Unmanned Maritime Autonomy Architecture (UMAA)
Mission Management (MM)
Interface Control Document (ICD)
(UMAA-SPEC-MMICD)

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). As such, its focus is on services that support performing the overall mission and managing the unmanned vehicle in its operating environment. These services would support mission planning/re-planning, mission execution, managing collaboration with other unmanned and manned vehicles, assessing mission performance, and providing overall control and decision-making for the mission. The services and their corresponding interfaces covered in this ICD encompass the functionality to specify an Unmanned Maritime Vehicle (UMV) (surface or undersea) mission. 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.

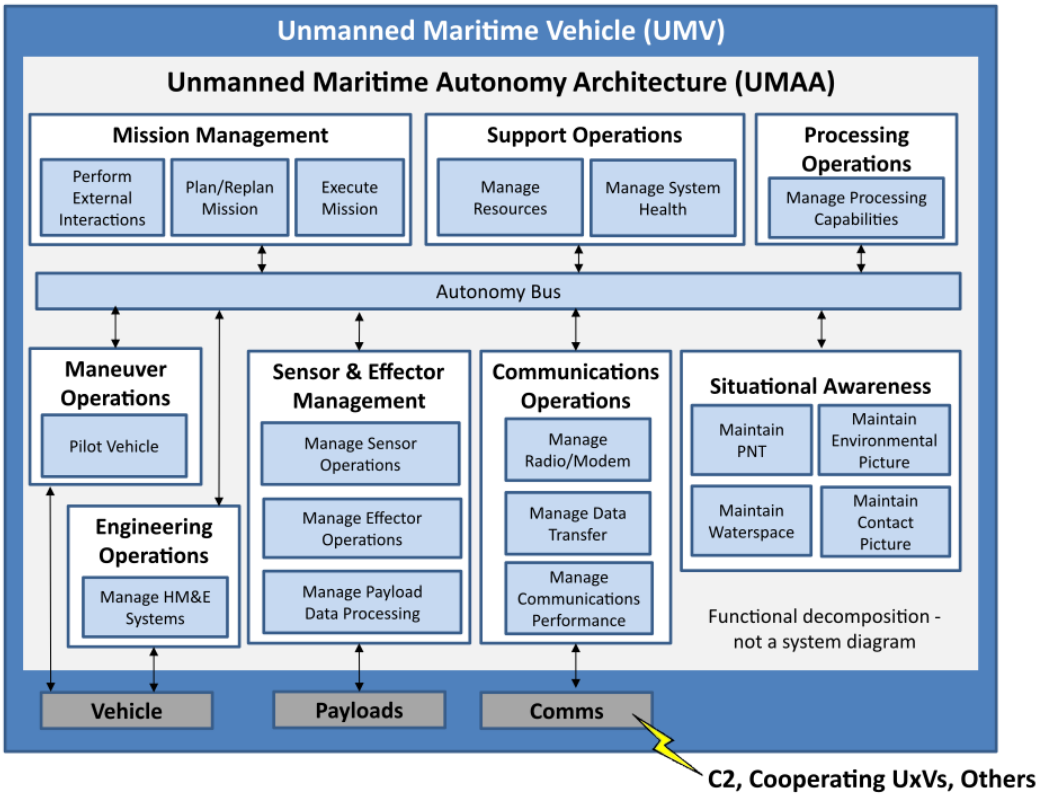

Figure 1: UMAA Functional Organization.

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.

Mission Management defines the services required to specify an unmanned vehicle mission. Figure 2 depicts an example of possible component service groupings (designated by dashed lines).

