Unmanned Maritime Autonomy Architecture (UMAA)
Maneuver Operations (MO)
Interface Control Document (ICD)
(UMAA-SPEC-MOICD)

Version 6.0

6 June 2024

Contents

List of Figures

List of Tables

1 Scope

1.1 Identification

This document defines a set of services as part of the Unmanned Maritime Autonomy Architecture (UMAA). The services and their corresponding interfaces covered in this ICD encompass the functionality to control and maneuver an Unmanned Maritime Vehicle (UMV) (surface or undersea). As such, it includes the commands and status to/from an unmanned vehicle's control systems for controlling all aspects of maneuvering and its associated dynamics. This includes both low-level controls such as heading and speed, as well as higher-level behaviors for traversing waypoints. This ICD also includes instructions for managing driving constraints such as setting bounds on desired speed range or setting a desired maximum turn rate. This document is generated automatically from data models that define its services and their 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.

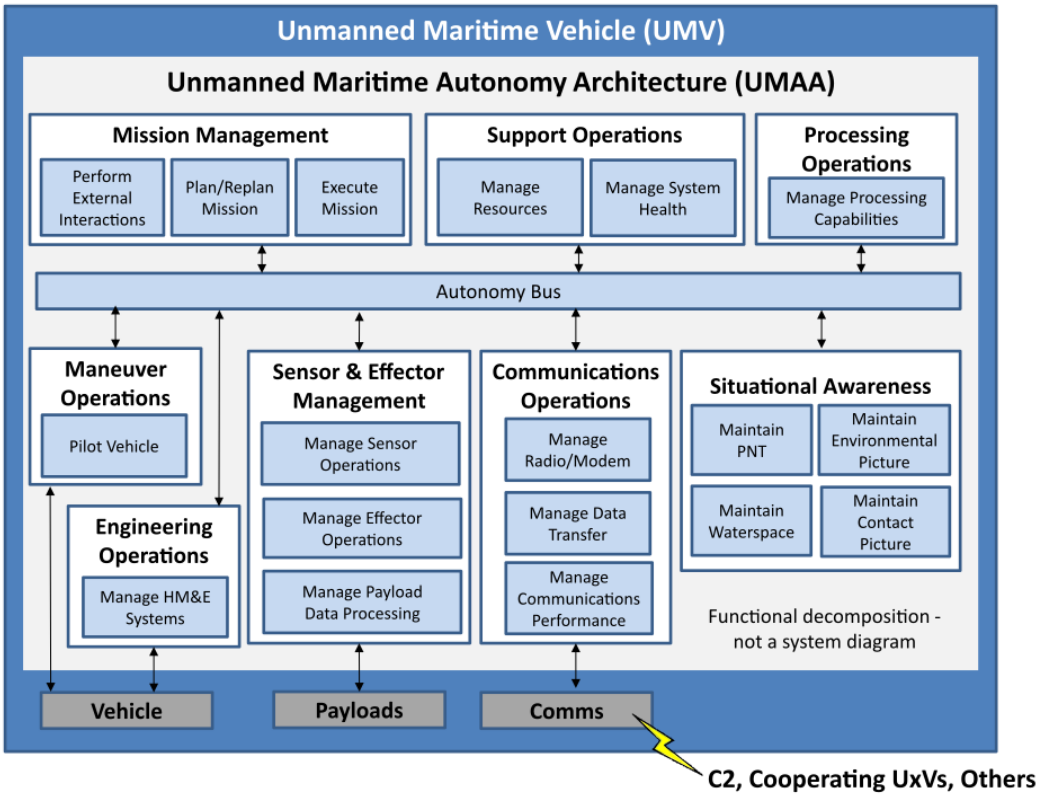

Figure 1: UMAA Functional Organization.

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.

Maneuver Operations defines the services required to drive an unmanned vehicle. Figure 2 depicts an example of various levels of maneuvering behaviors in relation to navigation sensing and Hull, Mechanical, & Electrical (HM&E) control services provided in separate ICDs. Figure 2 depicts an example of possible component service groupings (designated by dashed lines).

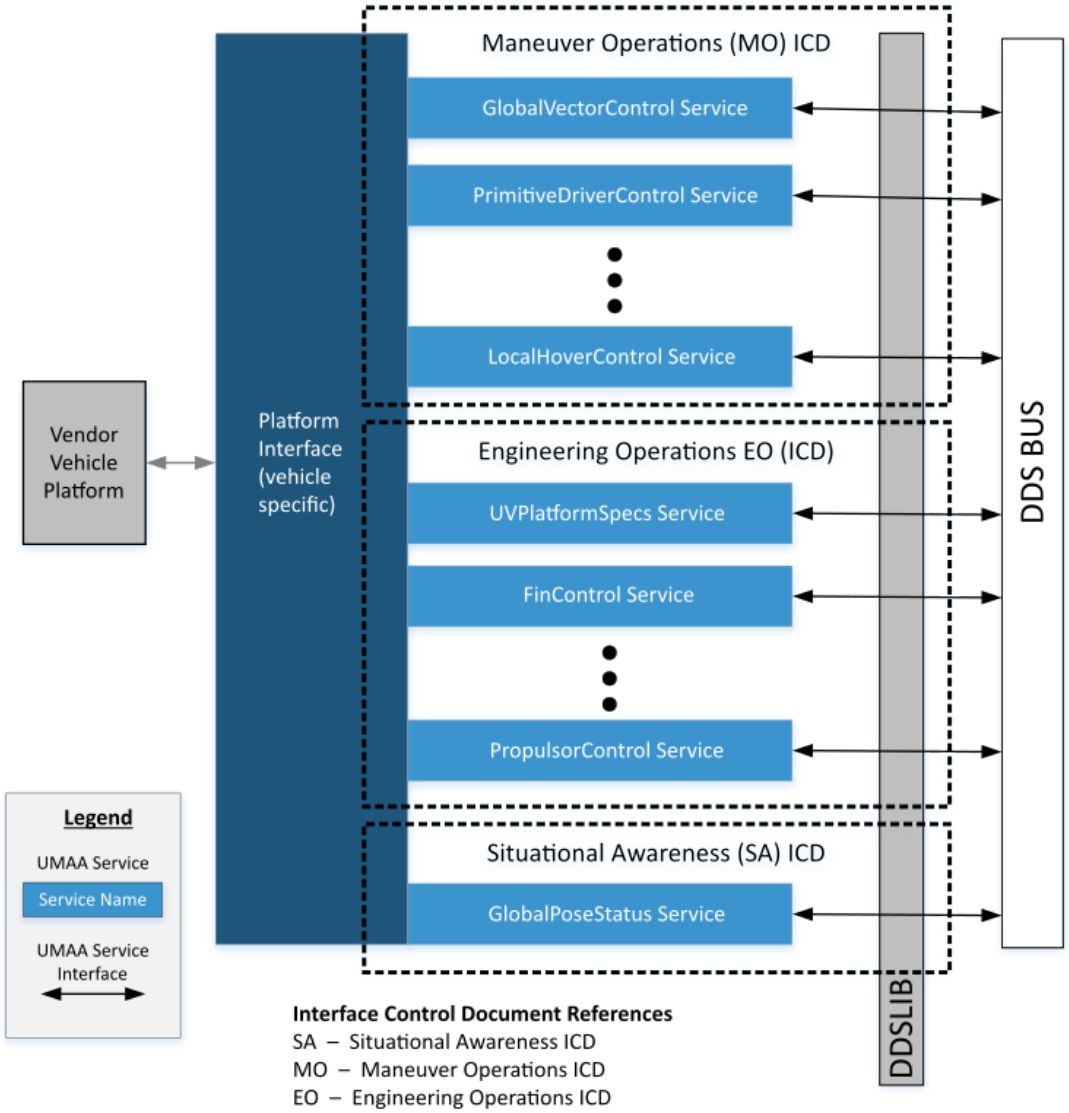

Figure 2: UMAA Services and Interfaces Example.

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 – Maneuver Operations (MO) 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.

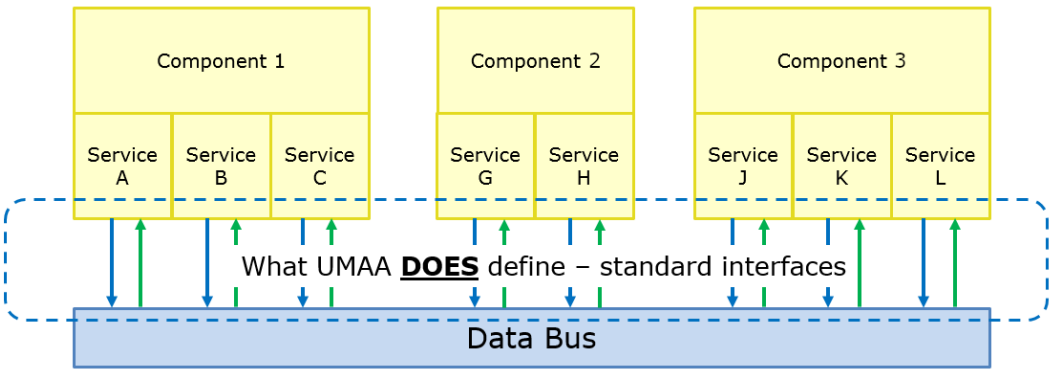

Figure 3: Services and Interfaces Exposed on the UMAA Data Bus.

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:

[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::MO::PrimitiveDriverControl
Access the ContactReport Service under Situational Awareness:
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`.

Note that Large Collection Metadata and Elements are sent on separate DDS topics. DDS QoS does not guarantee ordering across topics. For this reason, implementations must be able to handle cases where elements arrive before or after the associated metadata. Memory must be allocated to await the proper metadata and associated elements.

3.8.2 Creating Large Collections

To create a large collection, a series of element messages and a metadata message must be sent from one DDS participant (the sender) to another (the receiver). The messages should be buffered on the receiving side until a synchronization point is

reached which indicates an atomic update. That is, when both a metadata message and an element message corresponding by list ID, timestamp, and last element ID have been received, yield a complete collection. Figure 4 shows the sequence of exchanges to establish a collection with 3 elements.

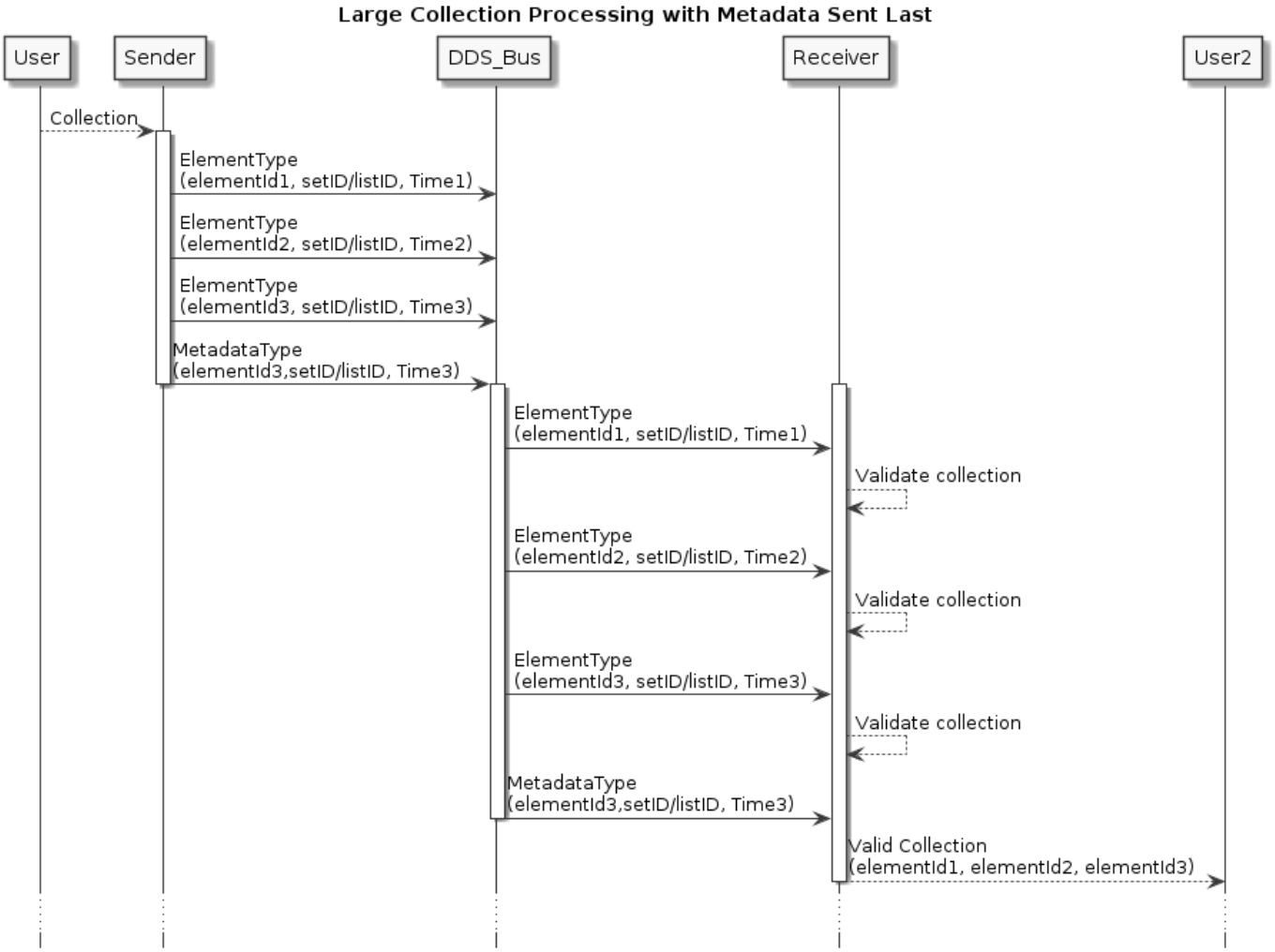

Figure 4: Sequence Diagram for initialization of a Large Collection with 3 elements.

The same collection could be established where the element data arrives after the metadata, creating the same list as depicted in figure 5.

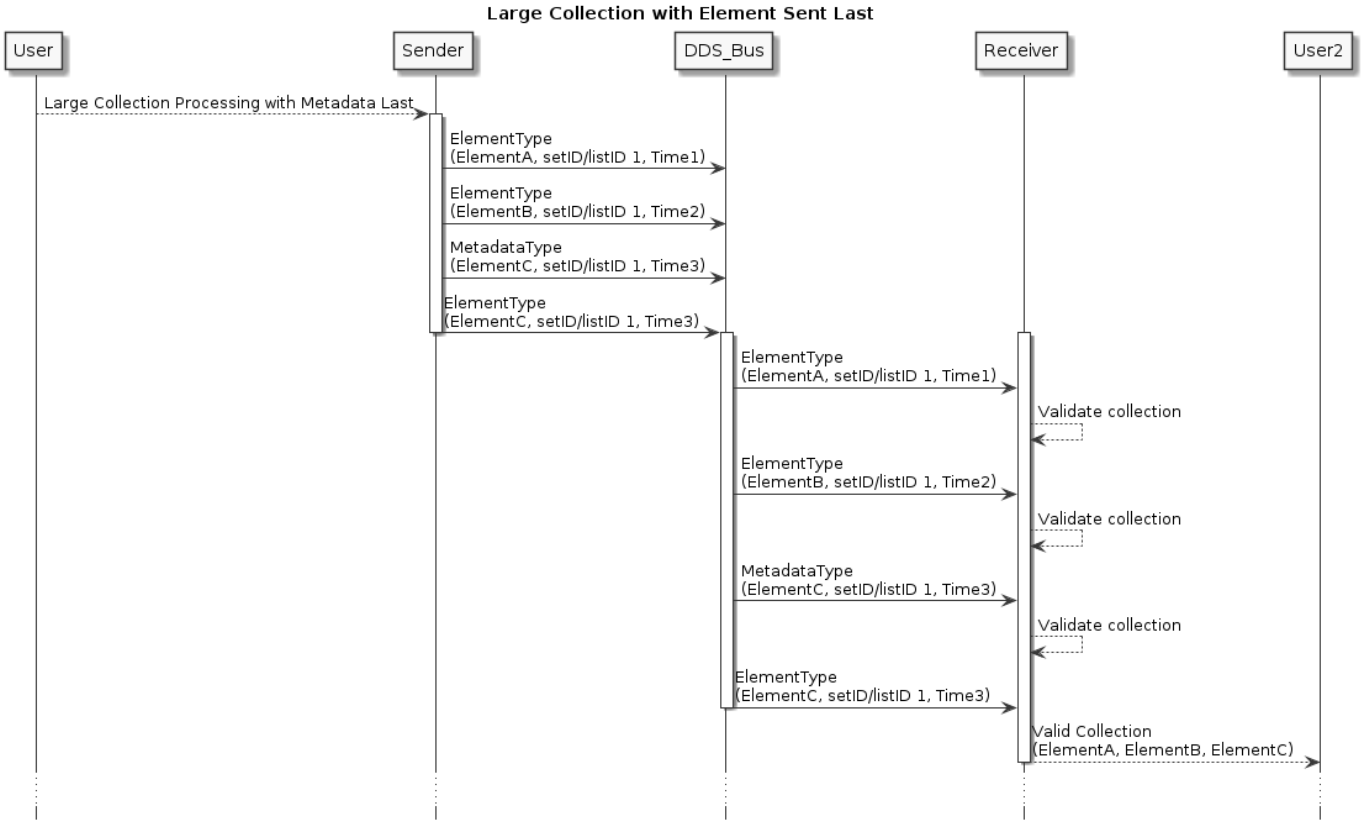

Figure 5: Sequence Diagram for initialization of a Large Collection with 3 elements.

3.8.3 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.

Figure 6 shows the sequence of exchanges to update a collection of 3 elements and add a 4th element.

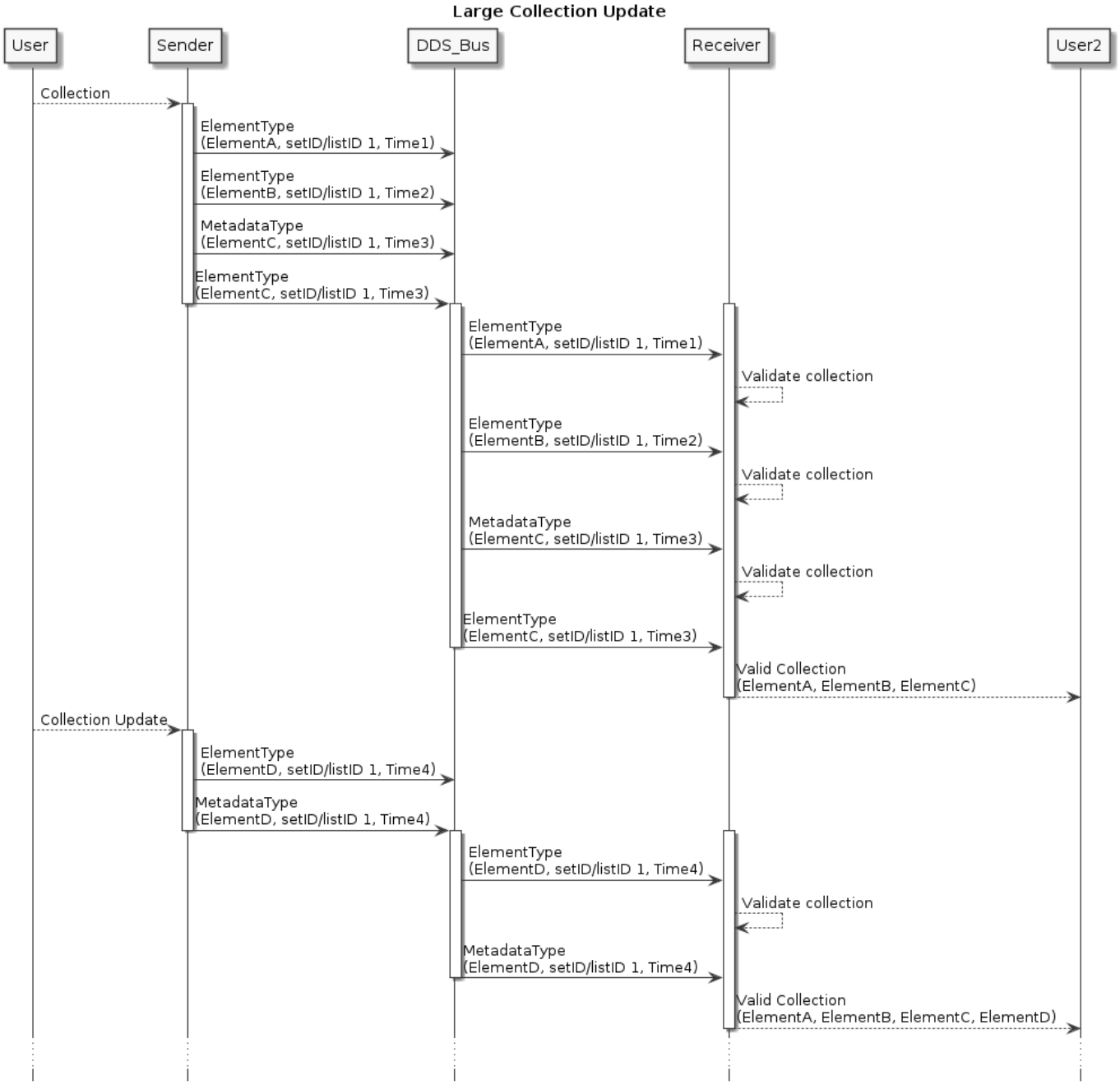

Figure 6: Sequence Diagram for update of Large Collection.

Figure 7 shows the sequence of exchanges to update an element of a collection multiple times.

