::Common::MaritimeEnumeration::VisualClassificationEnumType

Description: Defines a mutually exclusive set of values that defines the visual classification.

Table 75: VisualClassificationEnumType Enumeration

Enumeration Value	Description
AID_TO_NAVIGATION_CHANNEL_MARKER	A channel marker used as an aid to navigation.
AID_TO_NAVIGATION_GENERAL	A general aid to navigation.
AID_TO_NAVIGATION_LARGE_BUOY	A large buoy used as an aid to navigation.
AID_TO_NAVIGATION_LIGHTHOUSE	A lighthouse used as an aid to navigation.
AID_TO_NAVIGATION_SMALL_BUOY	A small buoy used as an aid to navigation.
LARGE_GENERAL_OBSTACLE	A large general obstacle.
LARGE_VESSEL_CARGO	A large cargo vehicle.
LARGE_VESSEL_GENERAL	A large general vehicle.
LARGE_VESSEL_MILITARY	A large military vehicle.
LARGE_VESSEL_OTHER	A large vehicle that does not fit into other LARGE_VESSEL categories.
LARGE_VESSEL_PASSENGER	A large passenger vehicle.
MEDIUM_VESSEL_FISHING	A fishing vehicle.
MEDIUM_VESSEL_GENERAL	A medium general vehicle.
MEDIUM_VESSEL_MILITARY	A medium military vehicle.
MEDIUM_VESSEL_OTHER	A medium vehicle that does not fit into other MEDIUM_VESSEL categories.
MEDIUM_VESSEL_TUG	A tug vehicle.
MEDIUM_VESSEL_TUG_IN_TOW	A tug vehicle towing another vehicle.
MEDIUM_VESSEL_YACHT	A yacht.

Enumeration Value	Description
SAILBOAT	A sailboat.
SMALL_GENERAL_OBSTACLE	A small general obstacle.
SMALL_VESSEL_GENERAL	A small general vehicle.
SMALL_VESSEL_JET_SKI	A jet ski.
SMALL_VESSEL_MILITARY	A small military vehicle.
SMALL_VESSEL_OTHER	A small vehicle that does not fit into other SMALL_VESSEL categories
SMALL_VESSEL_POWER_BOAT	A power boat.

6.3.11 ZoneKindEnumType

Namespace: UMAA::Common::MaritimeEnumeration::ZoneKindEnumType

Description: Defines a mutually exclusive set of zone kinds.

Table 76: ZoneKindEnumType Enumeration

Enumeration Value	Description
AREA_OF_INTEREST	Defines a zone that should be covered by the vehicle's sensors and contains something interesting (e.g. a contact).
KEEP_IN	Defines a zone that the vehicle is required to keep in.
KEEP_OUT	Defines a zone that the vehicle is required to keep out.

6.4 Type Definitions

This section describes the type definitions for UMAA. The table below lists how UMAA defined types are mapped to the DDS primitive types.

Table 77: Type Definitions

Type Name	Primitive Type	Range of Values	Description
CourseTrueNorth	double	fractionDigits=3 maxInclusive=3.142 minInclusive=-3.142 units=Radian referenceFrame=TrueNorth	Specifies the direction of the platform's motion relative to true north. The measurement is stated in radians between 0 and 2 pi.
CovarOrientation	double	referenceFrame=Counting units=N/A fractionDigits=3	Specifies a radians-radians measure of linear dependence that indicates the 1-sigma error covariance of the orientation angle.
CovarOrientationRate	double	units=RadiansSquaredPerSecondSquared	Represents the error covariance in the rate of change of angles.
CovarPosPosECEF	double	referenceFrame=Counting	Describes a meters-meters measure of linear dependence that indicates position-position error covariance in the earth-centered, earth-fixed reference frame.
CovarPosPosNED	double	referenceFrame=Counting	Describes a meters-meters measure of linear dependence that indicates position-position error covariance in the NED coordinate system
CovarVelVel	double	referenceFrame=Counting units=N/A fractionDigits=3	Specifies a meters squared-seconds squared measure of linear dependence that indicates velocity-velocity error covariance in the NED coordinate system
DateTimeNanoseconds	long	units=Nanoseconds minInclusive=0 maxInclusive=999999999 fractionDigits=0	number of nanoseconds elapsed within the current second.
DateTimeSeconds	longlong	units=Seconds minInclusive=-9223372036854775807 maxInclusive=9223372036854775807 fractionDigits=0	seconds offset from the standard POSIX (IEEE Std 1003.1-2017) epoch reference point of January 1st, 1970 00:00:00 UTC.
Distance	double	fractionDigits=3 maxInclusive=401056000 minInclusive=0 units=Meter referenceFrame=Counting	This type stores a distance in meters.
DistanceAGL	double	fractionDigits=3 minInclusive=0.0 units=Meter referenceFrame=AGL	Describes the height above ground level of the vehicle.