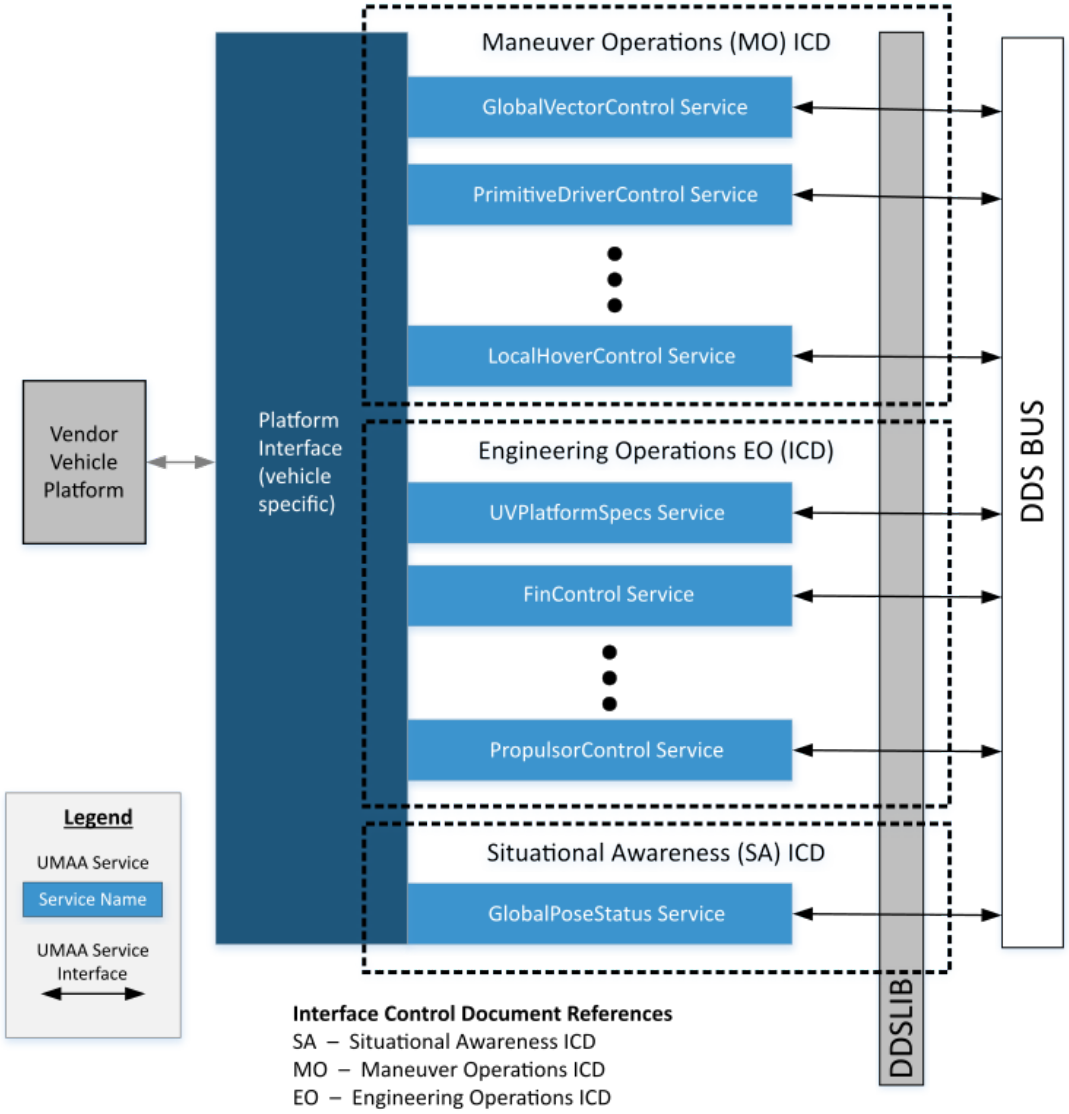

Figure 2: UMAA Services and Interfaces Example.

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 – Mission Management (MM) 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.

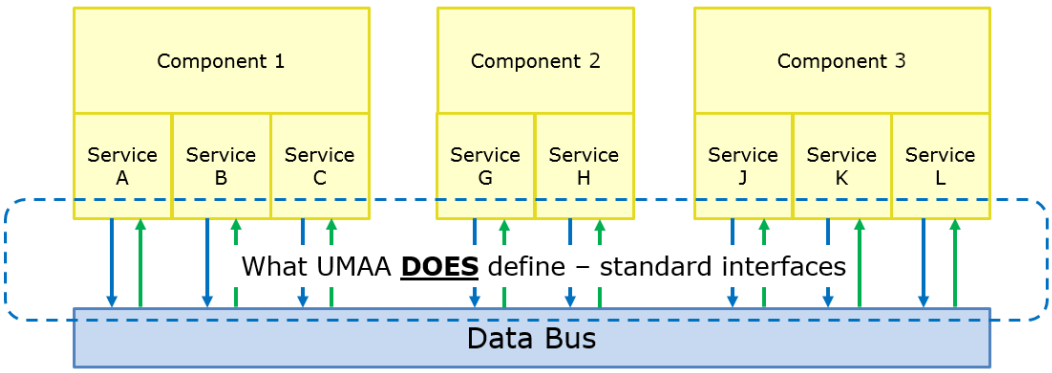

Figure 3: Services and Interfaces Exposed on the UMAA Data Bus.

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.