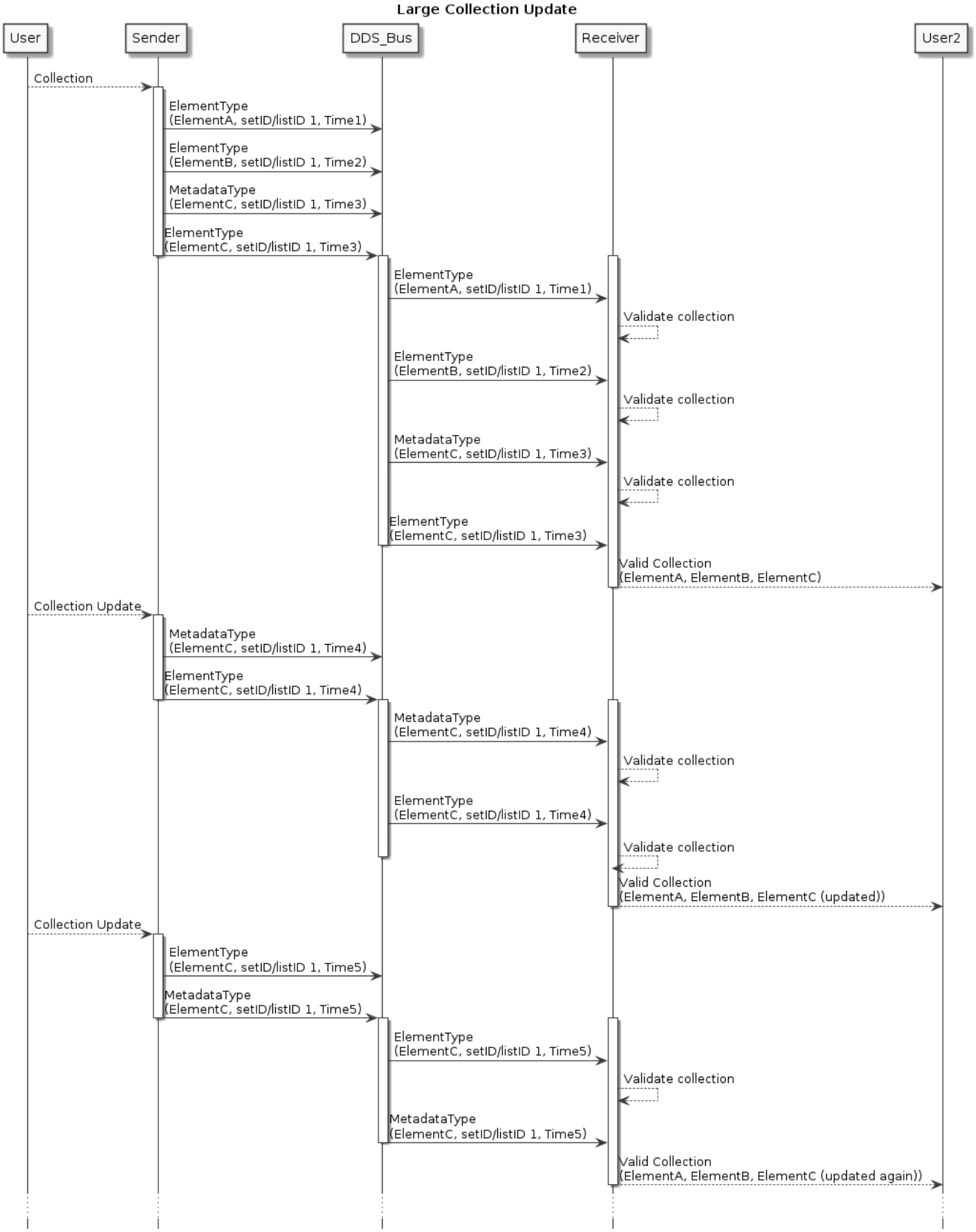

Figure 7: Sequence Diagram for update of an element of a Large Collection multiple times.

3.8.4 Removing an element from Large Collections

To remove an element from a collection, dispose of the element on the element topic and re-publish the metadata. Multiple deletes and inserts can happen for a single metadata update. In the case where the final element of the collection is deleted, the updateElementTimestamp should be unset in the metadata.

Figure 8 shows the sequence of exchanges to delete an element from a Large Collection.

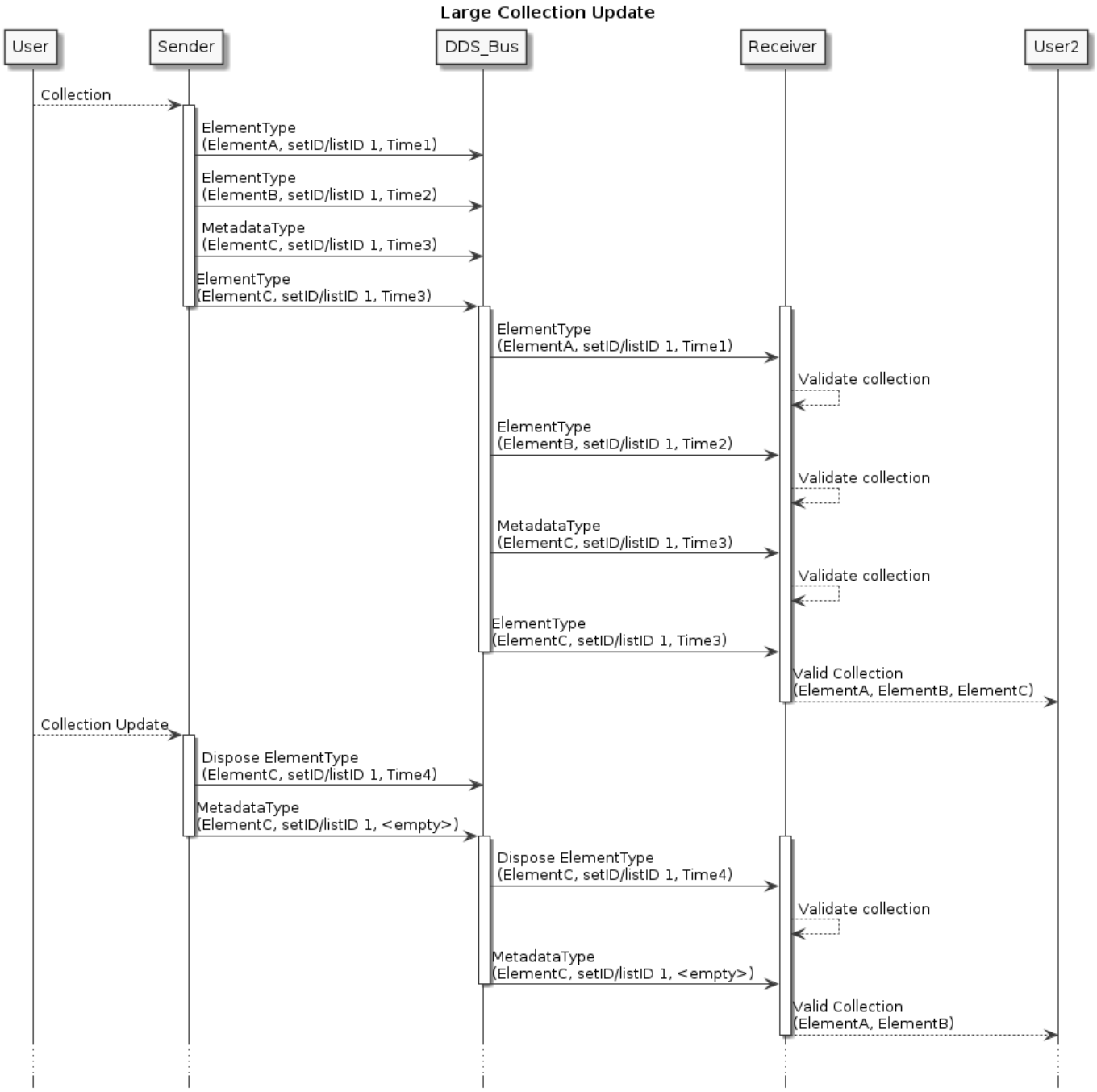

Figure 8: Sequence Diagram for delete of element from Large Collection.

For Large Lists, it may be necessary to update the nextElementID references during delete operations to ensure that the list is still valid. This would cause multiple element messages to be sent along with updated metadata.

3.8.5 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".

Figure 9 shows the sequence of exchanges to establish an initially empty Large Collection.

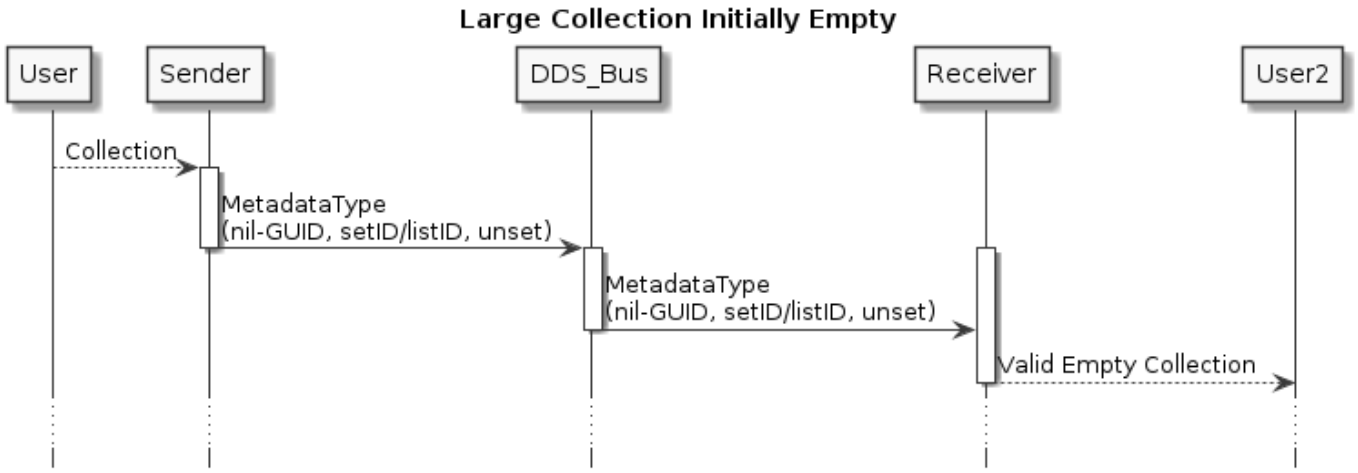

Figure 9: Sequence Diagram for initialization of an empty Large Collection.

3.8.6 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.

3.8.7 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.

Attribute Name	Attribute Type	Attribute Description
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.

3.9 Generalizations and Specializations

The UMAA standard makes use of generalization/specialization relationships when defining data types. The generalization/specialization relationship is one where a generalization data structure is defined to contain attributes that are common across some entity and specialization data structures are defined to contain attributes that are specific to a particular type of that entity. This relationship can be modeled as inheritance in UML as shown below.

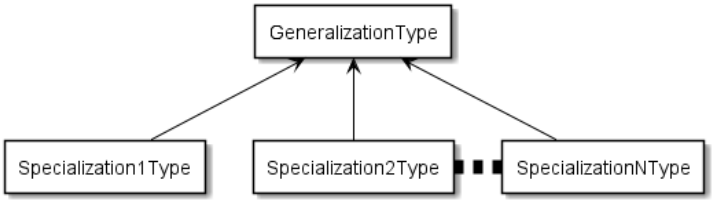

Figure 10: Generalization/Specialization UML diagram.

When the data type of an attribute within a message is a generalization, it is defined to be that generalization plus the data type of one of its specializations. In order to support this relationship, the generalization data structure and its specialization data structure are published to separate topics along with additional metadata linking the two topics. Specifically, the generalization data structure includes: specializationTopic, specializationID, and specializationTimestamp; and the specialization data structure includes: specializationID and specializationTimestamp. The specializationTopic specifies the topic name of the particular specialization, and the specializationID and specializationTimestamp must be equivalent in each topic, respectively, in order to establish the generalization/specialization relationship.

3.9.1 Creating a generalization/specialization

To create a generalization/specialization, both the GeneralizationType and SpecializationType topics must be sent from one DDS participant (the sender) to another (the receiver). The topics should be buffered on the receiving side until a synchronization point is reached that indicates an atomic update.

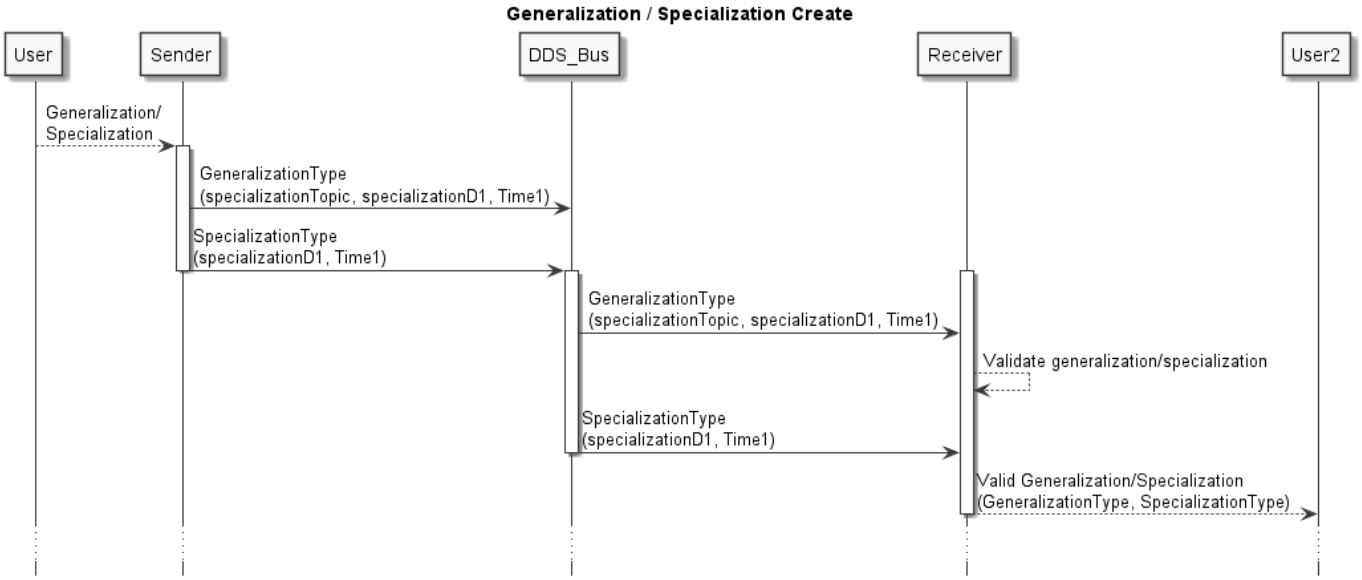

Figure 11: Sequence diagram for creating a generalization/specialization.

3.9.2 Updating a generalization/specialization

An update to a generalization/specialization can occur when there is a change in either data structure. In order for the update to be complete, the specializationTimestamp must be updated in both the GeneralizationType and the SpecializationType, and again they must be equal. Note that if a generalization/specialization exists within a large set or large list that their respective metadata must also be updated as defined in Section 3.8.

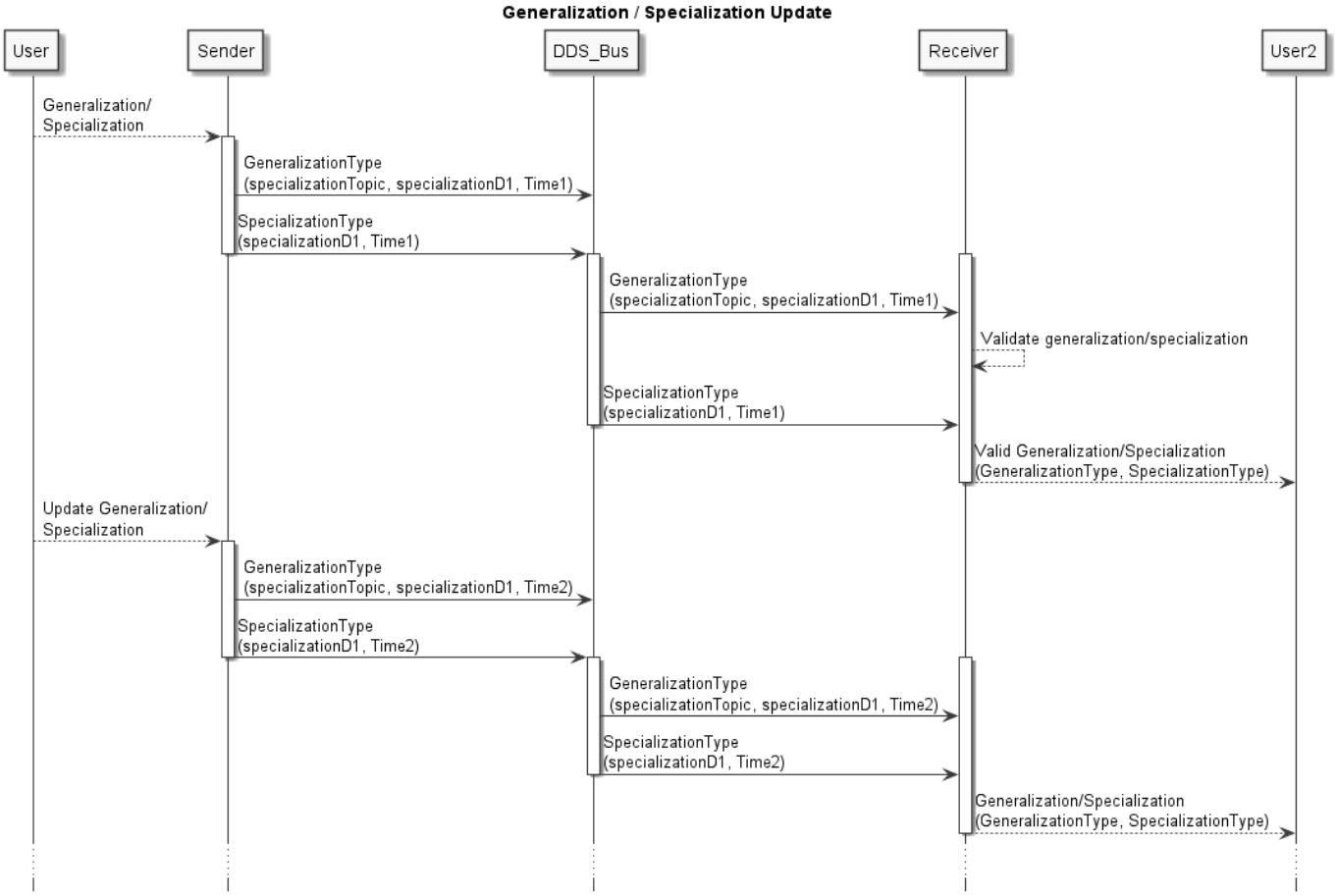

Figure 12: Sequence diagram for updating a generalization/specialization.

3.9.3 Removing a generalization/specialization

To remove a generalization/specialization, both topics must be disposed. Again, note that if a generalization/specialization exists within a large set or large list that their respective metadata must also be updated as defined in Section 3.8.

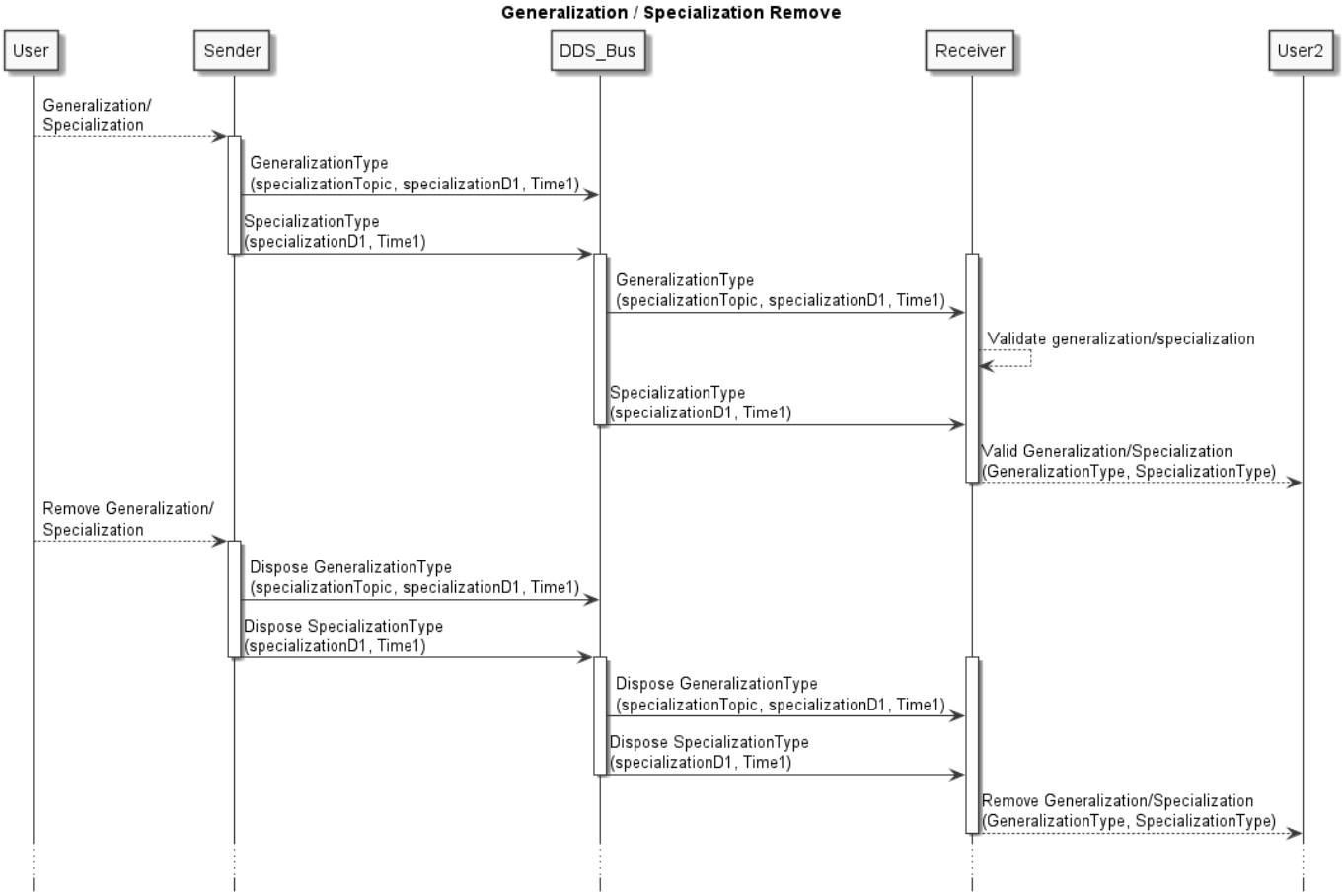

Figure 13: Sequence diagram for removing a generalization/specialization.