Type Name	Primitive Type	Range of Values	Description
DistanceASF	double	fractionDigits=3 maxInclusive=401056000 minInclusive=0 units=Meter referenceFrame=ASF	The altitude or distance above the sea floor in meters.
DistanceBSL	double	fractionDigits=3 maxInclusive=10000 minInclusive=0 units=Meter referenceFrame=BSL	The distance below sea level in meters.
Down	double	axisAbbrev=Z axisDirection=down axisUnit=Meter maximumValue=50000 minimumValue=-50000 rangeMeaning=exact resolution=0.001	Used for measuring position and increases in magnitude as values extend toward the center of the Earth. Down measurements are expressed in meters.
DownSpeed	double	axisDirection=down axisUnit=MeterPerSecond maximumValue=299,792,458 minimumValue=-299,792,458 rangeMeaning=exact resolution=0.001 units=MeterPerSecond fractionDigits=3	Used for measuring speed and increases in magnitude as speed toward the center of the Earth increases.
EastSpeed	double	axisDirection=east axisUnit=MeterPerSecond maximumValue=299,792,458 minimumValue=-299,792,458 rangeMeaning=exact resolution=0.001 units=MeterPerSecond fractionDigits=3	Used for measuring speed and increases in magnitude as speed in the easterly direction increases.
EngineSpeed	double	referenceFrame=Counting units=RevolutionsPerMinute minInclusive=-100000 maxInclusive=100000 fractionDigits=0	This type stores number of occurrences in revolutions per minute (RPM). Negative number is used for reverse RPM.
Forward	double	axisAbbrev=X axisDirection=fore axisUnit=Meter maximumValue=20000000 minimumValue=-20000000 rangeMeaning=exact resolution=0.001	Used for measuring position and increases in magnitude as position extends out the "front" of the reference body. Forward measurements are expressed in meters.
GeodeticAltitude	double	fractionDigits=6 maxInclusive=700000 minInclusive=-10000 units=Meter axisAbbrev=Altitude axisDirection=up axisUnit=Meter rangeMeaning=exact resolution=0.0000000001	Used for measuring position and increases in magnitude as position extends upward. Altitude measurements are expressed in meters.