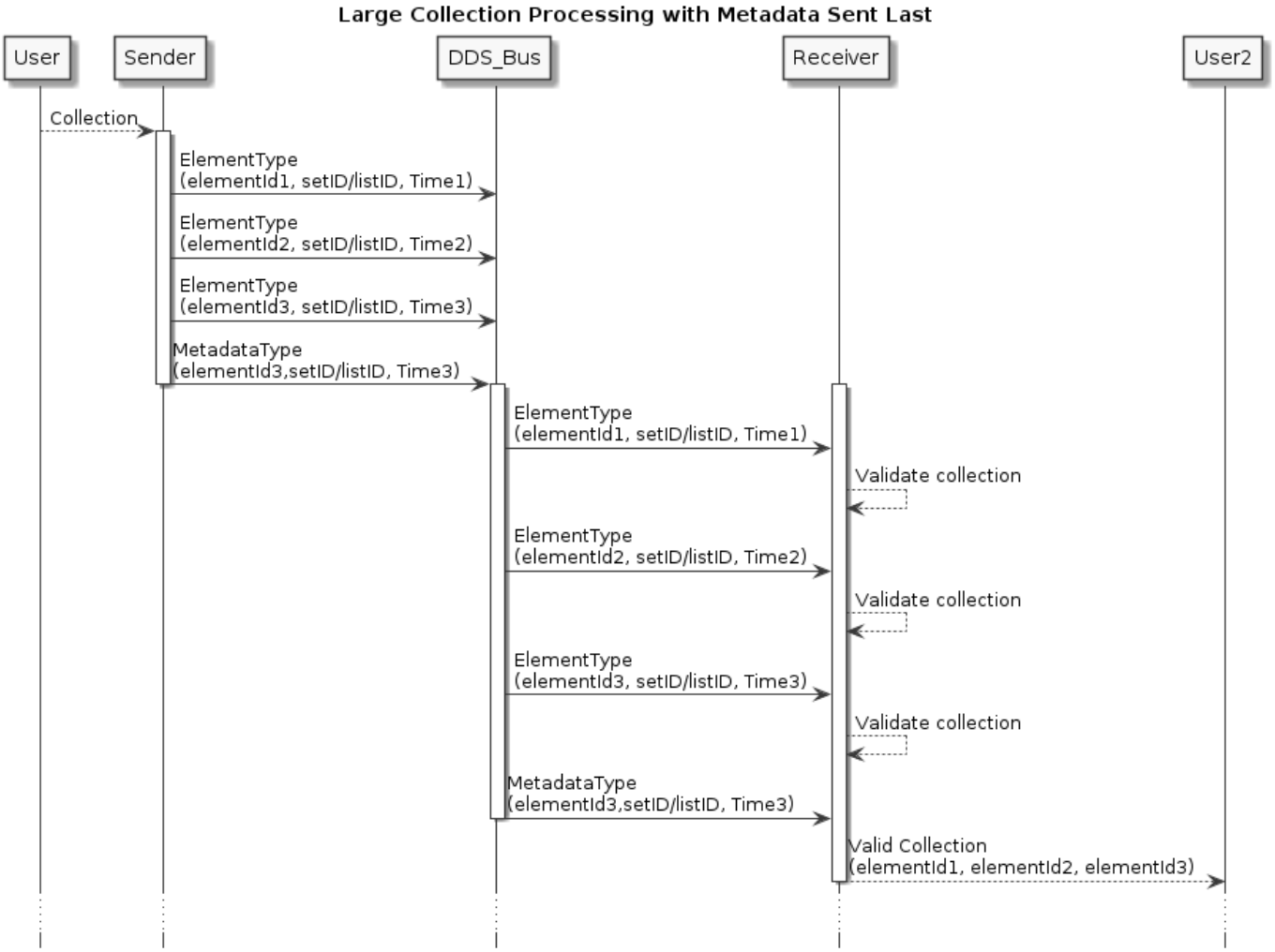

Figure 4: Sequence Diagram for initialization of a Large Collection with 3 elements.