4 Introduction to Coordinate Reference Frames and Position Model

4.1 Platform Reference Frame

In the following Service Definitions, we use the parameters yaw, pitch, and roll to define the platform 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 14). 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.

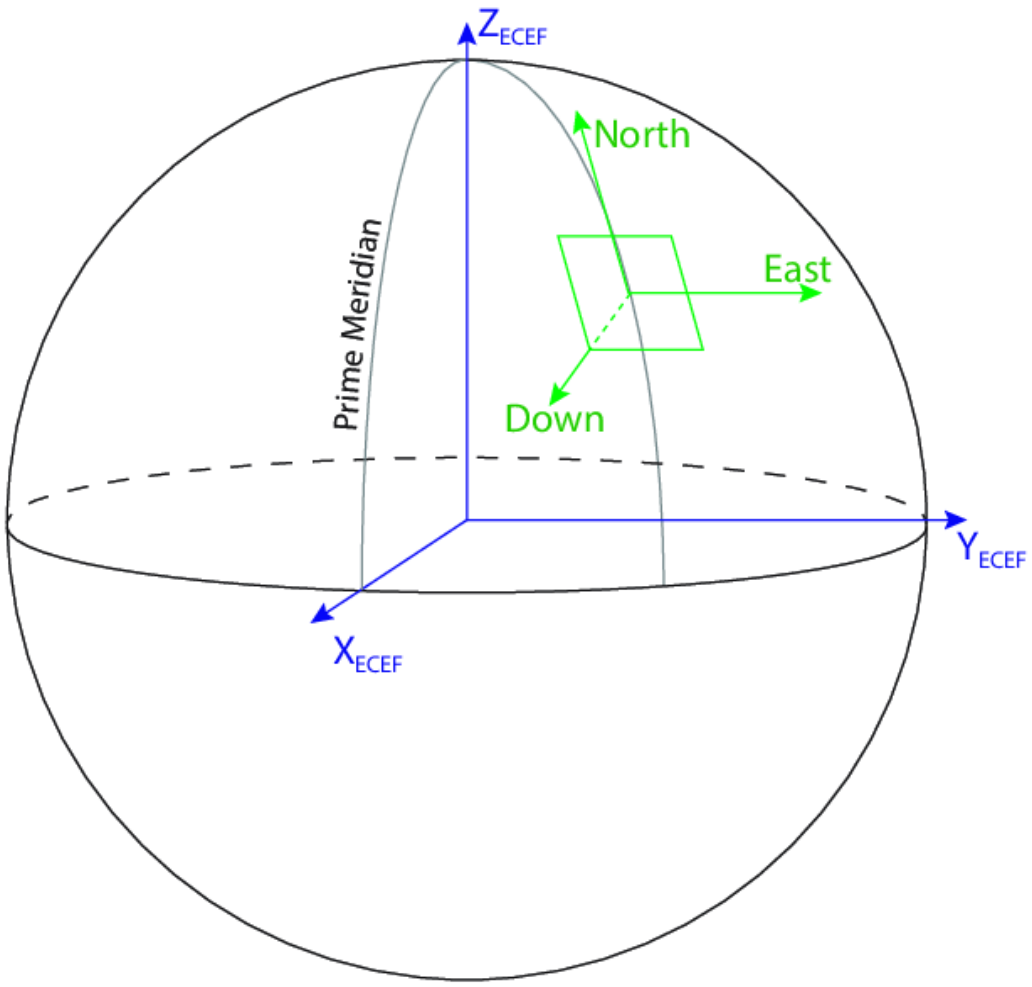

Figure 14: Origins and axes of the Earth-Centered Earth-Fixed (ECEF) and North-East-Down (NED) frames.

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 15) with respect to any reference frame is carried out via the following procedure (Figure 16).

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 17).

5. Rotate the vehicle about the vehicle's newly oriented Y axis by the pitch angle (Figure 18).
6. Rotate the vehicle about the vehicle's newly oriented X axis by the roll angle (Figure 19).

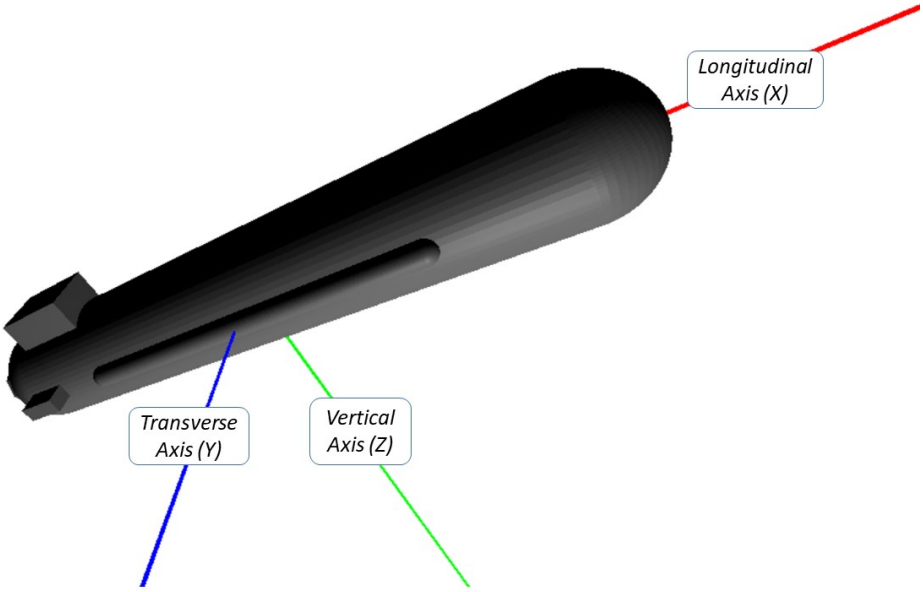

Figure 15: Define the Vehicle Coordinate System

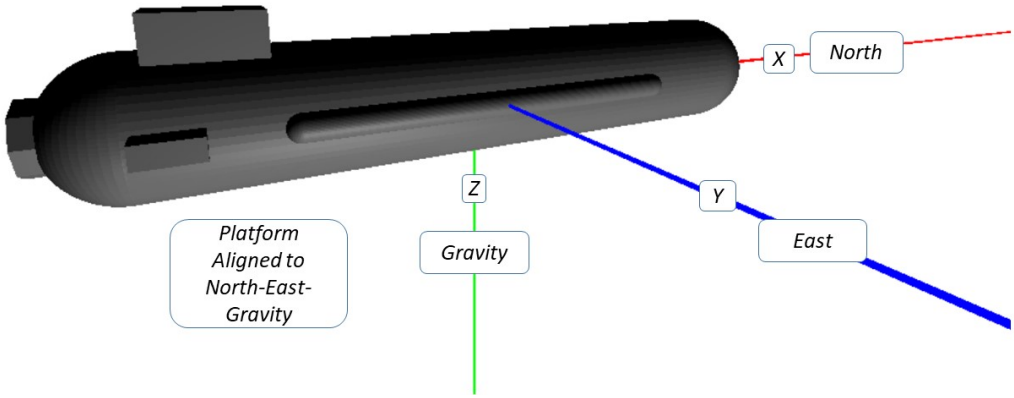

Figure 16: Align the Vehicle with the Reference Frame Axes.

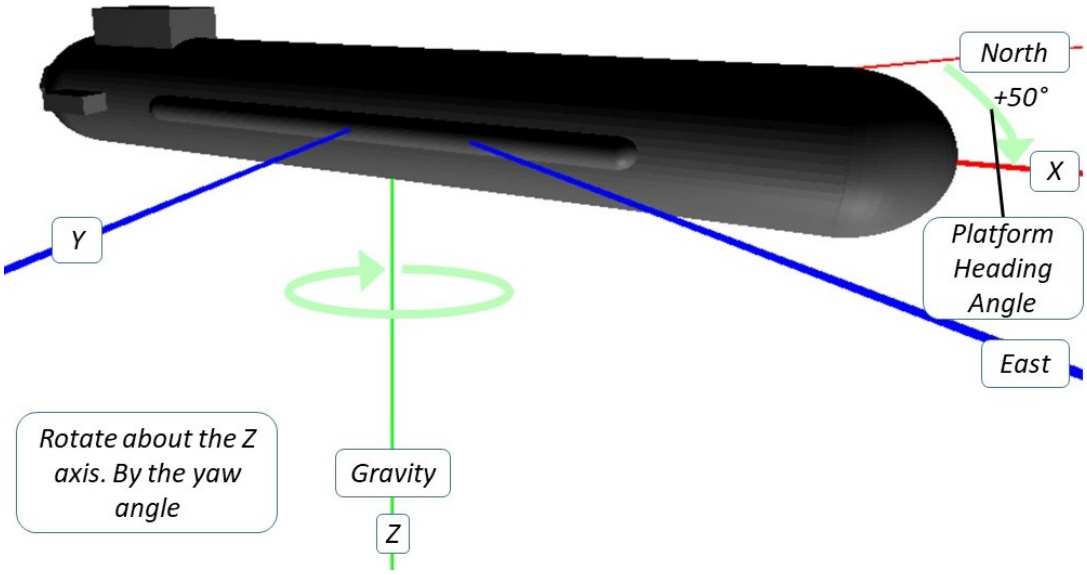

Figure 17: Rotate the Vehicle by the Yaw Angle.

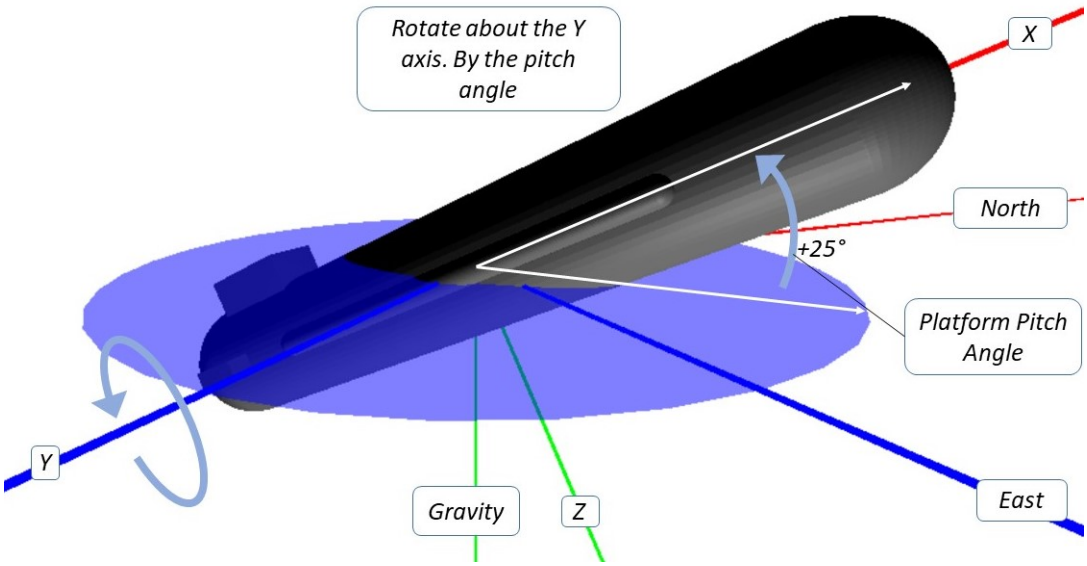

Figure 18: Rotate the Vehicle by the Pitch Angle.

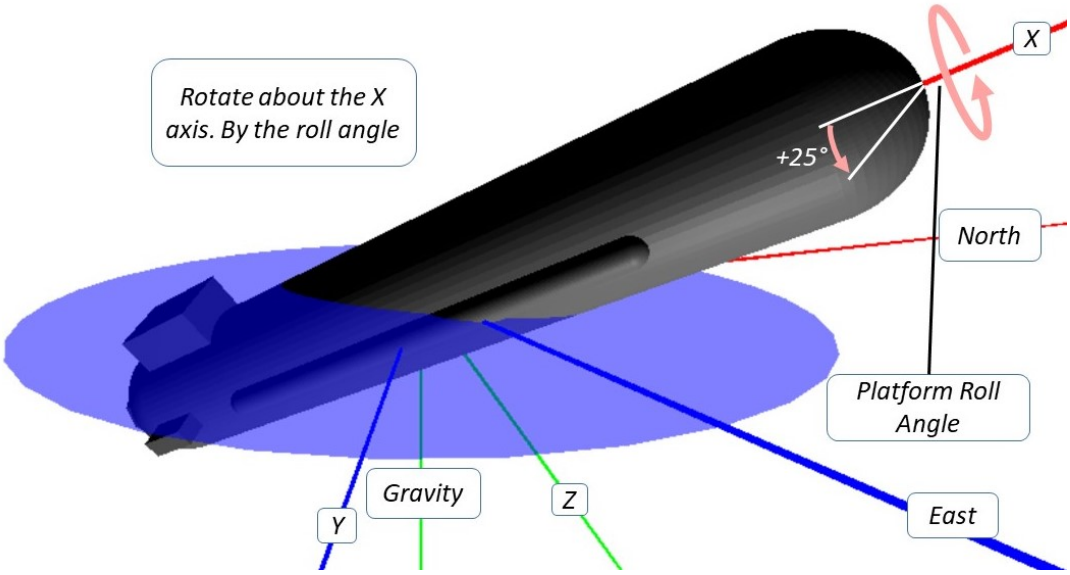

Figure 19: Rotate the Vehicle by the Roll Angle.

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.

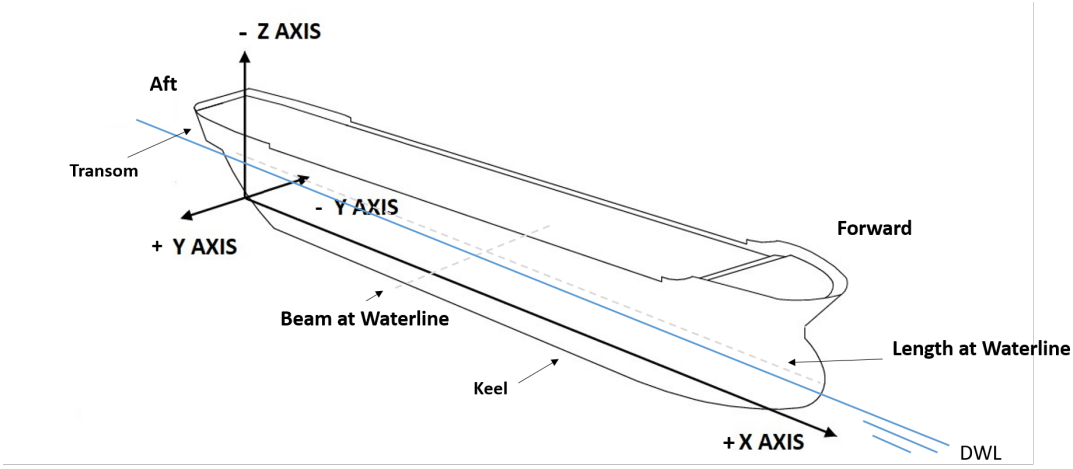

Figure 20: Keel Transom Intersection Origin Location on a USV as Example

- 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.

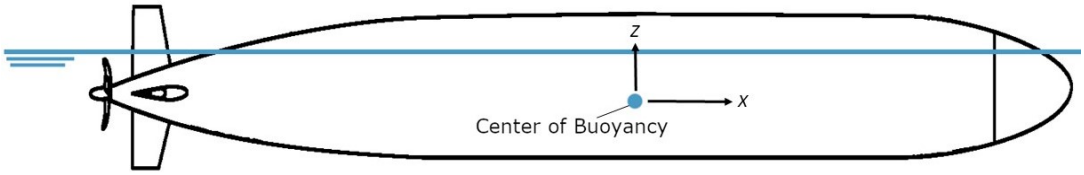

Figure 21: Center of Buoyancy Origin Location on a UUV as Example.

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 22. 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.

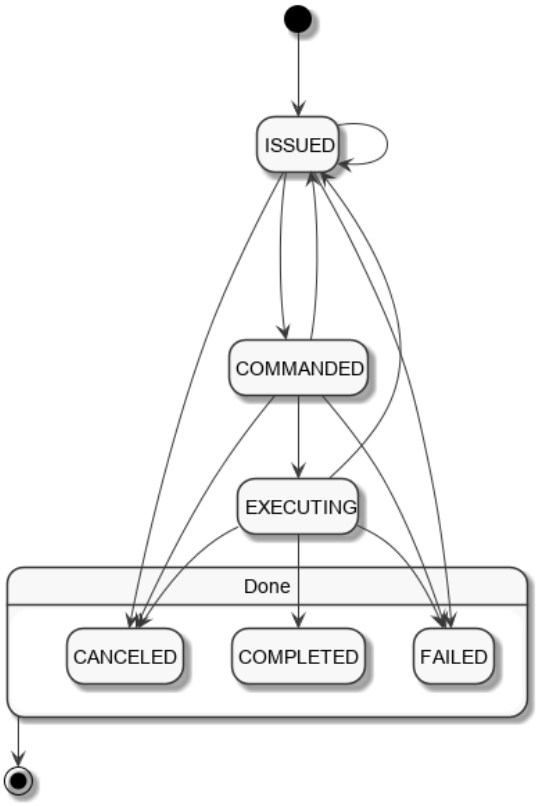

Figure 22: State transitions of the `commandStatus` as commands are processed.

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 23: 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 24 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.

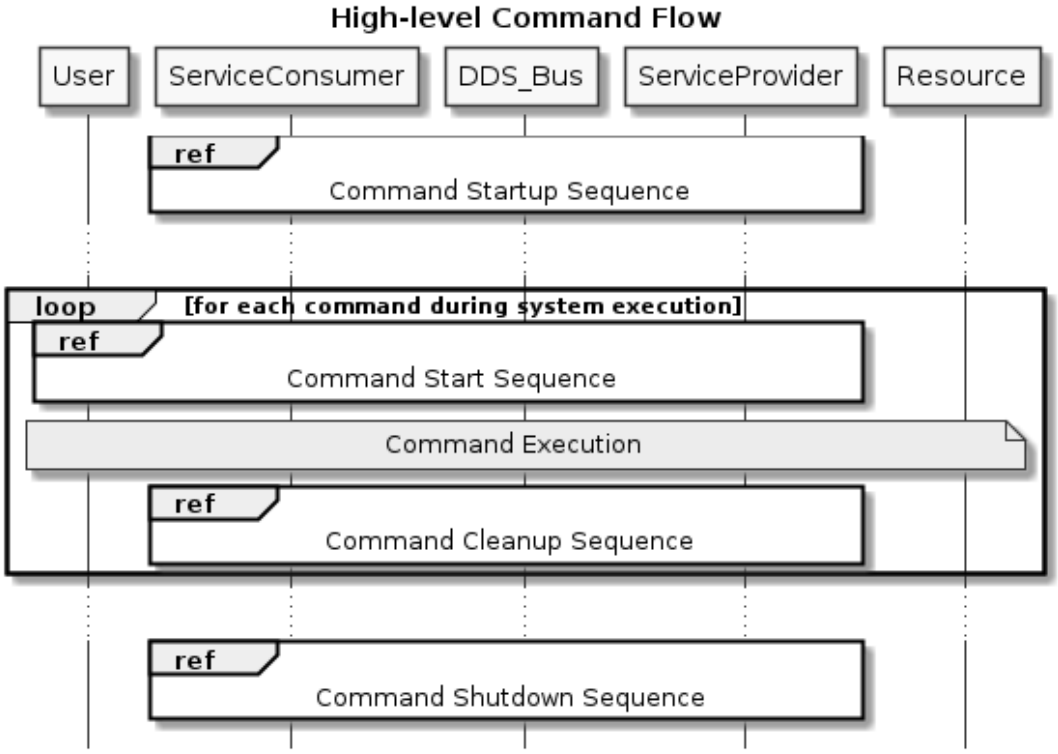

Figure 24: Sequence Diagram for the High-Level Description of a Command Execution.

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 25.

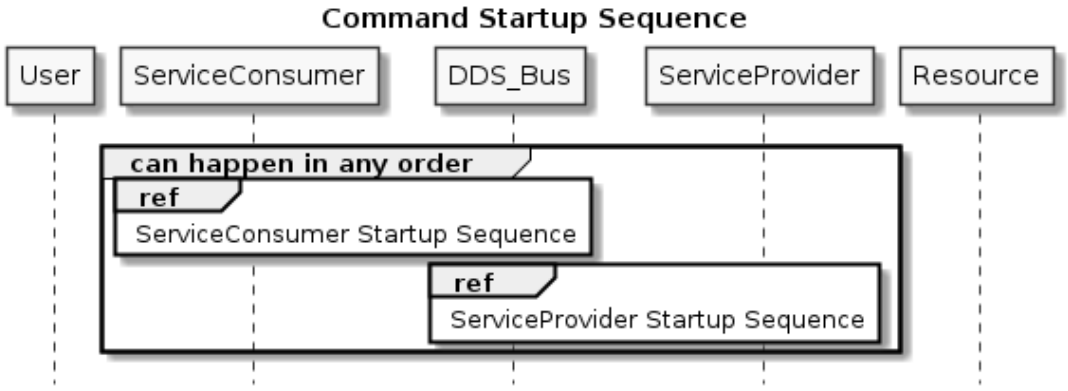

Figure 25: Sequence Diagram for Command Startup.