Type Name	Primitive Type	Range of Values	Description
GeodeticLatitude	double	axisAbbrev=Latitude axisDirection=north/south axisUnit=Degrees maximumValue=90.0 minimumValue=-90.0 rangeMeaning=exact resolution=0.0000000001	Used for measuring position and increases in magnitude as position extends from the south pole to the north pole. Latitude measurements are expressed in degrees.
GeodeticLongitude	double	axisAbbrev=Longitude axisDirection=east axisUnit=Degrees maximumValue=180.0 minimumValue=-180.0 rangeMeaning=wraparound resolution=0.0000000001	Used for measuring position and increases in magnitude as position extends eastward. Longitude measurements are expressed in degrees. Longitude measurements are periodic and whose limits (min and max), while mathematically discontinuous, represent a continuous range.
GroundSpeed	double	fractionDigits=3 maxInclusive=299,792,458 minInclusive=-299,792,458 units=MeterPerSecond referenceFrame=TrueNorth	The magnitude of the horizontal velocity vector of an aircraft relative to the ground.
HeadingTrueNorthAngle	double	fractionDigits=3 maxInclusive=3.142 minInclusive=-3.142 units=Radian referenceFrame=TrueNorth	Describes heading as a value between -pi and pi with respect to True North.
IndicatedAirspeed	double	fractionDigits=6 maxInclusive=299,792,458 minInclusive=0 units=MeterPerSecond referenceFrame=LocalAirMass	This type specifies the magnitude of an aircraft's velocity (the rate of change of its position). Indicated airspeed (IAS) is the airspeed read directly from the airspeed indicator on an aircraft, driven by the pitot-static system.
LargeCollectionSize	long	fractionDigits=0 maxInclusive=2147483647 minInclusive=0 units=N/A	Specifies the size of a Large Collection.
Left	double	fractionDigits=3 maxInclusive=20000000 minInclusive=-20000000 units=Meter axisAbbrev=Y axisDirection=port axisUnit=Meter rangeMeaning=exact resolution=0.001	Used for measuring position and increases in magnitude as position extends out the "left" of the reference body. Left measurements are expressed in meters.
MagneticVariation	double	fractionDigits=3 maxInclusive=3.142 minInclusive=-3.142 units=Radian referenceFrame=TrueNorth	Specifies the angle on the horizontal plane between true north and magnetic north (the direction the north end of the compass needle points). The measurement is stated in radians between -pi and pi.
MMSI	string	length=9 units=N/A	Maritime Mobile Service Identity, a series of nine digits which uniquely identifies a station.

Type Name	Primitive Type	Range of Values	Description
MSLAltitude	double	fractionDigits=3 minInclusive=0.0 units=Meter referenceFrame=Altitude	Describes the current orthometric height above the Geoid (Mean Sea Level).
NorthSpeed	double	axisDirection=north axisUnit=MeterPerSecond maximumValue=299,792,458 minimumValue=-299,792,458 rangeMeaning=exact resolution=0.001 units=MeterPerSecond fractionDigits=3	Used for measuring speed and increases in magnitude as speed in the northerly direction increases.
NumericGUID	octet[16]	units=N/A minInclusive=0 maxInclusive=(2^128)-1 fractionDigits=0	Represents a 128-bit number according to RFC 4122 variant 2.
Percent	double	fractionDigits=3 maxInclusive=1000 minInclusive=0 units=Percent referenceFrame=Counting	Defines a percentage where 100% = 100.0. Values greater than 100% are allowed.
PitchHalfAngle	double	fractionDigits=3 maxInclusive=1.571 minInclusive=-1.571 units=Radian referenceFrame=PlatformNED	Specifies the platform's rotation about the lateral axis (e.g. the axis parallel to the wings) in a locally level, North-East-Down coordinate system centered on the platform. Pitch is zero when the platform is "nose to tail level" in the North-East plane. The measurement is stated in radians between -0.5 pi and 0.5 pi.
PitchRate	double	fractionDigits=3 maxInclusive=32.767 minInclusive=0 units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the platform's pitch angle relative to a NED frame centered at the platform location.
RollAngle	double	fractionDigits=3 maxInclusive=3.142 minInclusive=-3.142 units=Radian referenceFrame=PlatformNED	Specifies a platform's rotation about the longitudinal axis (e.g. the axis through the body of the vehicle from tail to nose) in a locally level, North-East-Down coordinate system centered on the vehicle. Roll is zero when the platform is "wing-tip to wing-tip" level in the North-East plane. The measurement is stated in radians between -pi and pi.
RollRate	double	fractionDigits=3 maxInclusive=N/A minInclusive=N/A units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the platform's roll angle relative to a NED frame centered at the platform location.