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

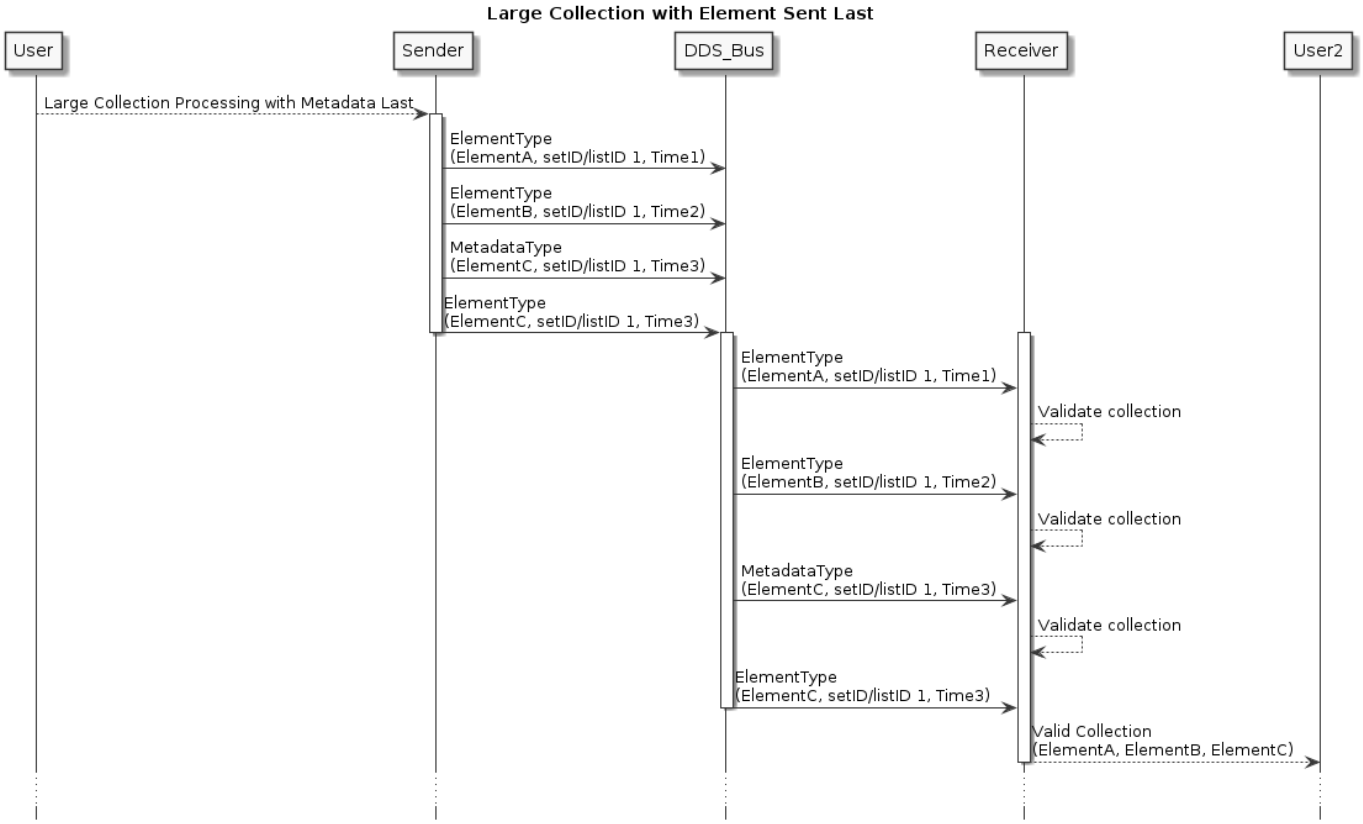

Figure 5: Sequence Diagram for initialization of a Large Collection with 3 elements.

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.