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 26.

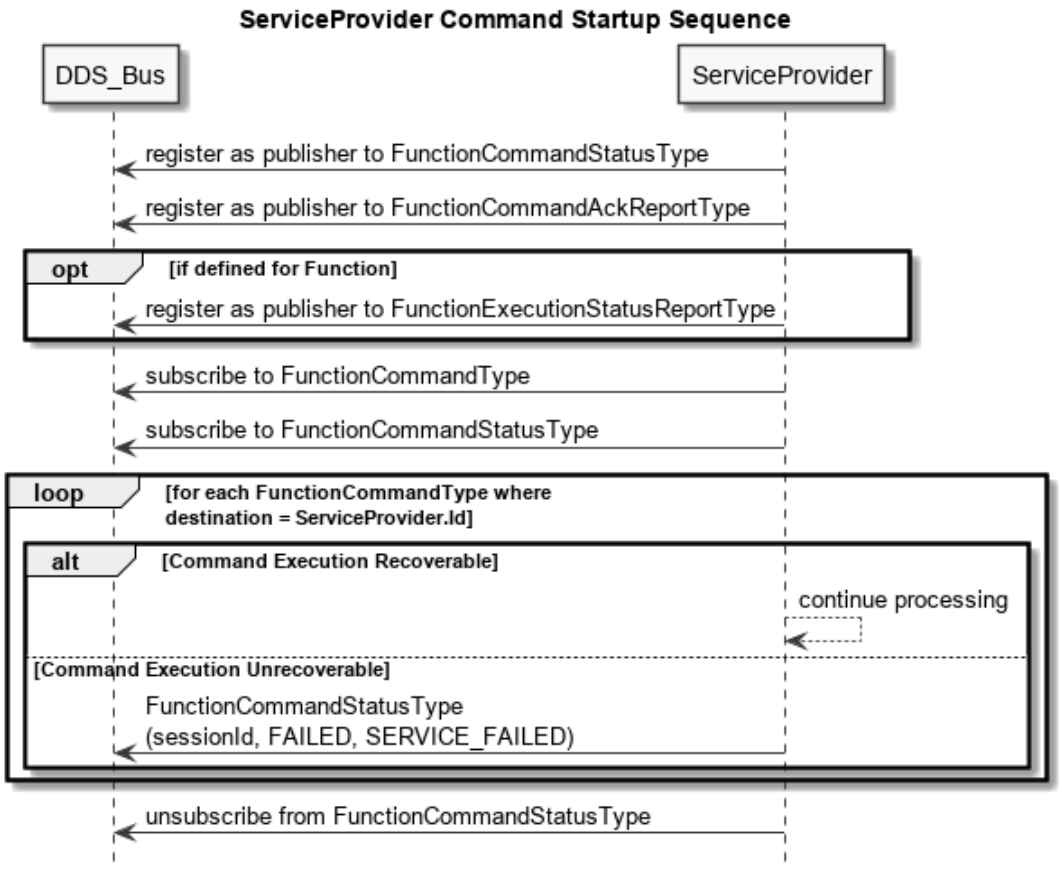

Figure 26: Sequence Diagram for Command Startup for Service Providers.

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 27.

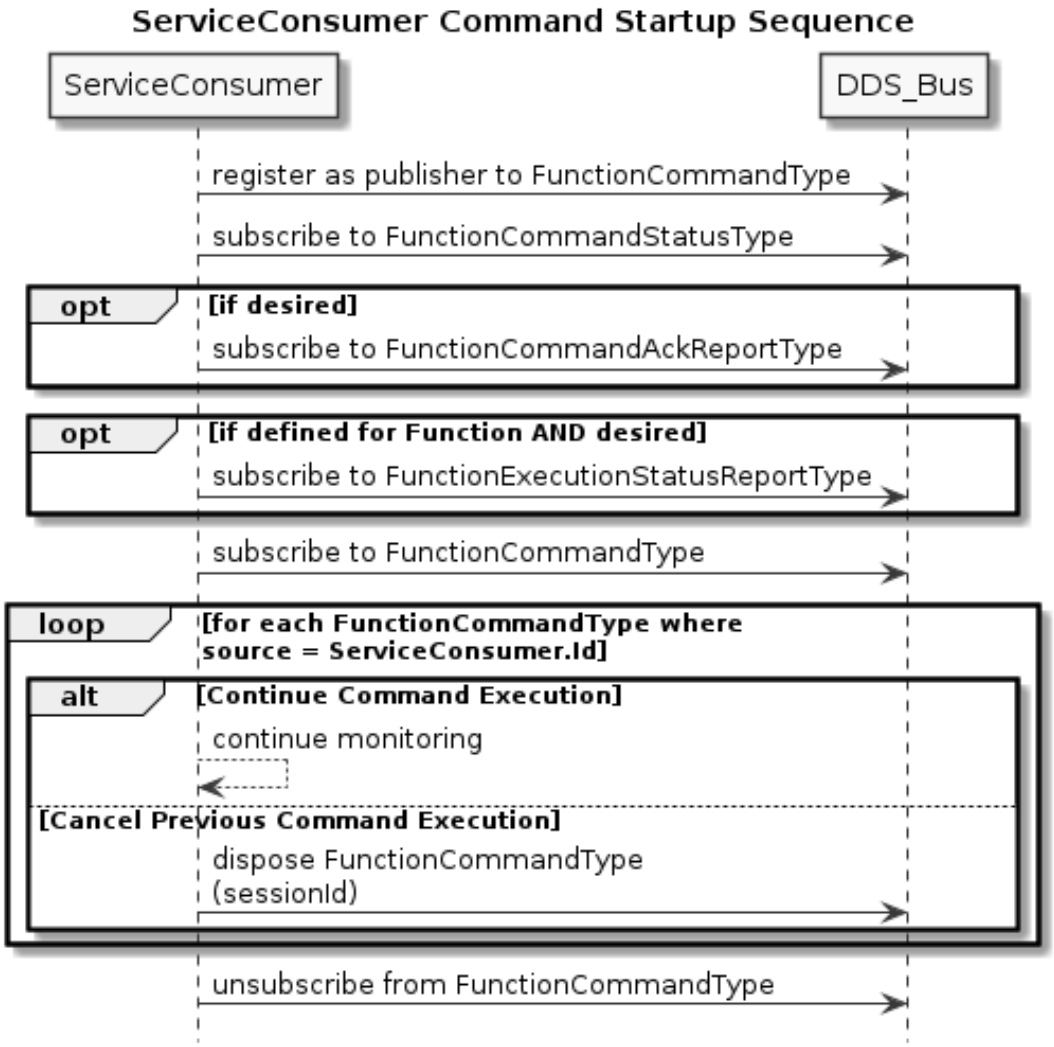

Figure 27: Sequence Diagram for Command Startup for Service Consumers.

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 28. 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 22 and eventually end with the Command Cleanup Sequence (shown in Figure 35).

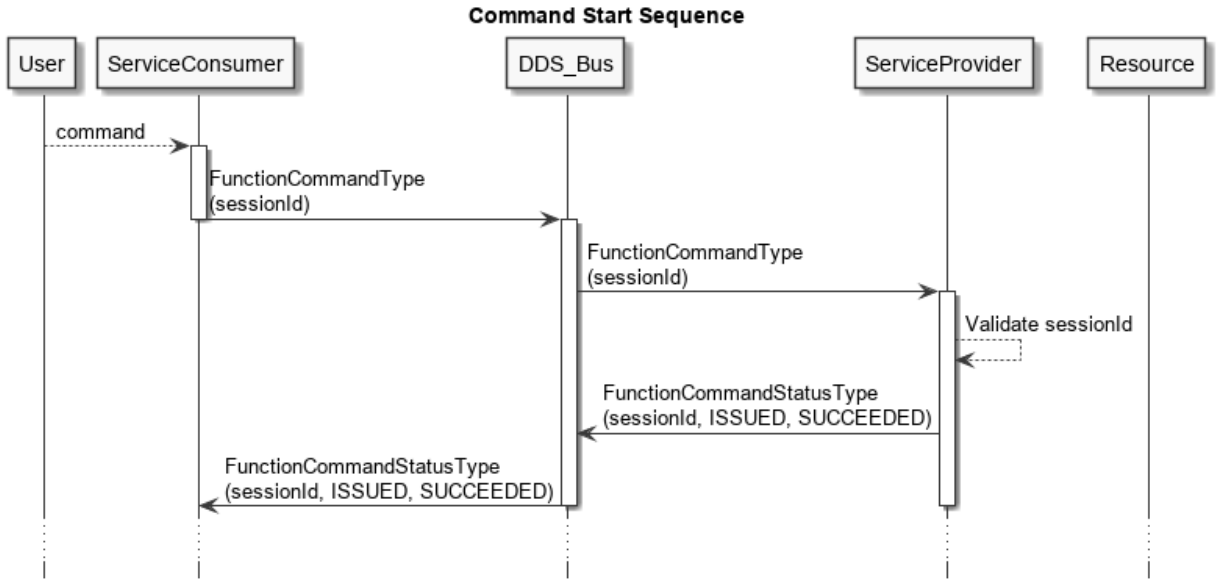

Figure 28: Sequence Diagram for the Start of a Command Execution.

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 29. This sequence holds until the command completes execution.

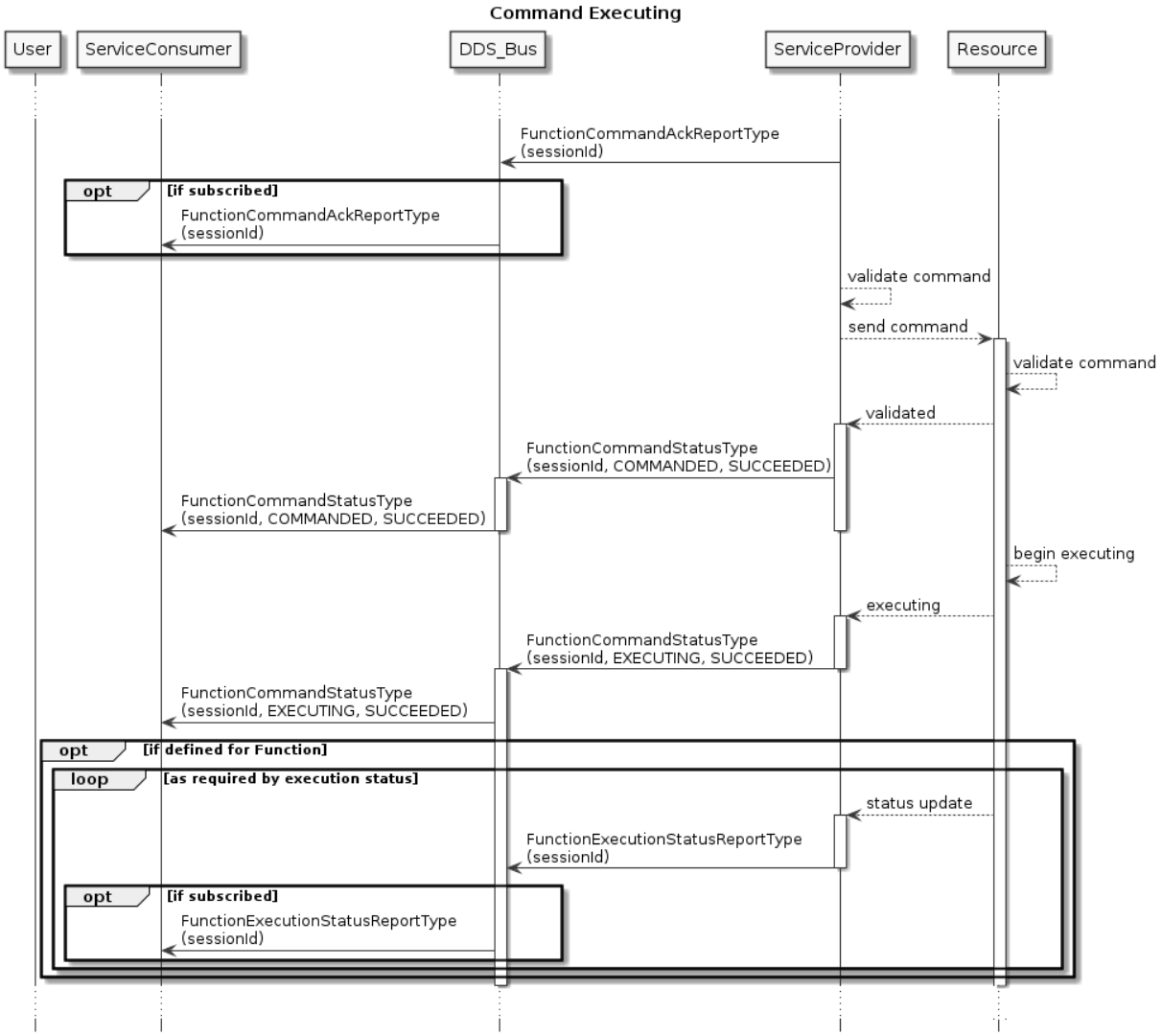

Figure 29: Beginning Sequence Diagram for a Command Execution.

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. If a command is in a terminal state, the Service Provider must ignore an update to that command.

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:

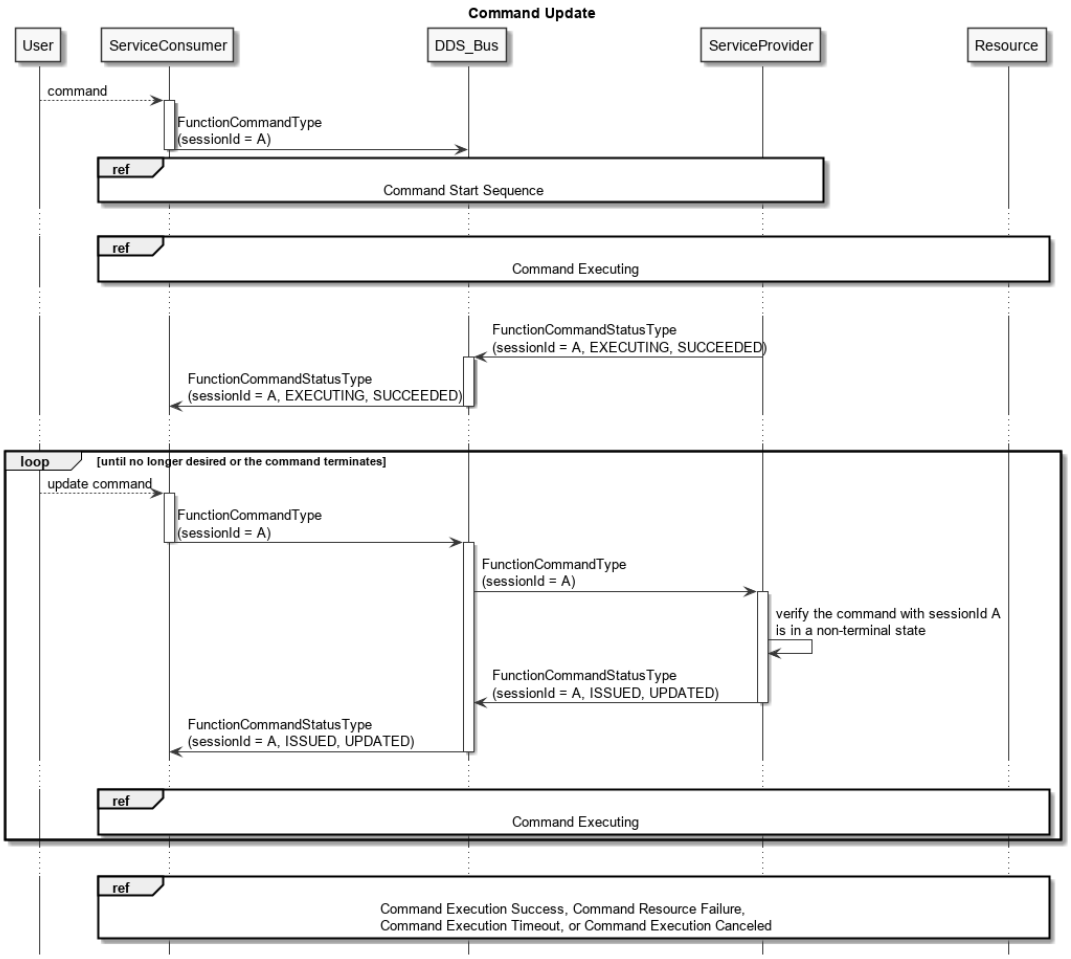

Figure 30: Sequence Diagram for Command Update.

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 31.

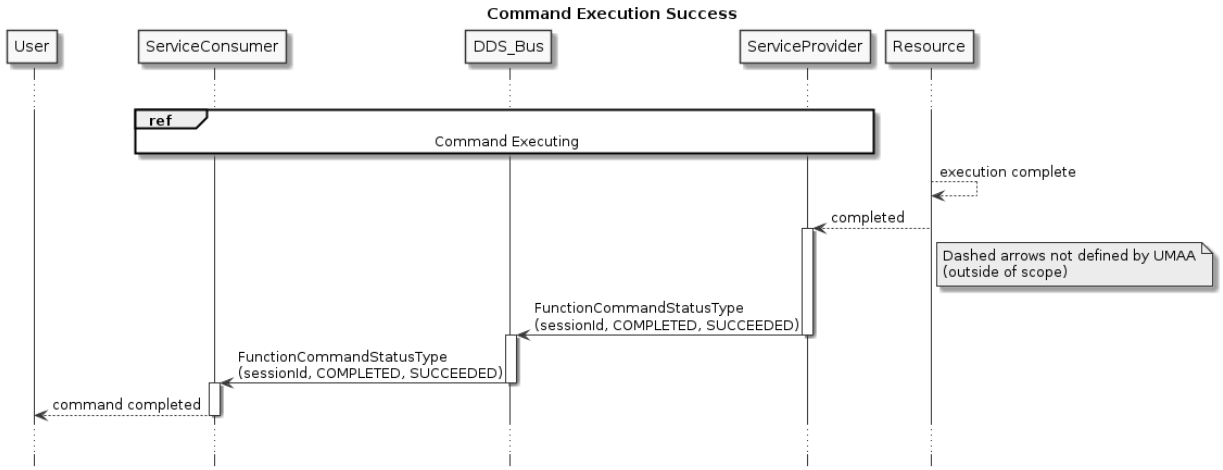

Figure 31: Sequence Diagram for a Command That Completes Successfully.

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 32 and Figure 33 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 32.

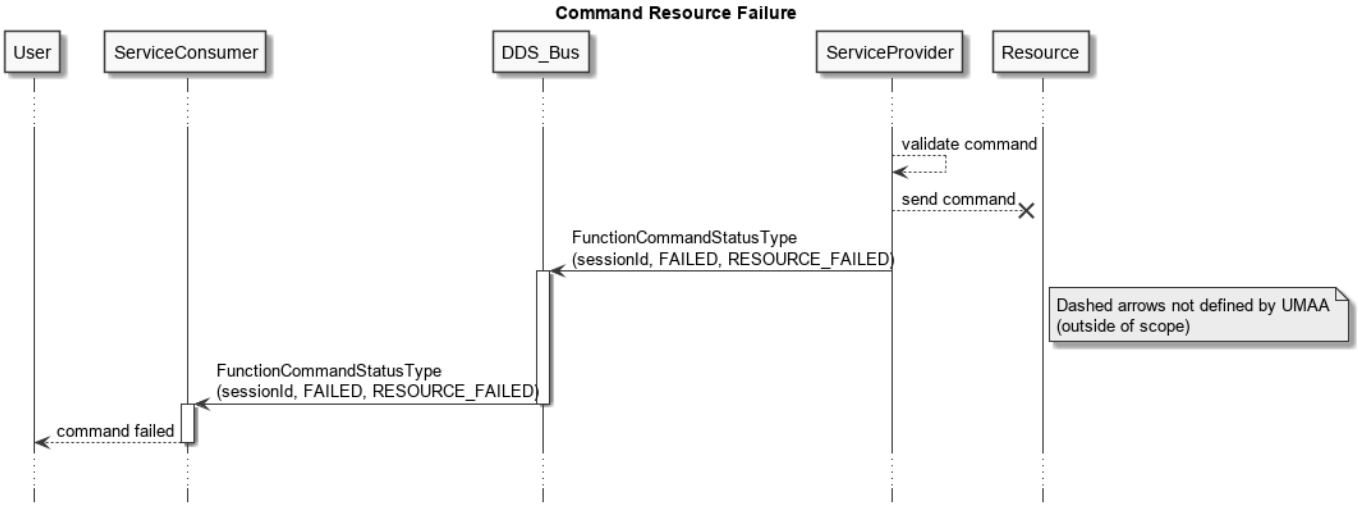

Figure 32: Sequence Diagram for a Command That Fails due to Resource Failure.

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 33.