Type Name	Primitive Type	Range of Values	Description
Speed	double	fractionDigits=6 maxInclusive=299,792,458 minInclusive=0 units=MeterPerSecond referenceFrame=Counting	This type stores speed in meters/s.
SpeedLocalWaterMass	double	fractionDigits=6 maxInclusive=299,792,458 minInclusive=0 units=MeterPerSecond referenceFrame=LocalWaterMass	This type stores speed in meters/s.
StringLongDescription	string	length=4095 units=N/A minInclusive=N/A maxInclusive=N/A	Represents a long format description.
StringShortDescription	string	length=1023 units=N/A minInclusive=N/A maxInclusive=N/A	Represents a short format description.
YawPosAngle	double	fractionDigits=3 maxInclusive=6.2831853071795864769252867665590 minInclusive=0 units=Radian referenceFrame=PlatformNED	The yaw angle relative to the NED coordinate system centered at the platform location.
YawRate	double	fractionDigits=3 maxInclusive=N/A minInclusive=N/A units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the platform's yaw angle relative to a NED frame centered at the platform location.

A Appendices

A.1 Glossary

Note: This glossary aims to define terms that are uncommon, or have a special meaning in the context of UMAA and/or the DoD. This glossary covers the complete UMAA specification. Not every word defined here appears in every ICD.

Almanac Data (GPS)	A navigation message that contains information about the time and status of the entire satellite constellation.
Coulomb	The SI unit of electric charge, equal to the quantity of electricity conveyed in one second by a current of one ampere.
Ephemeris Data (GPS)	A navigation message used to calculate the position of each satellite in orbit.
Glowplug or Glow Plug	A heating device used to aid in starting diesel engines.
Interoperability	1) The ability to act together coherently, effectively, and efficiently to achieve tactical, operational, and strategic objectives. 2) The condition achieved among communications-electronics systems or items of communications-electronics equipment when information or services can be exchanged directly and satisfactorily between them and/or their users.
Mean Sea Level	The average height of the surface of the sea for all stages of the tide; used as a reference for elevations.
Middleware	A type of computer software that provides services to software applications beyond those available from the operating system. Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.
SoaML	The Service oriented architecture Modeling Language (SoaML) specification that provides a metamodel and a UML profile for the specification and design of services within a service-oriented architecture. The specification is managed by the Object Management Group (OMG).

A.2 Acronyms

Note: This acronym list is included in every ICD and covers the complete UMAA specification. Not every acronym appears in every ICD.

ADD	Architecture Design Description
AGL	Above Sea Level
ASF	Above Sea Floor
BSL	Below Sea Level
BWL	Beam at Waterline
C2	Command and Control
CMD	Command
CO	Comms Operations
CPA	Closest Point of Approach
CTD	Conductivity, Temperature and Depth
DDS	Data Distribution Service
DTED	Digital Terrain Elevation Data
EGM	Earth Gravity Model
EO	Engineering Operations
FB	Feedback
GUID	Globally Unique Identifier
HM&E	Hull, Mechanical, & Electrical
ICD	Interface Control Document

ID	Identifier
IDL	Interface Definition Language Specification
IMO	International Maritime Organization
INU	Inertial Navigation Unit
LDM	Logical Data Model
LOA	Length Over All
LRC	Long Range Cruise
LWL	Length at Waterline
MDE	Maritime Domain Extensions
MEC	Maximum Endurance Cruise
MM	Mission Management
MMSI	Maritime Mobile Service Identity
MO	Maneuver Operations
MRC	Maximum Range Cruise
MSL	Mean Sea Level
OMG	Object Management Group
PIM	Platform Independent Model
PMC	Primary Mission Control
PNT	Precision Navigation and Timing
PO	Processing Operations
PSM	Platform Specific Model
RMS	Root-Mean-Square
RPM	Revolutions per minute
RTPS	Real Time Publish Subscribe
RTSP	Real Time Streaming Protocol
SA	Situational Awareness
SEM	Sensor and Effector Management
SO	Support Operations
SoaML	Service-oriented architecture Modeling Language
STP	Standard Temperature and Pressure
UCS	Unmanned Systems Control Segment
UMAA	Unmanned Maritime Autonomy Architecture
UML	Unified Modeling Language
UMS	Unmanned Maritime System
UMV	Unmanned Maritime Vehicle
UxS	Unmanned System
WGS84	Global Coordinate System
WMM	World Magnetic Model
WMO	World Meteorological Organization