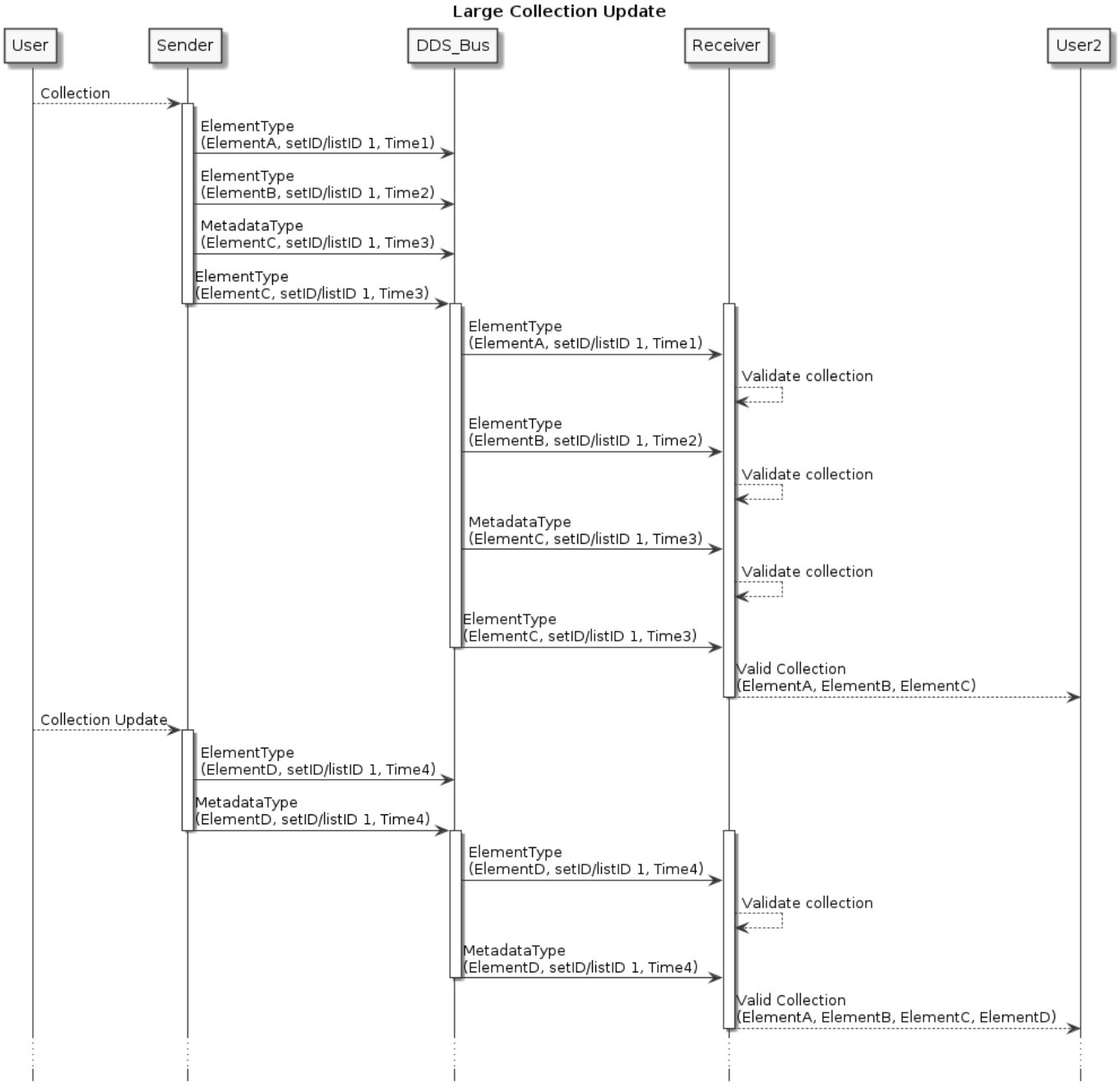

Figure 6: Sequence Diagram for update of Large Collection.