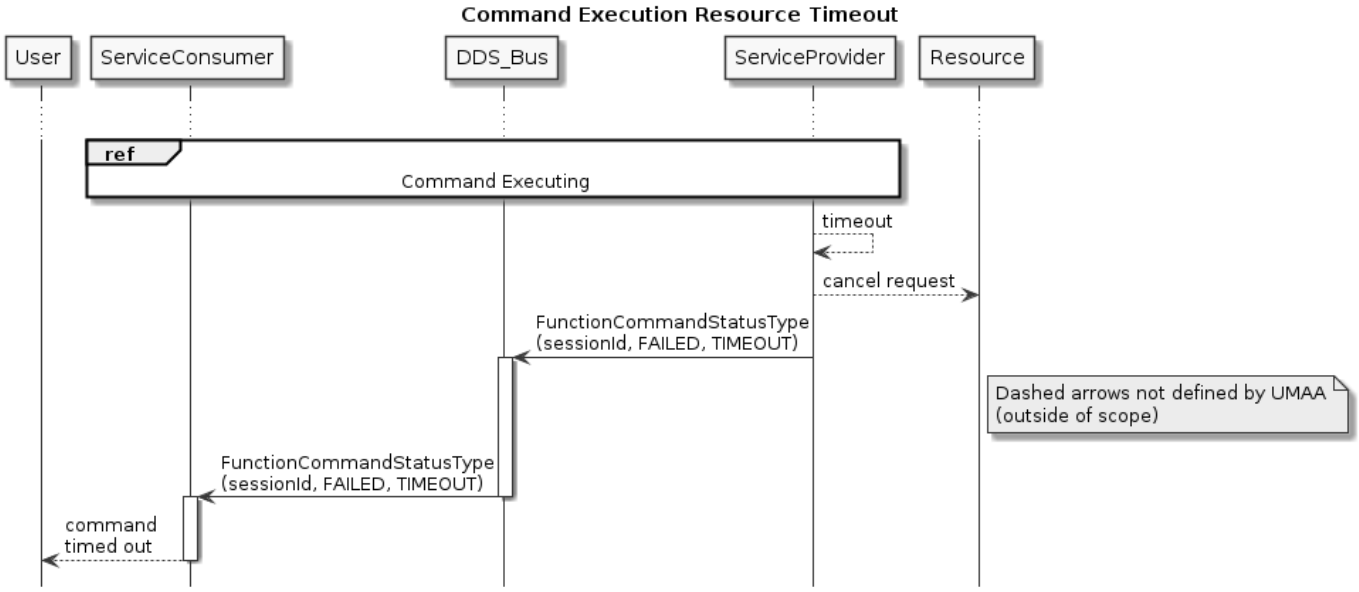

Figure 33: Sequence Diagram for a Command That Times Out Before Completing.

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 34. 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.

On loss of liveliness of a Service Provider while executing a command, all Service Consumers must cancel (dispose) all in-process commands with that Service Provider.

On loss of liveliness of a Service Consumer while executing a command, all Service Providers must treat the command as canceled. This means the service should report the `CANCELED` status for the command, and then dispose the command status, ack, and execution status (if one exists).

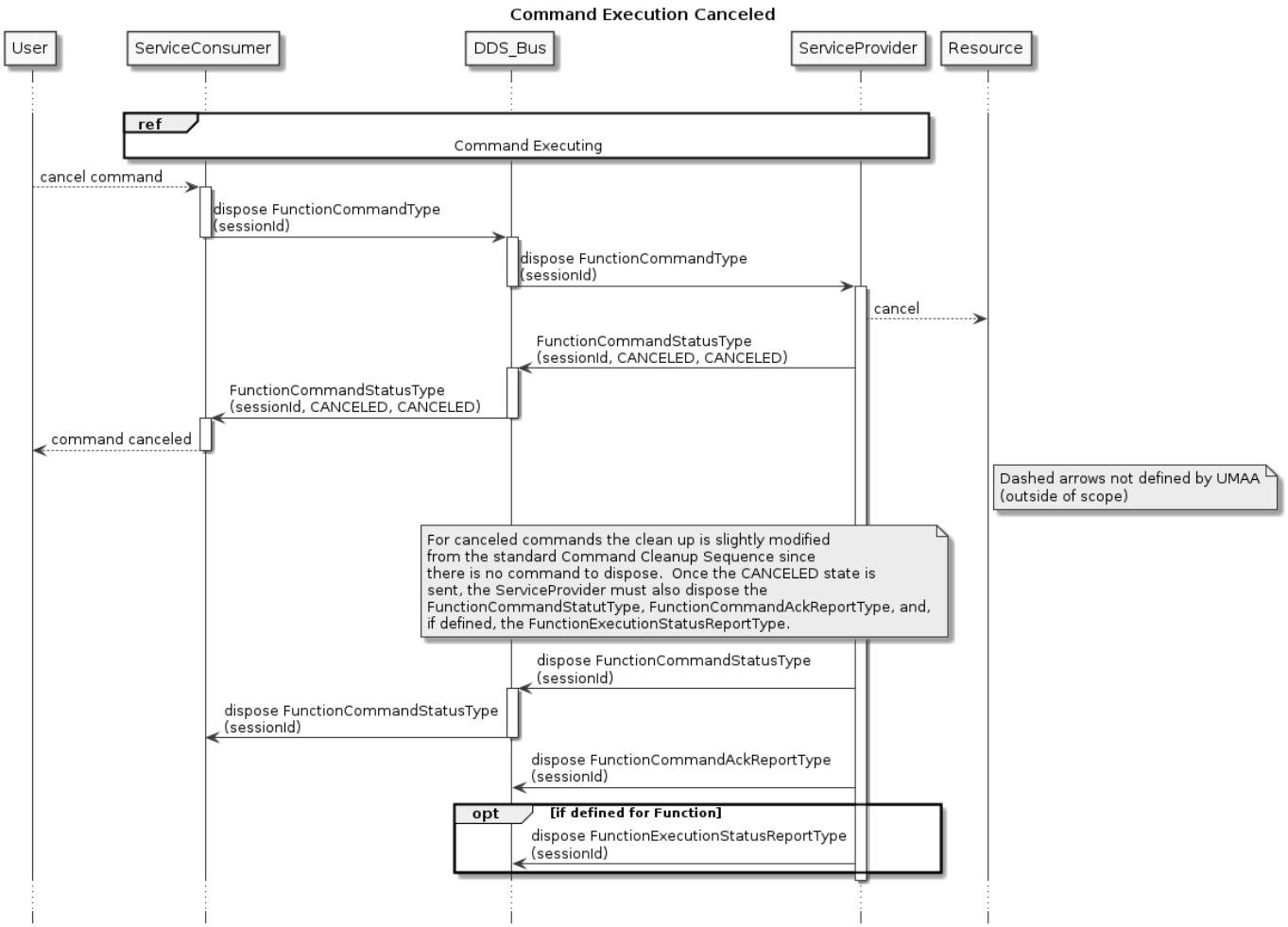

Figure 34: Sequence Diagram for a Command That is Canceled by the Service Consumer Before the Service Provider can Complete It.

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 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 35.

[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.

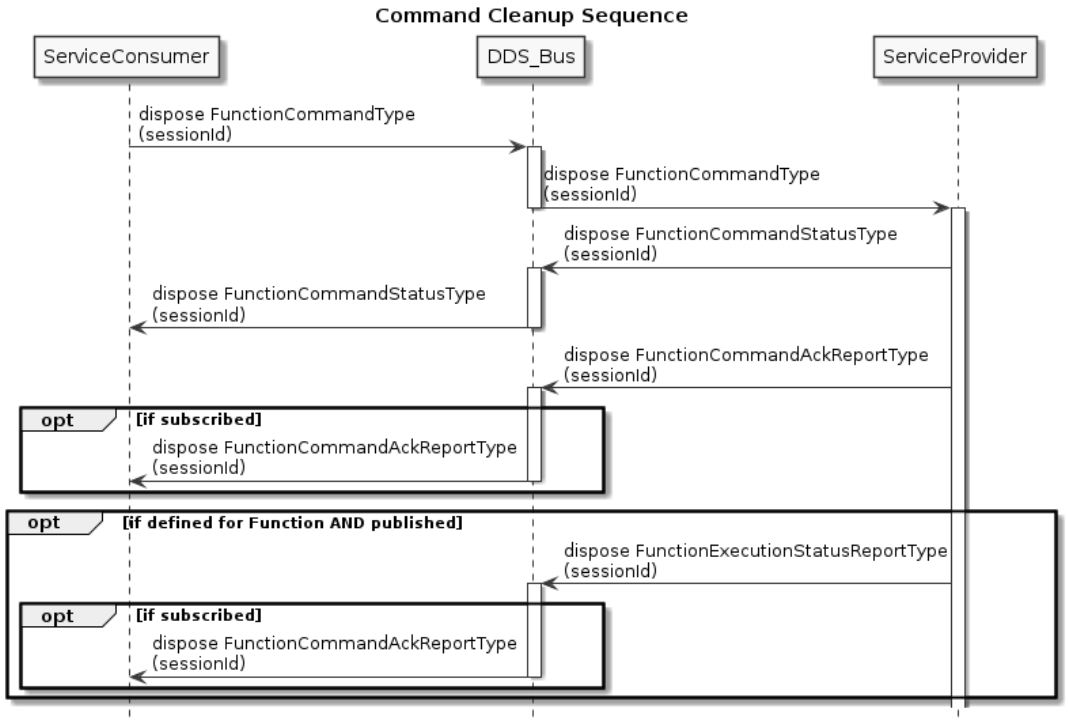

Figure 35: Sequence Diagram Showing Cleanup of the Bus When a Command Has Been Completed and the Service Consumer No Longer Wishes to Maintain the Commanded State.

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 36.

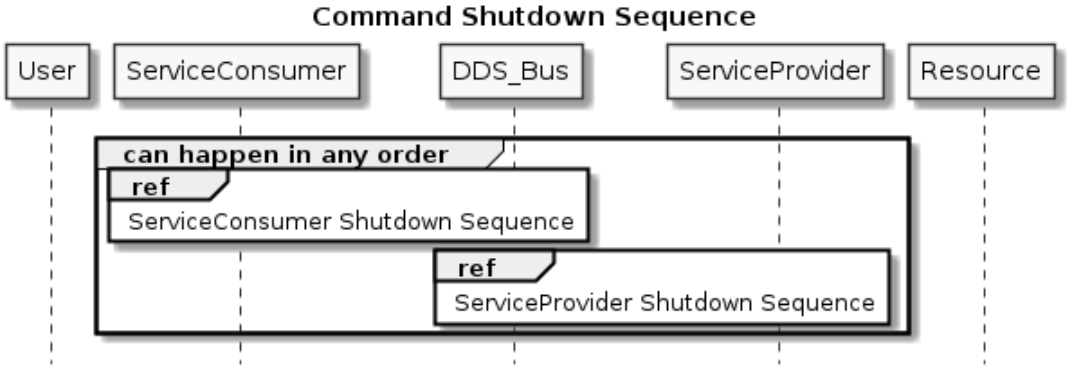

Figure 36: Sequence Diagram for Command Shutdown.

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 37.

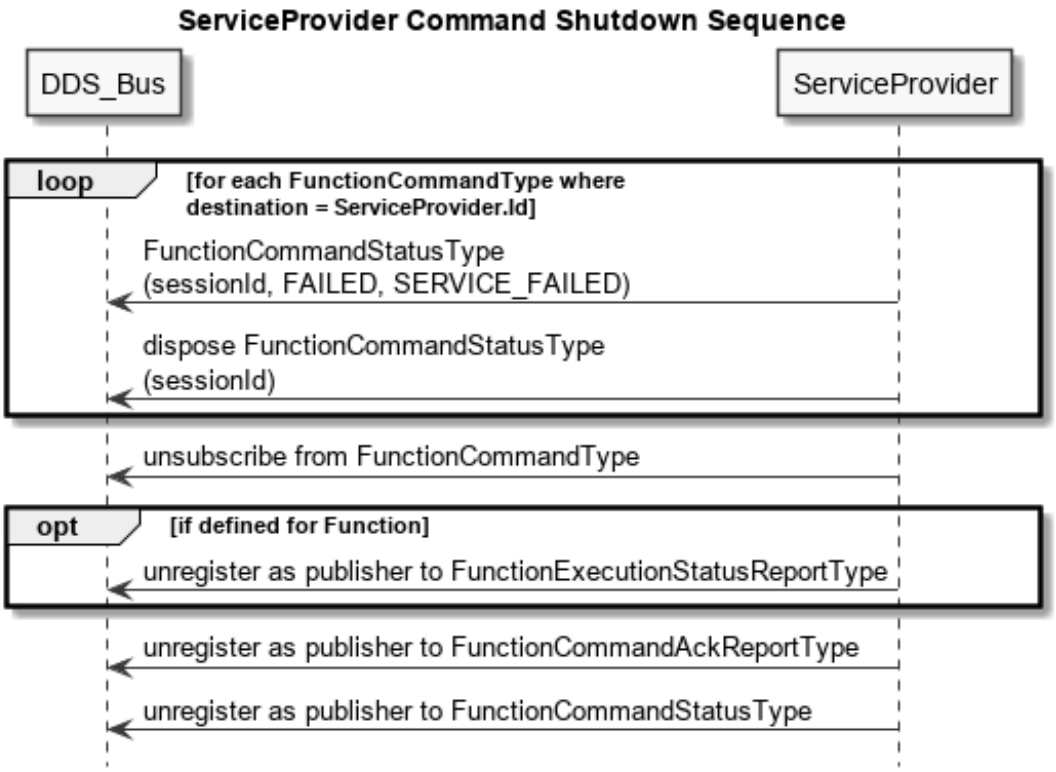

Figure 37: Sequence Diagram for Command Shutdown for Service Providers.

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 38.

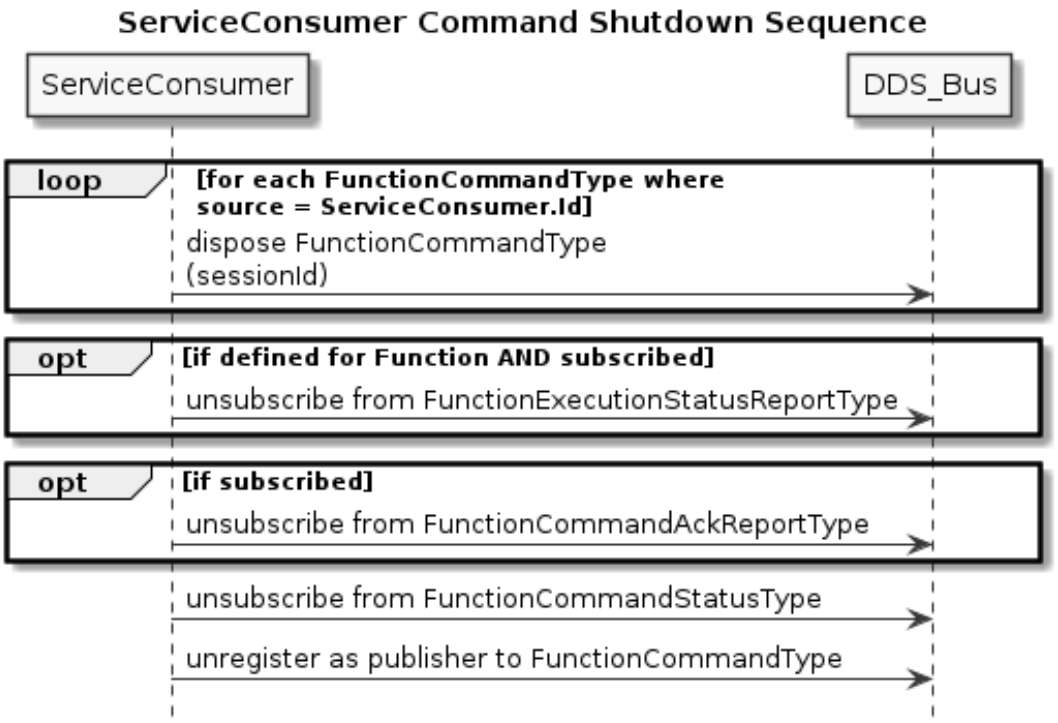

Figure 38: Sequence Diagram for Command Shutdown for Service Consumers.

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 39 through 43 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 39 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).

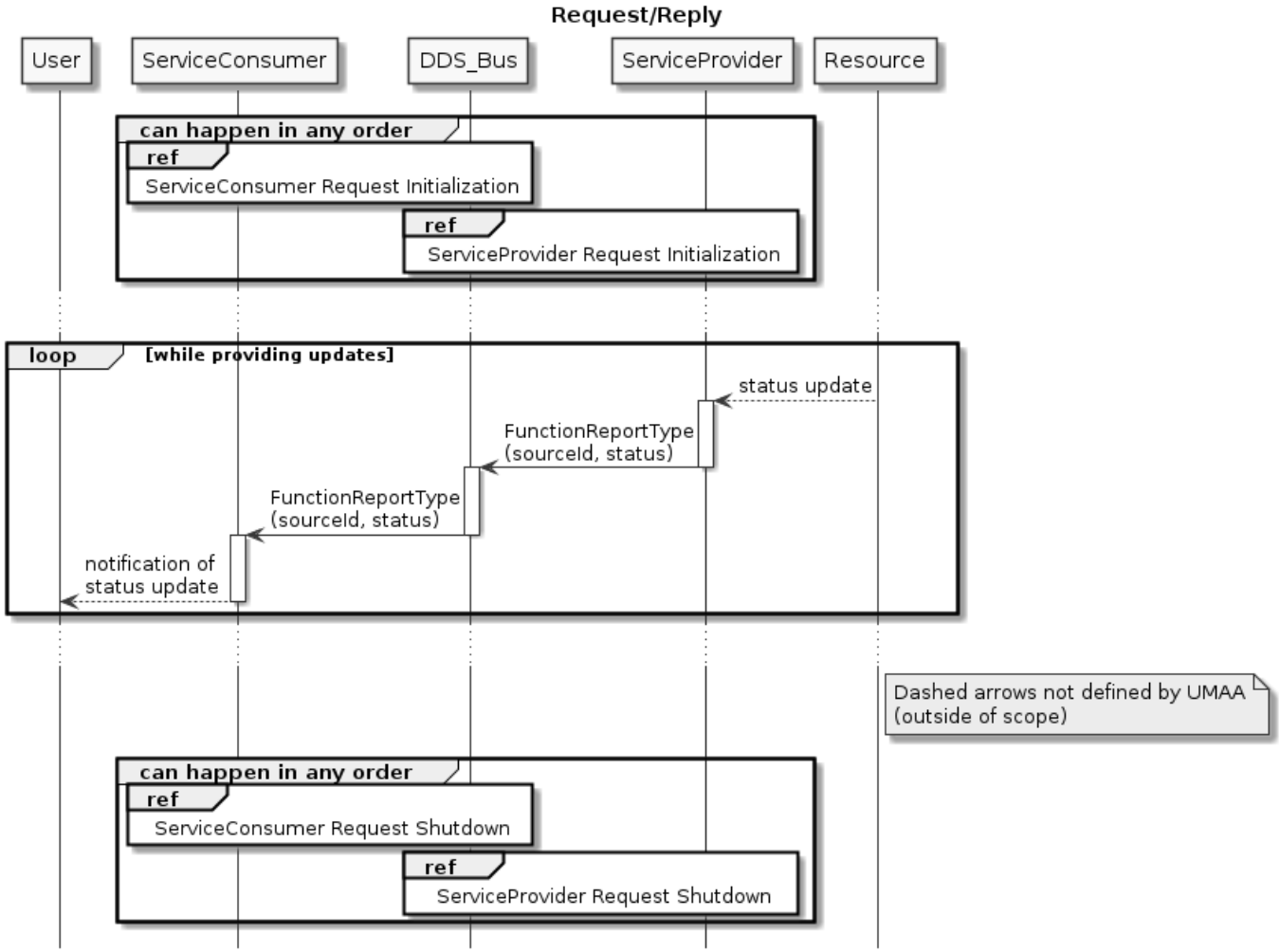

Figure 39: Sequence Diagram for a Request/Reply for Report Data That Does Not Require any Specific Query Data.

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 40.

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.

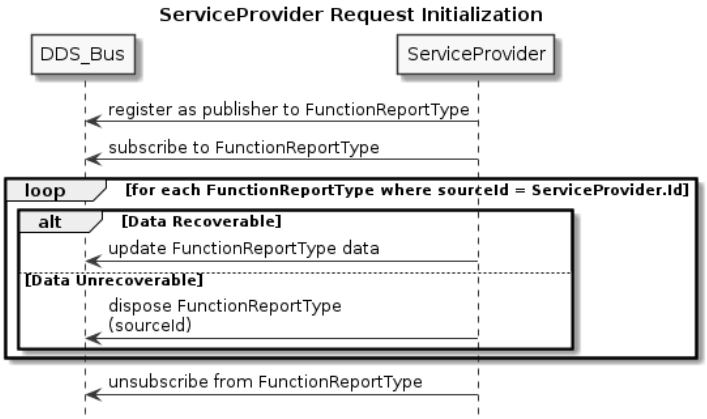

Figure 40: Sequence Diagram for Initialization of a Service Provider to Provide `FunctionReportType`s.

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 41.

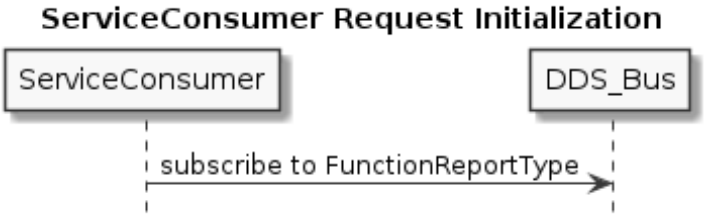

Figure 41: Sequence Diagram for Initialization of a Service Consumer to Request `FunctionReportType`s.

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 42).

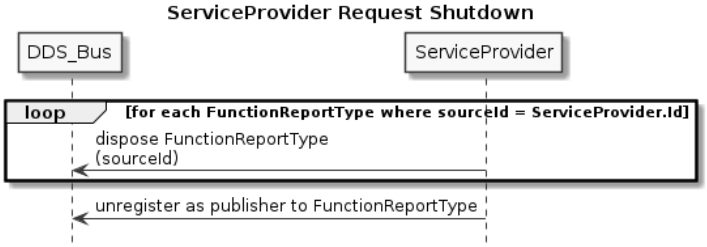

Figure 42: Sequence Diagram for Shutdown of a Service Provider.

5.2.1.4 Service Consumer Shutdown To no longer request `FunctionReportType`s, the Service Consumer unsubscribes from `FunctionReportType` (shown in Figure 43).

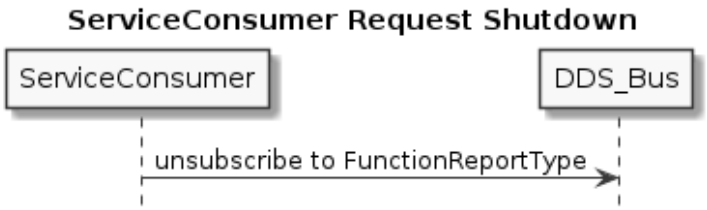

Figure 43: Sequence Diagram for Shutdown of a Service Consumer.

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 Maneuver Operations (MO) 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.

A standard feature of the maneuver operations driver services is that a new driving control command to a service overrides the previous driver command to that service.

Operations without DDS Topics

⊕ = Operations that are handled directly in DDS

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 ContactManeuverInfluenceStatus

The purpose of this service is to report the influence contacts have on the maneuvering of ownship.

Table 8: ContactManeuverInfluenceStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryContactManeuverInfluence⊕	reportContactManeuverInfluence

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

6.1.1.1 reportContactManeuverInfluence

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

Namespace: UMAA::MO::ContactManeuverInfluenceStatus

Topic: ContactManeuverInfluenceReportType

Data Type: ContactManeuverInfluenceReportType

Table 9: ContactManeuverInfluenceReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
influence	ContactManeuverInfluenceEnumType	Specifies the influence the associated contact is having on the maneuvering of ownship. If a contact could be considered to have multiple influences at the same time (e.g., CROSSING_LEFT_COMPLIANT and GUIDE) the more severe status is reported. Severity order for this purpose, from most to least severe, is: COLLISION, COLLISION AVOIDANCE, any COLREGS status, DYNAMIC AVOIDANCE, STATIC AVOIDANCE, GUIDE, PREEMPTIVE, and NONE.
contactID*	NumericGUID	An identifier of the contact.

6.1.2 CoordinationSituationalSignalStatus

The purpose of this service is to provide the current coordination status of this vessel with respect to situational signals (coordinating with other vehicles).

Table 10: CoordinationSituationalSignalStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryCoordinationSituationalSignal⊕	reportCoordinationSituationalSignal

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

6.1.2.1 reportCoordinationSituationalSignal

Description: This operation is used to report the current coordination status of this vessel with respect to situational signals (coordinating with other vehicles).

Namespace: UMAA::MO::CoordinationSituationalSignalStatus

Topic: CoordinationSituationalSignalReportType

Data Type: CoordinationSituationalSignalReportType

Table 11: CoordinationSituationalSignalReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
currentSituation	CoordinationSituationalSignalEnumType	The current coordination status of this vessel with respect to situational signals (coordinating with other vehicles).

6.1.3 GlobalDriftControl

The purpose of this service is to maintain the vehicle's position at a specified elevation (or current elevation if not specified) within a defined drift radius, while in a reduced power mode. See figure for reference. Drift radius and elevation include optional tolerances. If specified, the vehicle is allowed to operate within these tolerances without having the command fail. Otherwise, if any tolerances are violated after the command is initially achieved, then the command is considered to have failed. If a tolerance is not specified for any of these attributes, then it is a "best effort" to maintain the specified value for that attribute, and is therefore not considered a cause for the command to fail.

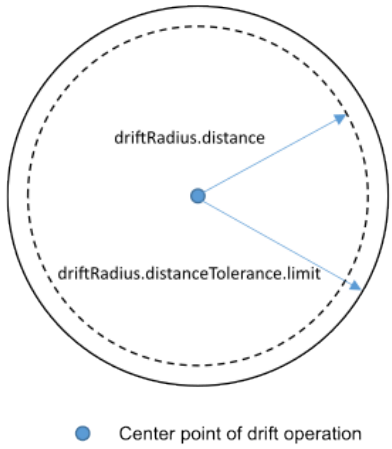

Figure 44: Example Drift Pattern

Table 12: GlobalDriftControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setGlobalDrift	reportGlobalDriftCommandStatus
queryGlobalDriftCommandAck⊕	reportGlobalDriftCommandAck
queryGlobalDriftExecutionStatus⊕	reportGlobalDriftExecutionStatus
cancelGlobalDriftCommand⊕	reportGlobalDriftCancelCommandStatus⊕

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

6.1.3.1 reportGlobalDriftCommandAck

Description: This operation is used to report the commanded values of the position and global drift and/or time that were commanded to the vehicle in the global coordinate system.

Namespace: UMAA::MO::GlobalDriftControl

Topic: GlobalDriftCommandAckReportType

Data Type: GlobalDriftCommandAckReportType

Table 13: GlobalDriftCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
command	GlobalDriftCommandType	The source command.

6.1.3.2 reportGlobalDriftCommandStatus

Description: This operation is used to report the status of the global drift command.

Namespace: UMAA::MO::GlobalDriftControl

Topic: GlobalDriftCommandStatusType

Data Type: GlobalDriftCommandStatusType

Table 14: GlobalDriftCommandStatusType Message Definition

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

6.1.3.3 reportGlobalDriftExecutionStatus

Description: This operation is used to report the current state of the vehicle drift in the global coordinate system.

Namespace: UMAA::MO::GlobalDriftControl

Topic: GlobalDriftExecutionStatusReportType

Data Type: GlobalDriftExecutionStatusReportType

Table 15: GlobalDriftExecutionStatusReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
distanceFromReference	Distance	Defines the distance from the reference position.
globalDriftState	GlobalDriftStateType	Defines the state of the global drift.
timeDriftAchieved	DateTime	Defines the absolute time at which drift is estimated to be achieved or was actually first achieved.
timeDriftCompleted†	DateTime	Defines the absolute time at which the drift is estimated to be completed (optional in case duration is forever).

6.1.3.4 setGlobalDrift

Description: This operation is used to set the desired position in the global coordinate system given the specified global drift and/or time.

Namespace: UMAA::MO::GlobalDriftControl

Topic: GlobalDriftCommandType

Data Type: GlobalDriftCommandType

Table 16: GlobalDriftCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
driftRadius	DistanceRequirementType	Defines the drift radius that specifies the maximum distance from the reference position the vehicle is allowed to drift.
elevation†	ElevationRequirementVariantType	Defines the elevation to maintain when within the driftTolerance of the drift position. If not specified, it is performed at the current elevation.
endTime†	DateTime	Specifies the end of the command execution time period for the drift operation. If not specified, the command runs indefinitely until externally changed.
position	GeoPosition2D	Defines the reference position for drifting.
speed	SpeedVariantType	The desired speed to return to the drift position when the drift tolerance is exceeded.
transitElevation†	ElevationVariantType	The elevation used when driving back to get within the driftTolerance of the drift position. If not specified, it is performed at the current elevation.
transitSpeed	SpeedVariantType	The speed at which one drives to get within the driftTolerance of the drift position.

6.1.4 GlobalHoverControl

The function of this service is to command the vehicle to actively maintain its position at a defined elevation (or current elevation if not defined) within the circle, and optionally with a defined heading. Elevation, heading and hoverRadius include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the command fail. Otherwise if any tolerance is violated after the command is initially achieved, then the command is considered to have failed. If a tolerance is not specified for any of these attributes, then it is a "best effort" to maintain the specified value for that attribute, and is therefore not considered a cause for the command to fail.

Table 17: GlobalHoverControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setGlobalHover	reportGlobalHoverCommandStatus
queryGlobalHoverCommandAck⊕	reportGlobalHoverCommandAck
queryGlobalHoverExecutionStatus⊕	reportGlobalHoverExecutionStatus
cancelGlobalHoverCommand⊕	reportGlobalHoverCancelCommandStatus⊕

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

6.1.4.1 reportGlobalHoverCommandAck

Description: This operation is used to report the commanded values of the position or time that was commanded to the vehicle in the global coordinate system.

Namespace: UMAA::MO::GlobalHoverControl

Topic: GlobalHoverCommandAckReportType

Data Type: GlobalHoverCommandAckReportType

Table 18: GlobalHoverCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
command	GlobalHoverCommandType	The source command.

6.1.4.2 reportGlobalHoverCommandStatus

Description: This operation is used to report the status of the global hover command.

Namespace: UMAA::MO::GlobalHoverControl

Topic: GlobalHoverCommandStatusType

Data Type: GlobalHoverCommandStatusType

Table 19: GlobalHoverCommandStatusType Message Definition

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

6.1.4.3 reportGlobalHoverExecutionStatus

Description: This operation is used to report the current position or time that the vehicle was hovering based in the global coordinate system.

Namespace: UMAA::MO::GlobalHoverControl

Topic: GlobalHoverExecutionStatusReportType

Data Type: GlobalHoverExecutionStatusReportType

Table 20: GlobalHoverExecutionStatusReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
globalHoverState	GlobalHoverStateType	Defines the state of the global hover.
timeHoverAchieved	DateTime	The absolute time at which hover is estimated to be achieved or was actually first achieved.
timeHoverCompleted†	DateTime	The absolute time at which the hover is estimated to be completed (optional in case duration is forever).

6.1.4.4 setGlobalHover

Description: This operation is used to set the desired hover position in the global coordinate system given the desired location.

Namespace: UMAA::MO::GlobalHoverControl

Topic: GlobalHoverCommandType

Data Type: GlobalHoverCommandType

Table 21: GlobalHoverCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
controlPriority	HoverKindEnumType	The desired priority to hover at the specified point.
elevation†	ElevationRequirementVariantType	The elevation used for the vehicle. If not specified, it is performed at the current elevation.
endTime†	DateTime	Specifies the end of the command execution time period for the hover. If not specified, the command runs indefinitely until externally changed.
heading†	DirectionRequirementVariantType	Defines the heading that the vehicle must maintain for hovering. If not specified, the system will determine the best heading (e.g. current heading, into the wind/current, etc.)
hoverRadius	DistanceRequirementType	Defines the hover radius that specifies the maximum distance from the reference position the vehicle is allowed to hover.
position	GeoPosition2D	The desired hover position of the vehicle in the global coordinate system.
transitElevation†	ElevationVariantType	The elevation used while driving to the hover location. If not specified, it is performed at the current elevation.
transitSpeed	SpeedVariantType	The speed at which one drives to the hover location.

6.1.5 GlobalVectorControl

The purpose of this service is to command the vehicle to maintain a provided speed, direction of travel, and altitude or depth (if supported). Direction, elevation, and speed include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the command fail. Otherwise, if any tolerance is violated after the command is initially achieved, then the command is considered to have failed. If a tolerance is not specified for any of these attributes, then it is a "best effort" to maintain the specified value for that attribute, and is therefore not considered a cause for the command to fail.

Table 22: GlobalVectorControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setGlobalVector	reportGlobalVectorCommandStatus
queryGlobalVectorCommandAck⊕	reportGlobalVectorCommandAck
queryGlobalVectorExecutionStatus⊕	reportGlobalVectorExecutionStatus
cancelGlobalVectorCommand⊕	reportGlobalVectorCancelCommandStatus⊕

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

6.1.5.1 reportGlobalVectorCommandAck

Description: This operation is used to report the current commanded values of the speed and depth or altitude to a vehicle in the global coordinate system.

Namespace: UMAA::MO::GlobalVectorControl

Topic: GlobalVectorCommandAckReportType

Data Type: GlobalVectorCommandAckReportType

Table 23: GlobalVectorCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
command	GlobalVectorCommandType	The source command.

6.1.5.2 reportGlobalVectorCommandStatus

Description: This operation is used to report the status of the global vector command.

Namespace: UMAA::MO::GlobalVectorControl

Topic: GlobalVectorCommandStatusType

Data Type: GlobalVectorCommandStatusType

Table 24: GlobalVectorCommandStatusType Message Definition

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

6.1.5.3 reportGlobalVectorExecutionStatus

Description: This operation is used to report the current vector data based in the global coordinate system.

Namespace: UMAA::MO::GlobalVectorControl

Topic: GlobalVectorExecutionStatusReportType

Data Type: GlobalVectorExecutionStatusReportType

Table 25: GlobalVectorExecutionStatusReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
directionAchieved	boolean	When the vector is executing, this indicates if the direction requested is within the commanded tolerances, or system configured tolerance if tolerance is not specified.
elevationAchieved	boolean	When the vector is executing, this indicates if the elevation requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.
speedAchieved	boolean	When the vector is executing, this indicates if the speed requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.

6.1.5.4 setGlobalVector

Description: This operation is used to command the speed and altitude or depth of a vehicle in the global coordinate system. If the command attributes do not specify a determinate end of execution, the consumer must perform a cancel of the command to initiate the end of command execution.

Namespace: UMAA::MO::GlobalVectorControl

Topic: GlobalVectorCommandType

Data Type: GlobalVectorCommandType

Table 26: GlobalVectorCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
depthChangePitch†	PitchYNEDRequirement	The desired angle of the vehicle when traversing to the requested elevation for UUVs.
direction	DirectionRequirementVariantType	The commanded direction of the vehicle motion.
directionMode	DirectionModeEnumType	Specifies the vehicle direction mode.
elevation†	ElevationRequirementVariantType	Specifies the elevation of the vector. If not specified, it is performed at the current elevation.
endTime†	DateTime	Specifies the end of the command execution time period for the vector. If not specified, the command runs indefinitely until externally changed or another terminating condition occurs.
speed	SpeedRequirementVariantType	The desired speed of the vehicle.

6.1.6 GlobalWaypointControl

The purpose of this service is to command the vehicle to traverse through a series of waypoints, each with a desired speed and the option to maintain track. Attitude, elevation, and position include optional tolerances. If specified, these tolerances are used to determine waypoint achievement. If the vehicle is unable to achieve the waypoint, then the command is considered to have failed. Speed and trackTolerance also include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without failing the command. Otherwise, if any tolerances are violated after all specified values are initially achieved, then the command is considered to have failed. If a tolerance is not specified for any of these attributes, then it is a "best effort" to achieve the specified value for that attribute, and is therefore not considered a cause for the command to fail.

Table 27: GlobalWaypointControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setGlobalWaypoint	reportGlobalWaypointCommandStatus
queryGlobalWaypointCommandAck⊕	reportGlobalWaypointCommandAck
queryGlobalWaypointExecutionStatus⊕	reportGlobalWaypointExecutionStatus
cancelGlobalWaypointCommand⊕	reportGlobalWaypointCancelCommandStatus⊕

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

6.1.6.1 reportGlobalWaypointCommandAck

Description: This operation is used to report the commanded values of the waypoint data based in the global coordinate system.

Namespace: UMAA::MO::GlobalWaypointControl

Topic: GlobalWaypointCommandAckReportType

Data Type: GlobalWaypointCommandAckReportType

Table 28: GlobalWaypointCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
command	GlobalWaypointCommandType	The source command.

6.1.6.2 reportGlobalWaypointCommandStatus

Description: This operation is used to report the status of the global waypoint command.

Namespace: UMAA::MO::GlobalWaypointControl

Topic: GlobalWaypointCommandStatusType

Data Type: GlobalWaypointCommandStatusType

Table 29: GlobalWaypointCommandStatusType Message Definition

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

6.1.6.3 reportGlobalWaypointExecutionStatus

Description: This operation is used to report execution status details related to a commanded series of waypoint data based in the global coordinate system.

Namespace: UMAA::MO::GlobalWaypointControl

Topic: GlobalWaypointExecutionStatusReportType

Data Type: GlobalWaypointExecutionStatusReportType

Table 30: GlobalWaypointExecutionStatusReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
arrivalTime	DateTime	The arrival time of the end of the route.

Attribute Name	Attribute Type	Attribute Description
attitudeAchieved†	boolean	When the waypoint is executing, this indicates if the attitude requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified. Optional if attitude is not included in the command.
crossTrackError†	Distance	Defines the current cross track error. This value must be set if trackTolerance is defined and must not be set if trackTolerance is not defined.
cumulativeDistance	Distance	Defines the ground distance travel from the start of the route to this point.
distanceRemaining	Distance	Defines the amount of distance remaining from a point to the end of the route.
distanceToWaypoint	Distance	Defines the remaining distance to the current waypoint.
elevationAchieved	boolean	When the waypoint is executing, this indicates if the elevation requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.
positionAchieved	boolean	When the waypoint is executing, this indicates if the position requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.
speedAchieved	boolean	When the waypoint is executing, this indicates if the speed requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.
timeToWaypoint	DateTime	The absolute time at which the waypoint is estimated to be achieved or was actually first achieved.
trackLineAchieved	boolean	When the waypoint is executing, this indicates if the track line requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.
waypointID	NumericGUID	Defines the current waypoint ID.
waypointsRemaining	Count	Defines the remaining number of waypoints, which includes the current waypoint.

6.1.6.4 setGlobalWaypoint

Description: This operation is used to command a series of waypoint data based in the global coordinate system.

Namespace: UMAA::MO::GlobalWaypointControl

Topic: GlobalWaypointCommandType

Data Type: GlobalWaypointCommandType

Table 31: GlobalWaypointCommandType Message Definition

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

Attribute Name	Attribute Type	Attribute Description
waypoints→listID	LargeList<GlobalWaypoint Type>	The desired series of waypoints in the global coordinate system. This attribute is implemented as a large list, see subsection 3.8 for an explanation. The associated topic is UMAA::MO::GlobalWaypointControl::GlobalWaypointCommandTypeWaypointsListElement.

6.1.7 PrimitiveDriverControl

This service provides mobility in six degrees of freedom using a percent of available effort in each direction. Additionally, no power plant is implied and the service functions strictly in an open loop manner, i.e., a velocity is not commanded or held since that requires a speed sensor. The service definition makes no assertion about the preventative actions that must be taken to avoid unintended consequences, such as losing positive control when given a zero propulsive effort. This service uses "effort" as a relative measure of the amount of drive power. This measure is intentionally kept agnostic of the underlying control system for portability across hardware types. As a result, the implementation of an "effort" driver may map the request to a percent of maximum current of an electric motor, fluid pressure of a hydraulic system, duty-cycle of a pulse-width modulated controller, or position of a control lever. These examples are meant to be illustrative; the actual mapping is not restricted so long as it can be expressed as a percent of some maximum.

Table 32: PrimitiveDriverControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setPrimitiveDriver	reportPrimitiveDriverCommandStatus
queryPrimitiveDriverCommandAck⊕	reportPrimitiveDriverCommandAck
queryPrimitiveDriverExecutionStatus⊕	reportPrimitiveDriverExecutionStatus
cancelPrimitiveDriverCommand⊕	reportPrimitiveDriverCancelCommandStatus⊕

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

6.1.7.1 reportPrimitiveDriverCommandAck

Description: This operation is used to report the current effort command.

Namespace: UMAA::MO::PrimitiveDriverControl

Topic: PrimitiveDriverCommandAckReportType

Data Type: PrimitiveDriverCommandAckReportType

Table 33: PrimitiveDriverCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
command	PrimitiveDriverCommandType	The source command.

6.1.7.2 reportPrimitiveDriverCommandStatus

Description: This operation is used to report the status of the effort command.

Namespace: UMAA::MO::PrimitiveDriverControl

Topic: PrimitiveDriverCommandStatusType

Data Type: PrimitiveDriverCommandStatusType

Table 34: PrimitiveDriverCommandStatusType Message Definition

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

6.1.7.3 reportPrimitiveDriverExecutionStatus

Description: This operation is used to report the current mobility in the vehicle coordinate frame.

Namespace: UMAA::MO::PrimitiveDriverControl

Topic: PrimitiveDriverExecutionStatusReportType

Data Type: PrimitiveDriverExecutionStatusReportType

Table 35: PrimitiveDriverExecutionStatusReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
propulsiveLinearEffort	LinearEffort	The current propulsive linear effort (X, Y, Z).
propulsiveRotationalEffort	RotationalEffort	The current propulsive rotational effort (X, Y, Z).
resistiveLinearEffort	LinearEffort	The current resistive linear effort (X, Y, Z).
resistiveRotationalEffort	RotationalEffort	The current resistive rotational effort (X, Y, Z).

6.1.7.4 setPrimitiveDriver

Description: This operation is used to set the mobility of the vehicle using the effort. The consumer must perform a cancel of the command to initiate the end of command execution as this command has no determinate end of execution.

Namespace: UMAA::MO::PrimitiveDriverControl

Topic: PrimitiveDriverCommandType

Data Type: PrimitiveDriverCommandType

Table 36: PrimitiveDriverCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
propulsiveLinearEffort	LinearEffort	The desired propulsive linear effort (X, Y, Z). Propulsive linear effort represents an action that results in a linear motion along the respective axis.
propulsiveRotationalEffort	RotationalEffort	The desired propulsive rotational effort (X, Y, Z). Propulsive rotational effort represents an action that results in a rotational motion about the respective axis.
resistiveLinearEffort	LinearEffort	The desired resistive linear effort (X, Y, Z). Resistive linear effort represents an action that impedes linear motion along the respective axis.
resistiveRotationalEffort	RotationalEffort	The desired resistive rotational effort (X, Y, Z). Resistive rotational effort represents an action that impedes rotational motion about the respective axis.

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 37: UCSMDEInterfaceSet Structure Definition

Attribute Name	Attribute Type	Attribute Description
timeStamp	DateTime	The origination time of the data being conveyed in the message, or as close to the data or command generation time as is reasonably possible.

6.2.2 UMAACommand

Namespace: UMAA::UMAACommand

Description: Defines the common UMAA Command Message Fields.

Table 38: UMAACommand Structure Definition

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

6.2.3 UMAAStatus

Namespace: UMAA::UMAAStatus

Description: Defines the common UMAA Status Message Fields.