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

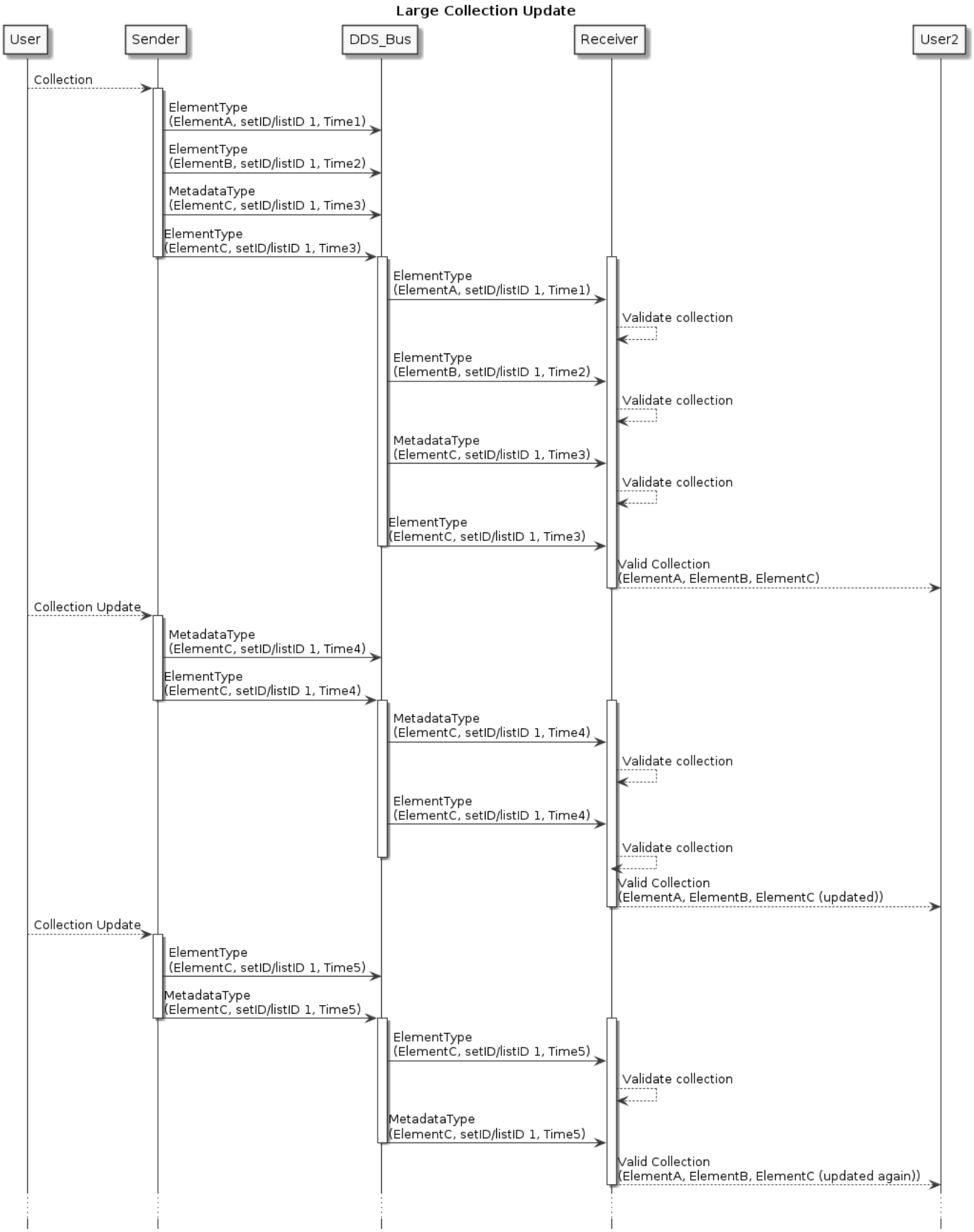

Figure 7: Sequence Diagram for update of an element of a Large Collection multiple times.

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.

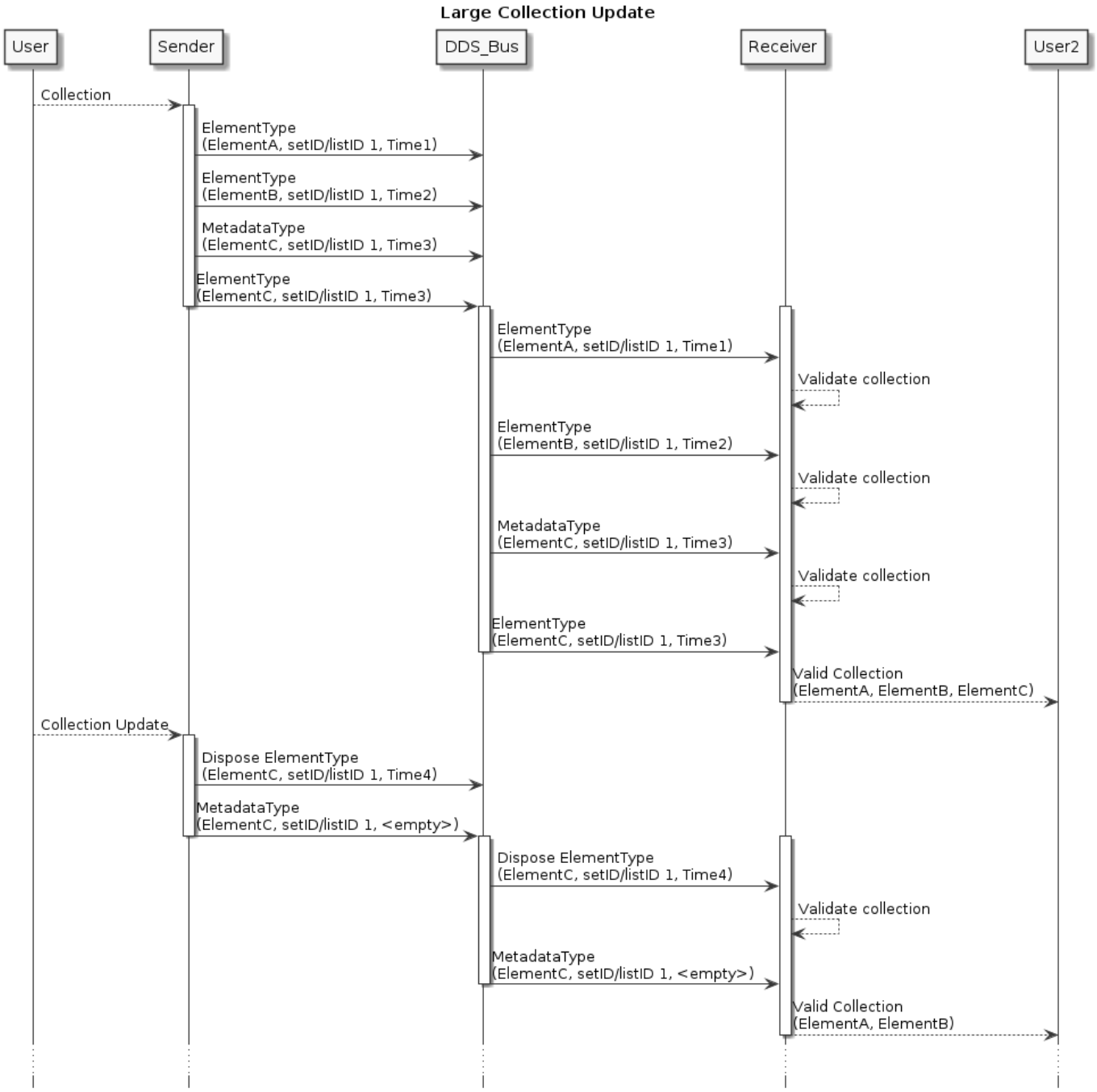

Figure 8: Sequence Diagram for delete of element from Large Collection.