Table 39: UMAAStatus Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UCSMDEInterfaceSet		
source*	IdentifierType	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 40: UMAACommandStatusBase Structure Definition

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

6.2.5 UMAACommandStatus

Namespace: UMAA::UMAACommandStatus

Description: Defines the common UMAA Command Status Message Fields.

Table 41: 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::Common::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 42: 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 AirSpeedRequirement

Namespace: UMAA::Common::Speed::AirSpeedRequirement

Description: Defines the speed through air.

Table 43: AirSpeedRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	IndicatedAirspeed	Specifies speed through air.
speedTolerance†	AirSpeedTolerance	Specifies the tolerance for a speed through air.

6.2.8 AirSpeedRequirementVariantType

Namespace: UMAA::Common::Speed::AirSpeedRequirementVariantType

Description: Defines the speed through air.

Table 44: AirSpeedRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	AirSpeedRequirement	Specifies speed through air.

6.2.9 AirSpeedTolerance

Namespace: UMAA::Common::Speed::AirSpeedTolerance

Description: Defines the speed through air tolerance.

Table 45: AirSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	IndicatedAirspeed	Specifies the lower limit of allowable values for the air speed.
upperlimit	IndicatedAirspeed	Specifies the upper limit of allowable values for the air speed.

6.2.10 AirSpeedVariantType

Namespace: UMAA::Common::Speed::AirSpeedVariantType

Description: Defines the speed through air.

Table 46: AirSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	IndicatedAirspeed	Specifies speed through air.

6.2.11 AltitudeAGLRequirementType

Namespace: UMAA::Common::Measurement::AltitudeAGLRequirementType

Description: Defines the distance above ground level.

Table 47: AltitudeAGLRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	DistanceAGL	Specifies the distance above ground level.
altitudeTolerance†	AltitudeAGLToleranceType	Specifies the tolerance for the distance above ground level.

6.2.12 AltitudeAGLRequirementVariantType

Namespace: UMAA::Common::Measurement::AltitudeAGLRequirementVariantType

Description: The height above ground level.

Table 48: AltitudeAGLRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	AltitudeAGLRequirementType	Specifies the distance above ground level.

6.2.13 AltitudeAGLToleranceType

Namespace: UMAA::Common::Measurement::AltitudeAGLToleranceType

Description: Defines the distance above ground level tolerance.

Table 49: AltitudeAGLToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerLimit	DistanceAGL	Specifies the lower limit of allowable values for the distance above ground level.
upperlimit	DistanceAGL	Specifies the upper limit of allowable values for the distance above ground level.

6.2.14 AltitudeAGLVariantType

Namespace: UMAA::Common::Measurement::AltitudeAGLVariantType

Description: The height above ground level.

Table 50: AltitudeAGLVariantType Structure Definition

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

6.2.15 AltitudeASFRequirementType

Namespace: UMAA::Common::Measurement::AltitudeASFRequirementType

Description: Defines the height above sea floor.

Table 51: AltitudeASFRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	DistanceASF	Specifies the height above sea floor.
altitudeTolerance†	AltitudeASFToleranceType	Specifies the tolerance for the height above sea floor.

6.2.16 AltitudeASFRequirementVariantType

Namespace: UMAA::Common::Measurement::AltitudeASFRequirementVariantType

Description: The height above sea floor.

Table 52: AltitudeASFRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	AltitudeASFRequirementType	The height above the sea floor.

6.2.17 AltitudeASFToleranceType

Namespace: UMAA::Common::Measurement::AltitudeASFToleranceType

Description: Defines the height above sea floor tolerance.

Table 53: AltitudeASFToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerLimit	DistanceASF	Specifies the lower limit of allowable values for the height above sea floor.
upperlimit	DistanceASF	Specifies the upper limit of allowable values for the height above sea floor.

6.2.18 AltitudeASFVariantType

Namespace: UMAA::Common::Measurement::AltitudeASFVariantType

Description: The height above sea floor.

Table 54: AltitudeASFVariantType Structure Definition

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

6.2.19 AltitudeGeodeticRequirementType

Namespace: UMAA::Common::Measurement::AltitudeGeodeticRequirementType

Description: Defines the geodetic height above the ellipsoid.

Table 55: AltitudeGeodeticRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	GeodeticAltitude	Specifies the geodetic height above the ellipsoid.
altitudeTolerance†	AltitudeGeodeticToleranceType	Specifies the tolerance for the geodetic height above the ellipsoid.

6.2.20 AltitudeGeodeticRequirementVariantType

Namespace: UMAA::Common::Measurement::AltitudeGeodeticRequirementVariantType

Description: The geodetic height above the ellipsoid.

Table 56: AltitudeGeodeticRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	AltitudeGeodeticRequirementType	The altitude above the reference ellipsoid.

6.2.21 AltitudeGeodeticToleranceType

Namespace: UMAA::Common::Measurement::AltitudeGeodeticToleranceType

Description: Defines the geodetic height above the ellipsoid tolerance.

Table 57: AltitudeGeodeticToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.

Attribute Name	Attribute Type	Attribute Description
lowerLimit	GeodeticAltitude	Specifies the lower limit of allowable values for the geodetic height above the ellipsoid.
upperlimit	GeodeticAltitude	Specifies the upper limit of allowable values for the geodetic height above the ellipsoid.

6.2.22 AltitudeGeodeticVariantType

Namespace: UMAA::Common::Measurement::AltitudeGeodeticVariantType

Description: The geodetic height above the ellipsoid.

Table 58: AltitudeGeodeticVariantType Structure Definition

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

6.2.23 AltitudeMSLRequirementType

Namespace: UMAA::Common::Measurement::AltitudeMSLRequirementType

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

Table 59: AltitudeMSLRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	MSLAltitude	Specifies the orthometric height above the Geoid (Mean Sea Level).
altitudeTolerance†	AltitudeMSLToleranceType	Specifies the tolerance for the orthometric height above the Geoid (Mean Sea Level).

6.2.24 AltitudeMSLRequirementVariantType

Namespace: UMAA::Common::Measurement::AltitudeMSLRequirementVariantType

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

Table 60: AltitudeMSLRequirementVariantType Structure Definition

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

6.2.25 AltitudeMSLToleranceType

Namespace: UMAA::Common::Measurement::AltitudeMSLToleranceType

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

Table 61: AltitudeMSLToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerLimit	MSLAltitude	Specifies the lower limit of allowable values for the orthometric height above the Geoid (Mean Sea Level).
upperlimit	MSLAltitude	Specifies the upper limit of allowable values for the orthometric height above the Geoid (Mean Sea Level).

6.2.26 AltitudeMSLVariantType

Namespace: UMAA::Common::Measurement::AltitudeMSLVariantType

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

Table 62: AltitudeMSLVariantType Structure Definition

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

6.2.27 AltitudeRateASFRequirementType

Namespace: UMAA::Common::Measurement::AltitudeRateASFRequirementType

Description: Defines the change in altitude as a function of time.

Table 63: AltitudeRateASFRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeRate	SpeedASF	Specifies the change in altitude as a function of time.
altitudeRateTolerance†	AltitudeRateASFToleranceType	Specifies the altitude rate tolerance.

6.2.28 AltitudeRateASFRequirementVariantType

Namespace: UMAA::Common::Measurement::AltitudeRateASFRequirementVariantType

Description: The change in altitude as a function of time.

Table 64: AltitudeRateASFRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitudeRate	AltitudeRateASFRequirementType	Specifies the change in altitude as a function of time.

6.2.29 AltitudeRateASFToleranceType

Namespace: UMAA::Common::Measurement::AltitudeRateASFToleranceType

Description: Defines the altitude rate tolerance.

Table 65: AltitudeRateASFToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerLimit	SpeedASF	Specifies the lower limit of allowable values for the change in altitude as a function of time.
upperlimit	SpeedASF	Specifies the upper limit of allowable values for the change in altitude as a function of time.

6.2.30 DepthRateRequirementType

Namespace: UMAA::Common::Measurement::DepthRateRequirementType

Description: Defines the change in depth as a function of time.

Table 66: DepthRateRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
depthRate	SpeedBSL	Specifies the change in depth as a function of time.
depthRateTolerance†	DepthRateToleranceType	Specifies the depth rate tolerance.

6.2.31 DepthRateRequirementVariantType

Namespace: UMAA::Common::Measurement::DepthRateRequirementVariantType

Description: The change in depth as a function of time.

Table 67: DepthRateRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
depthRate	DepthRateRequirementType	Specifies the change in depth as a function of time.

6.2.32 DepthRateToleranceType

Namespace: UMAA::Common::Measurement::DepthRateToleranceType

Description: Defines the depth rate tolerance.

Table 68: DepthRateToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerLimit	SpeedBSL	Specifies the lower limit of allowable values for the change in depth as a function of time.
upperlimit	SpeedBSL	Specifies the upper limit of allowable values for the change in depth as a function of time.

6.2.33 DepthRequirementType

Namespace: UMAA::Common::Measurement::DepthRequirementType

Description: Defines the depth below sea level.

Table 69: DepthRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
depth	DistanceBSL	Specifies the depth below sea level.
depthTolerance†	DepthToleranceType	Specifies the tolerance for the depth below sea level.

6.2.34 DepthRequirementVariantType

Namespace: UMAA::Common::Measurement::DepthRequirementVariantType

Description: The depth below sea level.

Table 70: DepthRequirementVariantType Structure Definition

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

6.2.35 DepthToleranceType

Namespace: UMAA::Common::Measurement::DepthToleranceType

Description: Defines the depth below sea level tolerance.

Table 71: DepthToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerLimit	DistanceBSL	Specifies the lower limit of allowable values for the depth below sea level.

Attribute Name	Attribute Type	Attribute Description
upperlimit	DistanceBSL	Specifies the upper limit of allowable values for the depth below sea level.

6.2.36 DepthVariantType

Namespace: UMAA::Common::Measurement::DepthVariantType

Description: The depth below sea level.

Table 72: DepthVariantType Structure Definition

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

6.2.37 DirectionCurrentRequirement

Namespace: UMAA::Common::Orientation::DirectionCurrentRequirement

Description: A requirement that specifies the direction with respect to the current, where 0 is defined to be with the direction of the current (i.e. downstream).

Table 73: DirectionCurrentRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	HeadingCurrentDirection	Specifies the heading offset angle relative to the current, where 0 is defined to be with the direction of the current (i.e. downstream).
directionTolerance†	DirectionToleranceType	Specifies the heading reference angle tolerance relative to the current, where 0 is defined to be with the direction of the current (i.e. downstream).

6.2.38 DirectionCurrentRequirementVariantType

Namespace: UMAA::Common::Orientation::DirectionCurrentRequirementVariantType

Description: Specifies the direction with respect to the current, where 0 is defined to be with the direction of the current (i.e. downstream).

Table 74: DirectionCurrentRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	DirectionCurrentRequirement	Specifies the heading offset angle relative to the current, where 0 is defined to be with the direction of the current (i.e. downstream).

6.2.39 DirectionMagneticNorthRequirement

Namespace: UMAA::Common::Orientation::DirectionMagneticNorthRequirement

Description: A requirement that specifies the direction with respect to magnetic north.

Table 75: DirectionMagneticNorthRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	HeadingMagneticNorth	Specifies the heading reference angle relative to magnetic north.
directionTolerance†	DirectionToleranceType	Specifies the heading reference angle tolerance relative to magnetic north.

6.2.40 DirectionMagneticNorthRequirementVariantType

Namespace: UMAA::Common::Orientation::DirectionMagneticNorthRequirementVariantType

Description: Specifies the direction with respect to magnetic north.

Table 76: DirectionMagneticNorthRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	DirectionMagneticNorthRequirement	Specifies the heading reference angle relative to magnetic north.

6.2.41 DirectionRequirementVariantType

Namespace: UMAA::Common::Orientation::DirectionRequirementVariantType

Description: **Union Type**. Direction of the vehicle motion or pattern being performed.

Table 77: DirectionRequirementVariantType Union(s)

Type Name	Type Description
DirectionCurrentRequirementVariantType	Specifies the direction with respect to the current, where 0 is defined to be with the direction of the current (i.e. downstream).
DirectionMagneticNorthRequirementVariantType	Specifies the direction with respect to magnetic north.
DirectionTrueNorthRequirementVariantType	Specifies the direction with respect to true north.
DirectionTurnRateRequirementVariantType	Specifies the change in direction as a function of time.
DirectionWindRequirementVariantType	Specifies the direction with respect to the direction of the wind, where 0 is defined to be the direction into the wind.

6.2.42 DirectionToleranceType

Namespace: UMAA::Common::Orientation::DirectionToleranceType

Description: An angle tolerance associated with a direction.

Table 78: DirectionToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	Angle	Describes the direction bound counterclockwise from the specified direction.
upperlimit	Angle	Describes the direction bound clockwise from the specified direction.

6.2.43 DirectionTrueNorthRequirement

Namespace: UMAA::Common::Orientation::DirectionTrueNorthRequirement

Description: A requirement that specifies the direction with respect to true north.

Table 79: DirectionTrueNorthRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	HeadingTrueNorthAngle	Specifies the heading reference angle relative to true north.
directionTolerance†	DirectionToleranceType	Specifies the heading reference angle tolerance relative to true north.

6.2.44 DirectionTrueNorthRequirementVariantType

Namespace: UMAA::Common::Orientation::DirectionTrueNorthRequirementVariantType

Description: Specifies the direction with respect to true north.

Table 80: DirectionTrueNorthRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	DirectionTrueNorthRequirement	Specifies the heading reference angle relative to true north.

6.2.45 DirectionTurnRateRequirementType

Namespace: UMAA::Common::Orientation::DirectionTurnRateRequirementType

Description: A requirement that specifies the change in direction of the vehicle's motion as a function of time.

Table 81: DirectionTurnRateRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
directionRate	TurnRate	Specifies a change in direction as a function of time.
directionRateTolerance†	DirectionTurnRateToleranceType	Specifies the direction turn rate tolerance.

6.2.46 DirectionTurnRateRequirementVariantType

Namespace: UMAA::Common::Orientation::DirectionTurnRateRequirementVariantType

Description: Specifies the change in direction as a function of time.

Table 82: DirectionTurnRateRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
directionRate	DirectionTurnRateRequirementType	Specifies the change in direction of the vehicle's motion as a function of time.

6.2.47 DirectionTurnRateToleranceType

Namespace: UMAA::Common::Orientation::DirectionTurnRateToleranceType

Description: Defines the direction turn rate tolerance.

Table 83: DirectionTurnRateToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	TurnRate	Specifies the lower limit of allowable values for the change in direction as a function of time.
upperlimit	TurnRate	Specifies the upper limit of allowable values for the change in direction as a function of time.

6.2.48 DirectionWindRequirement

Namespace: UMAA::Common::Orientation::DirectionWindRequirement

Description: A requirement that specifies the direction with respect to the direction of the wind, where 0 is defined to be the direction into the wind

Table 84: DirectionWindRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	HeadingWindDirection	Specifies the heading offset angle relative to the wind, where 0 is defined to be the direction into the wind.
directionTolerance†	DirectionToleranceType	Specifies the heading reference angle tolerance relative to the wind direction, where 0 is defined to be the direction into the wind.

6.2.49 DirectionWindRequirementVariantType

Namespace: UMAA::Common::Orientation::DirectionWindRequirementVariantType

Description: Specifies the direction with respect to the direction of the wind, where 0 is defined to be the direction into the wind.

Table 85: DirectionWindRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	DirectionWindRequirement	Specifies the heading offset angle relative to the wind, where 0 is defined to be the direction into the wind.

6.2.50 DistanceRequirementType

Namespace: UMAA::Common::Distance::DistanceRequirementType

Description: Defines a distance requirement.

Table 86: DistanceRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
distance	Distance	Specifies the required distance.
distanceTolerance†	DistanceToleranceType	Specifies the distance tolerance.

6.2.51 DistanceToleranceType

Namespace: UMAA::Common::Distance::DistanceToleranceType

Description: Defines the distance tolerance.

Table 87: DistanceToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
limit	Distance	Specifies the limit of the tolerance.

6.2.52 ElevationRequirementVariantType

Namespace: UMAA::Common::Measurement::ElevationRequirementVariantType

Description: Union Type. The desired elevation used for the vehicle.

Table 88: ElevationRequirementVariantType Union(s)

Type Name	Type Description
AltitudeAGLRequirementVariantType	The height above ground level.
AltitudeASFRequirementVariantType	The height above sea floor.
AltitudeGeodeticRequirementVariantType	The geodetic height above the ellipsoid.
AltitudeMSLRequirementVariantType	The orthometric height above the Geoid (Mean Sea Level).
AltitudeRateASFRequirementVariantType	The change in altitude as a function of time.
DepthRateRequirementVariantType	The change in depth as a function of time.
DepthRequirementVariantType	The depth below sea level.

6.2.53 ElevationVariantType

Namespace: UMAA::Common::Measurement::ElevationVariantType

Description: Union Type. The desired elevation used for the vehicle.

Table 89: ElevationVariantType Union(s)

Type Name	Type Description
AltitudeAGLVariantType	The height above ground level.
AltitudeASFVariantType	The height above sea floor.
AltitudeGeodeticVariantType	The geodetic height above the ellipsoid.
AltitudeMSLVariantType	The orthometric height above the Geoid (Mean Sea Level).
DepthVariantType	The depth below sea level.

6.2.54 EngineRPMSpeedRequirement

Namespace: UMAA::Common::Speed::EngineRPMSpeedRequirement

Description: Defines the engine rpm.

Table 90: EngineRPMSpeedRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	FrequencyRPM	Specifies speed via engine rpm.
speedTolerance†	EngineRPMSpeedTolerance	Specifies the tolerance for an engine rpm.

6.2.55 EngineRPMSpeedRequirementVariantType

Namespace: UMAA::Common::Speed::EngineRPMSpeedRequirementVariantType

Description: Defines the engine RPM.

Table 91: EngineRPMSpeedRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
rpm	EngineRPMSpeedRequirement	Specifies engine rpm.

6.2.56 EngineRPMSpeedTolerance

Namespace: UMAA::Common::Speed::EngineRPMSpeedTolerance

Description: Defines the speed through engine rpm.

Table 92: EngineRPMSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	FrequencyRPM	Specifies the lower limit of allowable values for the engine rpm.
upperlimit	FrequencyRPM	Specifies the upper limit of allowable values for the engine rpm.

6.2.57 EngineRPMSpeedVariantType

Namespace: UMAA::Common::Speed::EngineRPMSpeedVariantType

Description: Defines the engine RPM.

Table 93: EngineRPMSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
rpm	FrequencyRPM	Specifies speed via engine rpm.

6.2.58 GeoPosition2D

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

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

Table 94: 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.

6.2.59 GeoPosition2DRequirement

Namespace: UMAA::Common::Position::GeoPosition2DRequirement

Description: Defines a position requirement.

Table 95: GeoPosition2DRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
tolerance†	GeoPosition2DTolerance	Specifies the required position tolerance.
value	GeoPosition2D	Specifies the required position.

6.2.60 GeoPosition2DTolerance

Namespace: UMAA::Common::Position::GeoPosition2DTolerance

Description: Defines a position tolerance.

Table 96: GeoPosition2DTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
limit	Distance	Specifies the limit of the tolerance.

6.2.61 GlobalDriftStateType

Namespace: UMAA::MO::GlobalDriftState::GlobalDriftStateType

Description: Union Type. State of the global drift. While first transiting to the drift position, the selector will be GlobalTransitDriftType until the position and elevation are first achieved within their respective tolerances. Once achieved, the union selector will change to GlobalRegionDriftType. The selector will not change as a result of any of the GlobalRegionDriftType achievements states being lost and regained as a result of tolerance settings being violated. This is true until the service determines that the elevation or drift tolerances are violated by a sufficient margin that it is more effective for the vehicle to return to transiting to the drift location. In that case, the GlobalRegionDriftType reverts to the GlobalTransitDriftType selector and those transit achievements then are actively set.

Table 97: GlobalDriftStateType Union(s)

Type Name	Type Description
GlobalRegionDriftType	Indicates that the vehicle is in the global drift region.
GlobalTransitDriftType	Indicates that vehicle is in transit to the global drift region.

6.2.62 GlobalHoverStateType

Namespace: UMAA::MO::GlobalHoverState::GlobalHoverStateType

Description: Union Type. State of the global hover. While first transiting to the hover location, the selector will be GlobalTransitHoverType until the position, heading, and elevation are first achieved within their respective tolerances. Once achieved, the union selector will change to GlobalHoveringHoverType. The selector will not change as a result of any of the GlobalHoveringHoverType achievements states being lost and regained as a result of tolerance settings being violated. The service is expected to make driving adjustments to attempt to keep all achievement states satisfied. This is true until the service determines that tolerance(s) are violated by a sufficient margin that it is more effective for the vehicle to return to transiting to the hover location. In that case, the GlobalHoverStateType reverts to the GlobalTransitHoverType selector and those transit achievements then are actively set.

Table 98: GlobalHoverStateType Union(s)

Type Name	Type Description
GlobalHoveringHoverType	Indicates that the global hover is currently executing.

Type Name	Type Description
GlobalTransitHoverType	Indicates that the vehicle is in transit to where the global hover is to be performed.

6.2.63 GlobalHoveringHoverType

Namespace: UMAA::MO::GlobalHoverState::GlobalHoveringHoverType

Description: Indicates that the global hover is currently executing.

Table 99: GlobalHoveringHoverType Structure Definition

Attribute Name	Attribute Type	Attribute Description
elevationAchieved	boolean	Indicates if the elevation requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.
headingAchieved†	boolean	Indicates if the heading requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified. This attribute is not provided if the command does not specify a desired heading.
hoverRadiusAchieved	boolean	Indicates if the vehicle is within the commanded hover radius.

6.2.64 GlobalRegionDriftType

Namespace: UMAA::MO::GlobalDriftState::GlobalRegionDriftType

Description: Indicates that the vehicle is in the global drift region.

Table 100: GlobalRegionDriftType Structure Definition

Attribute Name	Attribute Type	Attribute Description
driftRadiusAchieved	boolean	Indicates if the vehicle is within the commanded drift radius.
elevationAchieved	boolean	Indicates if the elevation requested is within the commanded tolerance, or system configured tolerance if tolerance is not specified.

6.2.65 GlobalTransitDriftType

Namespace: UMAA::MO::GlobalDriftState::GlobalTransitDriftType

Description: Indicates that vehicle is in transit to the global drift region.

Table 101: GlobalTransitDriftType Structure Definition

Attribute Name	Attribute Type	Attribute Description
elevationAchieved	boolean	Indicates if the transit elevation requested is within the system configured tolerances.
speedAchieved	boolean	Indicates if the transit speed requested is within the system configured tolerances.

6.2.66 GlobalTransitHoverType

Namespace: UMAA::MO::GlobalHoverState::GlobalTransitHoverType

Description: Indicates that the vehicle is in transit to where the global hover is to be performed.

Table 102: GlobalTransitHoverType Structure Definition

Attribute Name	Attribute Type	Attribute Description
elevationAchieved	boolean	Indicates if the transit elevation requested is within the system configured tolerances.
speedAchieved	boolean	Indicates if the transit speed requested is within the system configured tolerances.

6.2.67 GlobalWaypointType

Namespace: UMAA::MO::GlobalWaypointControl::GlobalWaypointType

Description: The structure is used to describe a waypoint in a global reference frame.

Table 103: GlobalWaypointType Structure Definition

Attribute Name	Attribute Type	Attribute Description
attitude†	Orientation3DNEDRequirement	The desired orientation (roll, pitch, yaw) of the vehicle as arriving at the waypoint.
elevation†	ElevationRequirementVariantType	The target elevation used for the vehicle when reaching the waypoint. If not provided, any elevation at the waypoint is acceptable. Changes in elevation will be achieved as quickly as possible on the way to the waypoint.
name†	StringShortDescription	A short name for the waypoint.
position	GeoPosition2DRequirement	The desired waypoint position in the global coordinate system.

Attribute Name	Attribute Type	Attribute Description
speed	VariableSpeedVariantType	The desired transit speed to the waypoint of the vehicle with reference to the medium, the ground, the air, RPM, or true speed.
trackTolerance†	DistanceRequirementType	The desired tolerance of the path measured by distance from the waypoint position ignoring elevation. If defined, the vehicle must maintain track; if not defined, there is no need to maintain track. Use the vehicle position at time of command to define the track for the first waypoint.
waypointID	NumericGUID	The desired id to keep track of the waypoint.

6.2.68 GroundSpeedRequirement

Namespace: UMAA::Common::Speed::GroundSpeedRequirement

Description: Defines the speed over ground.

Table 104: GroundSpeedRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	GroundSpeed	Specifies speed over ground.
speedTolerance†	GroundSpeedTolerance	Specifies the tolerance for a speed over ground.

6.2.69 GroundSpeedRequirementVariantType

Namespace: UMAA::Common::Speed::GroundSpeedRequirementVariantType

Description: Defines the speed over ground.

Table 105: GroundSpeedRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	GroundSpeedRequirement	Specifies speed over ground.

6.2.70 GroundSpeedTolerance

Namespace: UMAA::Common::Speed::GroundSpeedTolerance

Description: Defines the speed over ground tolerance.

Table 106: GroundSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	GroundSpeed	Specifies the lower limit of allowable values for the ground speed.
upperlimit	GroundSpeed	Specifies the upper limit of allowable values for the ground speed.

6.2.71 GroundSpeedVariantType

Namespace: UMAA::Common::Speed::GroundSpeedVariantType

Description: Defines the speed over ground.

Table 107: GroundSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	GroundSpeed	Specifies speed over ground.

6.2.72 IdentifierType

Namespace: UMAA::Common::IdentifierType

Description: This structure defines a two-level hierarchical identifier, where the parent is defined to be a group or collection of entities.

Table 108: IdentifierType Structure Definition

Attribute Name	Attribute Type	Attribute Description
id	NumericGUID	Provides the identifier of an entity.
parentID	NumericGUID	Provides the identifier of the parent, which is a group or collection of one or more entities. If the entity has no parent (it is the root of the tree), this value will be the Nil UUID.

6.2.73 LinearEffort

Namespace: UMAA::Common::Measurement::LinearEffort

Description: Defines the along-axes efforts.

Table 109: LinearEffort Structure Definition

Attribute Name	Attribute Type	Attribute Description
xAxis	Effort	Linear effort along the x-axis.
yAxis	Effort	Linear effort along the y-axis.
zAxis	Effort	Linear effort along the z-axis.

6.2.74 Orientation3DNEDRequirement

Namespace: UMAA::Common::Orientation::Orientation3DNEDRequirement

Description: A requirement that describes a desired 3D orientation in a NED coordinate system.

Table 110: Orientation3DNEDRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitchY†	PitchYNEDRequirement	Defines a pitch relative to the NED coordinate system.
rollX†	RollXNEDRequirement	Defines a roll relative to the NED coordinate system.
yawZ	YawZNEDRequirement	Defines a yaw relative to the NED coordinate system.

6.2.75 PitchYNEDRequirement

Namespace: UMAA::Common::Orientation::PitchYNEDRequirement

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

Table 111: PitchYNEDRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitch	PitchYNEDType	Defines a pitch relative to the NED system.
pitchTolerance†	PitchYNEDTolerance	Describes the pitch bounding limits.

6.2.76 PitchYNEDTolerance

Namespace: UMAA::Common::Orientation::PitchYNEDTolerance

Description: A down or up angle tolerance.

Table 112: PitchYNEDTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	PitchYNEDType	Defines the steepest downangle allowed.
upperlimit	PitchYNEDType	Defines the steepest upangle allowed.

6.2.77 PitchYNEDType

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

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

Table 113: PitchYNEDType Structure Definition

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

6.2.78 RecommendedSpeedVariantType

Namespace: UMAA::Common::Speed::RecommendedSpeedVariantType

Description: Defines the recommended speed mode.

Table 114: RecommendedSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	SpeedVariantType	Specifies the recommended speed mode.

6.2.79 RequiredSpeedVariantType

Namespace: UMAA::Common::Speed::RequiredSpeedVariantType

Description: Defines the required speed mode.

Table 115: RequiredSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	SpeedRequirementVariantType	Specifies the required speed mode.

6.2.80 RollXNEDRequirement

Namespace: UMAA::Common::Orientation::RollXNEDRequirement

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

Table 116: RollXNEDRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
roll	RollXNEDType	Defines a roll relative to the NED system.
rollTolerance†	RollXNEDTolerance	Describes the roll bounding limits.

6.2.81 RollXNEDTolerance

Namespace: UMAA::Common::Orientation::RollXNEDTolerance

Description: A rotational tolerance.

Table 117: RollXNEDTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	RollXNEDType	Defines the lower bound.
upperlimit	RollXNEDType	Defines the upper bound.

6.2.82 RollXNEDType

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

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

Table 118: RollXNEDType Structure Definition

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

6.2.83 RotationalEffort

Namespace: UMAA::Common::Measurement::RotationalEffort

Description: Describes a set of efforts around each axis, using the right-hand rule.

Table 119: RotationalEffort Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitchEffort	Effort	Rotational effort around the y-axis.
rollEffort	Effort	Rotational effort around the x-axis.
yawEffort	Effort	Rotational effort around the z-axis.

6.2.84 SpeedRequirementVariantType

Namespace: UMAA::Common::Speed::SpeedRequirementVariantType

Description: Union Type. Speed of the vehicle.

Table 120: SpeedRequirementVariantType Union(s)

Type Name	Type Description
AirSpeedRequirementVariantType	Defines the speed through air.
EngineRPMSpeedRequirementVariantType	Defines the engine RPM.
GroundSpeedRequirementVariantType	Defines the speed over ground.
VehicleSpeedModeRequirementVariantType	Defines the speed mode.
WaterSpeedRequirementVariantType	Defines the speed through water.

6.2.85 SpeedVariantType

Namespace: UMAA::Common::Speed::SpeedVariantType

Description: Union Type. Speed of the vehicle.

Table 121: SpeedVariantType Union(s)

Type Name	Type Description
AirSpeedVariantType	Defines the speed through air.
EngineRPMSpeedVariantType	Defines the engine RPM.
GroundSpeedVariantType	Defines the speed over ground.
VehicleSpeedModeVariantType	Defines the speed mode.
WaterSpeedVariantType	Defines the speed through water.

6.2.86 TimeWithSpeedVariantType

Namespace: UMAA::Common::Speed::TimeWithSpeedVariantType

Description: Defines the time window and the recommended speed of a vehicle.

Table 122: TimeWithSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
arrivalTime	DateTime	Specifies the arrival time of the vehicle.
speed†	SpeedVariantType	Specifies the recommended speed of the vehicle.

6.2.87 VariableSpeedVariantType

Namespace: UMAA::Common::Speed::VariableSpeedVariantType

Description: Union Type. Speed specifier for the vehicle which may be based on explicit speed, a recommended speed, a time window, or a time window with an optional recommended speed.

Table 123: VariableSpeedVariantType Union(s)

Type Name	Type Description
RecommendedSpeedVariantType	Defines the recommended speed mode.
RequiredSpeedVariantType	Defines the required speed mode.
TimeWithSpeedVariantType	Defines the time window and the recommended speed of a vehicle.

6.2.88 VehicleSpeedModeRequirementVariantType

Namespace: UMAA::Common::Speed::VehicleSpeedModeRequirementVariantType

Description: Defines the speed mode.

Table 124: VehicleSpeedModeRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
mode	VehicleSpeedModeEnumType	Specifies the speed mode.

6.2.89 VehicleSpeedModeVariantType

Namespace: UMAA::Common::Speed::VehicleSpeedModeVariantType

Description: Defines the speed mode.

Table 125: VehicleSpeedModeVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
mode	VehicleSpeedModeEnumType	Specifies the speed mode.

6.2.90 WaterSpeedRequirement

Namespace: UMAA::Common::Speed::WaterSpeedRequirement

Description: Defines the speed through water.

Table 126: WaterSpeedRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	SpeedLocalWaterMass	Specifies speed through water.
speedTolerance†	WaterSpeedTolerance	Specifies the tolerance for a speed through water.

6.2.91 WaterSpeedRequirementVariantType

Namespace: UMAA::Common::Speed::WaterSpeedRequirementVariantType

Description: Defines the speed through water.

Table 127: WaterSpeedRequirementVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	WaterSpeedRequirement	Specifies speed through water.

6.2.92 WaterSpeedTolerance

Namespace: UMAA::Common::Speed::WaterSpeedTolerance

Description: Defines the speed through water tolerance.

Table 128: WaterSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	SpeedLocalWaterMass	Specifies the lower limit of allowable values for the water speed.
upperlimit	SpeedLocalWaterMass	Specifies the upper limit of allowable values for the water speed.

6.2.93 WaterSpeedVariantType

Namespace: UMAA::Common::Speed::WaterSpeedVariantType

Description: Defines the speed through water.

Table 129: WaterSpeedVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
speed	SpeedLocalWaterMass	Specifies speed through water.

6.2.94 YawZNEDRequirement

Namespace: UMAA::Common::Orientation::YawZNEDRequirement

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

Table 130: YawZNEDRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
yaw	YawZNEDType	Defines a yaw relative to the NED system.
yawTolerance†	YawZNEDTolerance	Describes the yaw bounding limits.

6.2.95 YawZNEDTolerance

Namespace: UMAA::Common::Orientation::YawZNEDTolerance

Description: A directional tolerance.

Table 131: YawZNEDTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
failureDelay†	DurationSeconds	After the value with a specified tolerance is achieved (either initially or after an update), defines the amount of time to delay transitioning a command or objective to a failed state when the specified tolerance is violated. This measured time is reset each time the value is determined to be within the specified tolerance. If not defined, then a system configured delay time is used.
lowerlimit	YawZNEDType	Defines the lower bound.
upperlimit	YawZNEDType	Defines the upper bound.

6.2.96 YawZNEDType

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

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

Table 132: YawZNEDType Structure Definition

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

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 CommandStatusReasonEnumType

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

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

Table 133: 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.
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.

6.3.2 ContactManeuverInfluenceEnumType

Namespace: UMAA::Common::MaritimeEnumeration::ContactManeuverInfluenceEnumType

Description: A mutually exclusive set of values that defines the influence a contact may have on the maneuvering of ownship.

Table 134: ContactManeuverInfluenceEnumType Enumeration

Enumeration Value	Description
COLLISION	Maneuvering or not maneuvering where a determination that a collision, allision, or near-miss has occurred or is imminent.
COLLISION_AVOIDANCE	Maneuvering to avoid a navigational hazard where the priority is to increase separation to avoid a likely collision or allision if no action is taken. Maneuvering requires no due regard to other influences, including mission objectives.
CROSSING_LEFT_COMPLIANT	COLREGS crossing left where the other vehicle is determined to be compliant.

Enumeration Value	Description
CROSSING_LEFT_NONCOMPLIANT	COLREGS crossing left where the other vehicle is determined to be non-compliant.
CROSSING_RIGHT_COMPLIANT	COLREGS crossing right where the other vehicle is determined to be compliant.
CROSSING_RIGHT_NONCOMPLIANT	COLREGS crossing right where the other vehicle is determined to be non-compliant.
DYNAMIC_AVOIDANCE	Maneuvering to avoid a dynamic contact where there is no obstacle avoidance influence other than maintaining a prescribed minimum standoff distance.
GUIDE	Contact is guiding or informing maneuvering (e.g., guide vehicle for Stationkeep or cooperating swarm member).
HEAD_ON_COMPLIANT	COLREGS head on where the other vehicle is determined to be compliant.
HEAD_ON_NONCOMPLIANT	COLREGS head on where the other vehicle is determined to be non-compliant.
NONE	The contact has been examined and it was determined it has no influence on the maneuvering of ownship.
OVERTAKEN_COMPLIANT	COLREGS being overtaken where the other vehicle is determined to be compliant.
OVERTAKEN_NONCOMPLIANT	COLREGS being overtaken where the other vehicle is determined to be non-compliant.
OVERTAKING_COMPLIANT	COLREGS overtaking where the other vehicle is determined to be compliant.
OVERTAKING_NONCOMPLIANT	COLREGS overtaking where the other vehicle is determined to be non-compliant.
PREEMPTIVE	Maneuvering to avoid a perceived future state but not in direct response to configured obstacle avoidance thresholds.
STATIC_AVOIDANCE	Maneuvering to avoid a static contact where there is no obstacle avoidance influence other than maintaining a prescribed minimum standoff distance.

6.3.3 CoordinationSituationalSignalEnumType

Namespace: UMAA::Common::MaritimeEnumeration::CoordinationSituationalSignalEnumType

Description: Defines a mutually exclusive set of values that define the current coordination status of this vessel with respect to situational signals (coordinating with other vehicles).

Table 135: CoordinationSituationalSignalEnumType Enumeration

Enumeration Value	Description
AGREE_TO_BE_OVERTAKEN	Vehicle is being overtaken.
ALTERING_COURSE_TO_PORT	Vehicle is turning to port.
ALTERING_COURSE_TO_STARBOARD	Vehicle is turning to starboard.
BLIND_BEND_SIGNAL	Vehicle is maneuvering around a blind bend.
DANGER_SIGNAL	Vehicle is in danger.
IN_DISTRESS_NEED_ASSISTANCE	Vehicle is in distress and needs assistance.
NONE	No signal active.
OPERATING_ASTERN_PROPULSION	Vehicle is operating astern propulsion.

Enumeration Value	Description
TO_OVERTAKE_LEAVE_VESSEL_TO_PORT	Vehicle is maneuvering to overtake, leave vessel to port.
TO_OVERTAKE_LEAVE_VESSEL_TO_STARBOARD	Vehicle is maneuvering to overtake, leave vessel to starboard.
VESSEL_LEAVING_DOCK	Vehicle is leaving the dock.
VISIBILITY_RESTRICTED_VEHICLE_STOPPED	Visibility restricted with vehicle stopped.
VISIBILITY_RESTRICTED_VEHICLE_UNDERWAY	Visibility restricted with vehicle underway.

6.3.4 DirectionModeEnumType

Namespace: UMAA::Common::MaritimeEnumeration::DirectionModeEnumType

Description: Specifies whether direction is a course or heading.

Table 136: DirectionModeEnumType Enumeration

Enumeration Value	Description
COURSE	Specifies that direction is the course of the vehicle, which is the direction of motion of the vehicle over the ground.
HEADING	Specifies that direction is the heading of the vehicle, which is the direction in which the vehicle's bow is pointing.

6.3.5 HoverKindEnumType

Namespace: UMAA::Common::MaritimeEnumeration::HoverKindEnumType

Description: A mutually exclusive set of values that defines the hover priority of the vehicle.

Table 137: HoverKindEnumType Enumeration

Enumeration Value	Description
LAT_LON_PRIORITY	Prioritize maintaining a latitude/longitude position
Z_PRIORITY	Prioritize maintaining an elevation

6.3.6 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 138: 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.
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.

6.3.7 VehicleSpeedModeEnumType

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

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

Table 139: 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

6.4 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 140: Type Definitions

Type Name	Primitive Type	Range of Values	Description
Angle	double	maxInclusive=3.141592653589 7932 minInclusive=-3.141592653589 7932 units=Radian referenceFrame=Counting	Specifies the amount of turning necessary to bring one ray, line or plane into coincidence with or parallel to another. The measurement is stated in radians between -pi and pi.
BooleanEnumType	boolean		A mutually exclusive set of values that defines the truth values of logical algebra.
Count	long	referenceFrame=Counting minInclusive=-2147483648 maxInclusive=2147483647	Represents a whole (non-fractional) number that can be positive, negative or zero.
DateTimeNanoseconds	long	units=Nanoseconds minInclusive=0 maxInclusive=999999999	The number of nanoseconds elapsed within the current second.
DateTimeSeconds	longlong	units=Seconds minInclusive=-92233720368547 75807 maxInclusive=92233720368547 75807	The seconds offset from the standard POSIX (IEEE Std 1003.1-2017) epoch reference point of January 1st, 1970 00:00:00 UTC.
Distance	double	maxInclusive=401056000 minInclusive=0 units=Meter referenceFrame=Counting	This type stores a distance in meters.
DistanceAGL	double	minInclusive=0.0 units=Meter referenceFrame=AGL	Describes the height above ground level of the vehicle.
DistanceASF	double	maxInclusive=401056000 minInclusive=0 units=Meter referenceFrame=ASF	The altitude or distance above the sea floor in meters.
DistanceBSL	double	maxInclusive=10000 minInclusive=0 units=Meter referenceFrame=BSL	The distance below sea level in meters.
DurationSeconds	double	maxInclusive=37817280 minInclusive=0 units=Seconds referenceFrame=Counting	Represents a time duration in seconds.
Effort	double	maxInclusive=100 minInclusive=-100 units=Percent referenceFrame=PlatformXYZ	Represents the level of effort measured in percent.

Type Name	Primitive Type	Range of Values	Description
FrequencyRPM	long	maxInclusive=100000 minInclusive=-100000 units=RevolutionsPerMinute referenceFrame=Counting	This type stores number of occurrences in revolutions per minute (RPM). Negative number is used for reverse RPM.
GeodeticAltitude	double	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.
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	maxInclusive=299792458 minInclusive=-299792458 units=MeterPerSecond referenceFrame=Ground	The magnitude of the horizontal velocity vector of a vehicle relative to the ground.
HeadingCurrentDirection	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 units=Radian referenceFrame=CurrentDirection	Describes heading with respect to the current direction.
HeadingMagneticNorth	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 units=Radian referenceFrame=MagneticNorth	Heading as an angle specified with respect to Magnetic North.
HeadingTrueNorthAngle	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 units=Radian referenceFrame=TrueNorth	Describes heading with respect to True North.
HeadingWindDirection	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 units=Radian referenceFrame=WindDirection	Describes heading with respect to the wind direction.

Type Name	Primitive Type	Range of Values	Description
IndicatedAirspeed	double	maxInclusive=299792458 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	maxInclusive=2147483647 minInclusive=0	Specifies the size of a Large Collection.
MSLAltitude	double	minInclusive=0.0 units=Meter referenceFrame=Altitude	Describes the current orthometric height above the Geoid (Mean Sea Level).
NumericGUID	octet[16]	minInclusive=0 maxInclusive=(2^128)-1	Represents a 128-bit number according to RFC 4122 variant 2.
PitchHalfAngle	double	maxInclusive=1.5707963267948966 minInclusive=-1.5707963267948966 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.
RollAngle	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 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.
SpeedASF	double	maxInclusive=299792458 minInclusive=-299792458 units=MeterPerSecond referenceFrame=ASF	This type stores speed in meters/s in an above sea floor reference frame.
SpeedBSL	double	maxInclusive=299792458 minInclusive=-299792458 units=MeterPerSecond referenceFrame=BSL	This type stores speed in meters/s in a below sea level reference frame.
SpeedLocalWaterMass	double	maxInclusive=299792458 minInclusive=0 units=MeterPerSecond referenceFrame=LocalWaterMass	This type stores speed in meters/s.
StringLongDescription	string	length=4095	Represents a long format description.
StringShortDescription	string	length=1023	Represents a short format description.

Type Name	Primitive Type	Range of Values	Description
TurnRate	double	maxInclusive=32.767 minInclusive=-32.767 units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the heading angle of a platform.
YawAngle	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 referenceFrame=PlatformNED units=Radian	The yaw angle relative to the NED coordinate system 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
ROC	Risk of Collision
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