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.

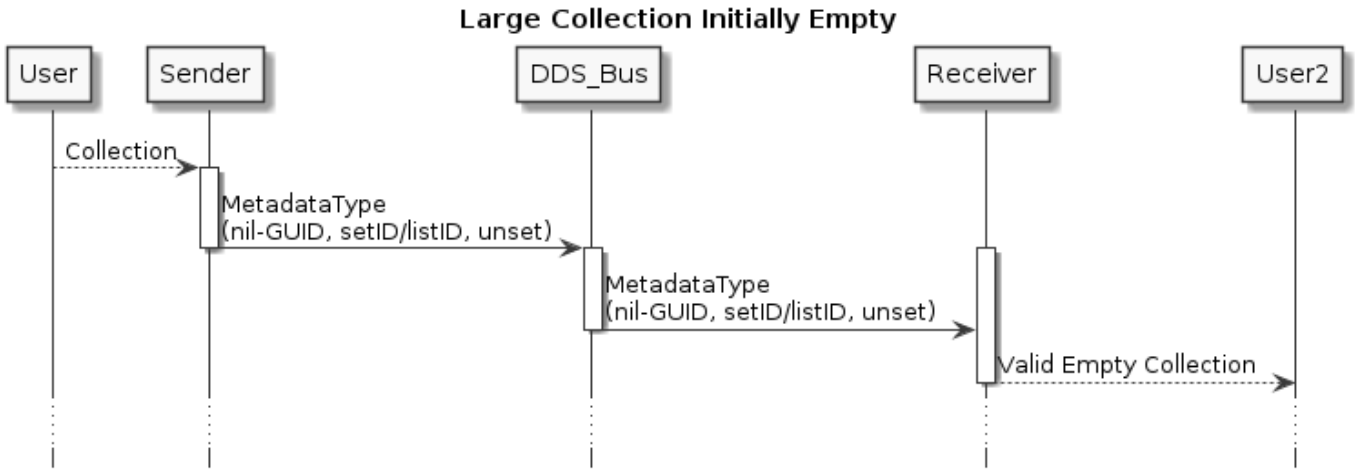

Figure 9: Sequence Diagram for initialization of an empty Large Collection.

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.

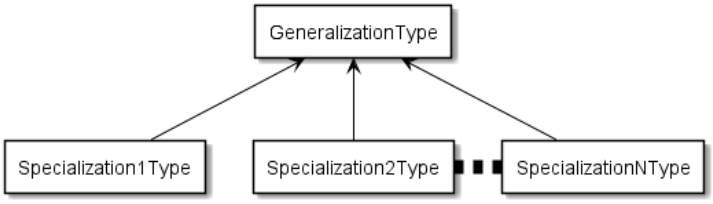

Figure 10: Generalization/Specialization UML diagram.

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.

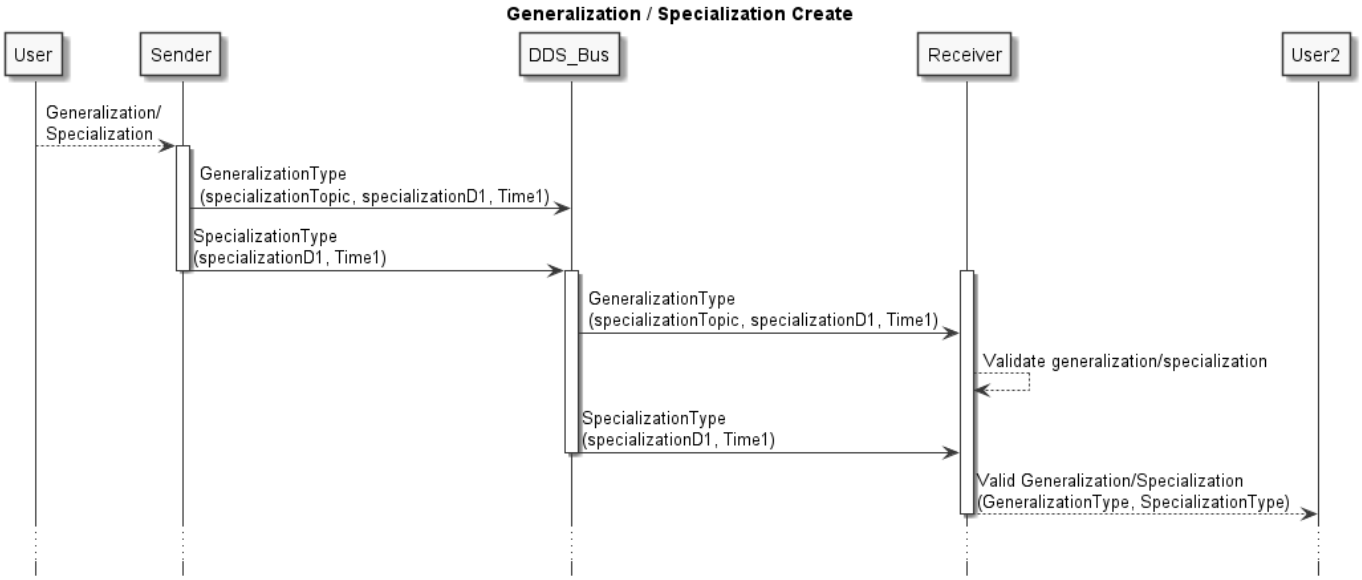

Figure 11: Sequence diagram for creating a generalization/specialization.

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.

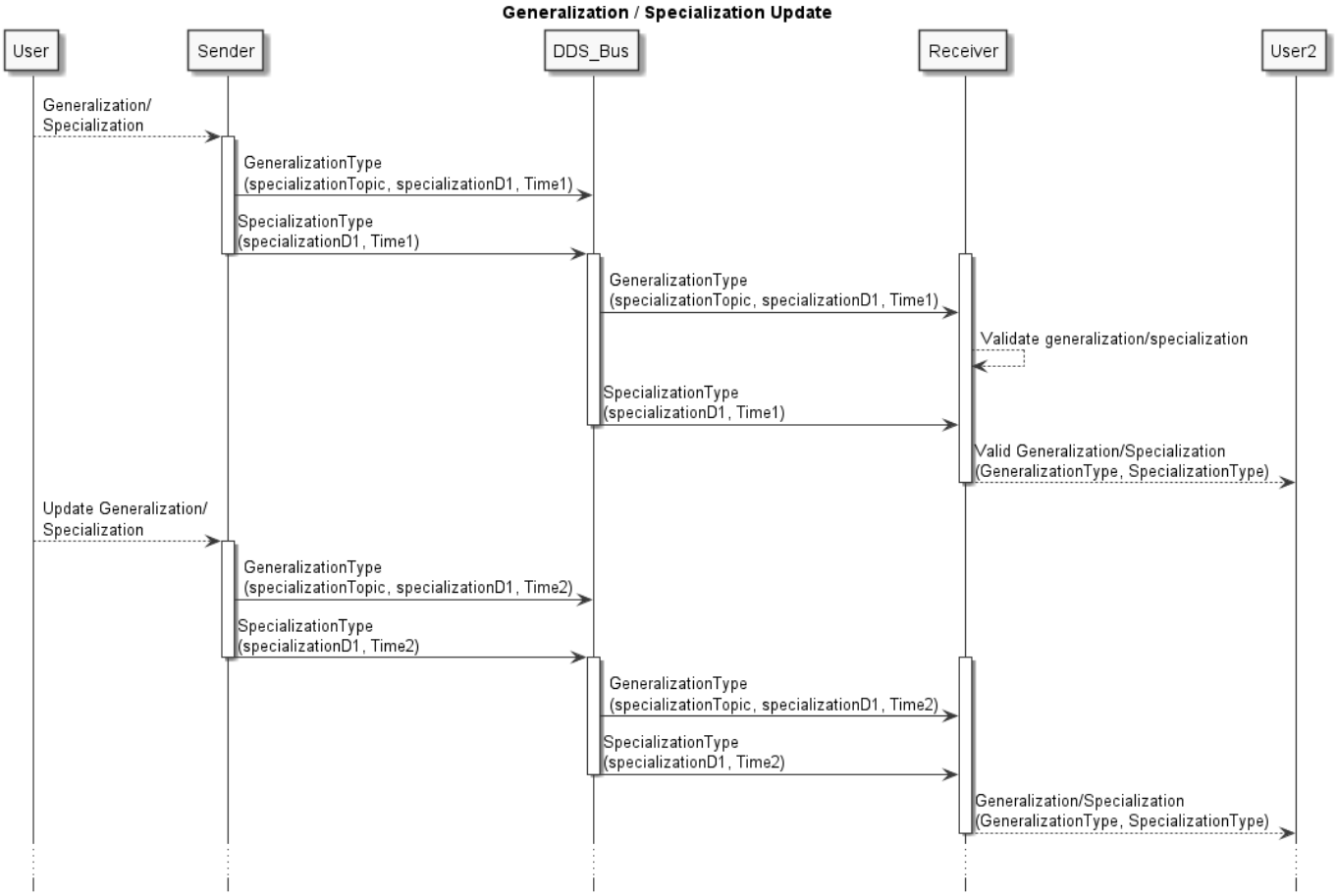

Figure 12: Sequence diagram for updating a generalization/specialization.

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.

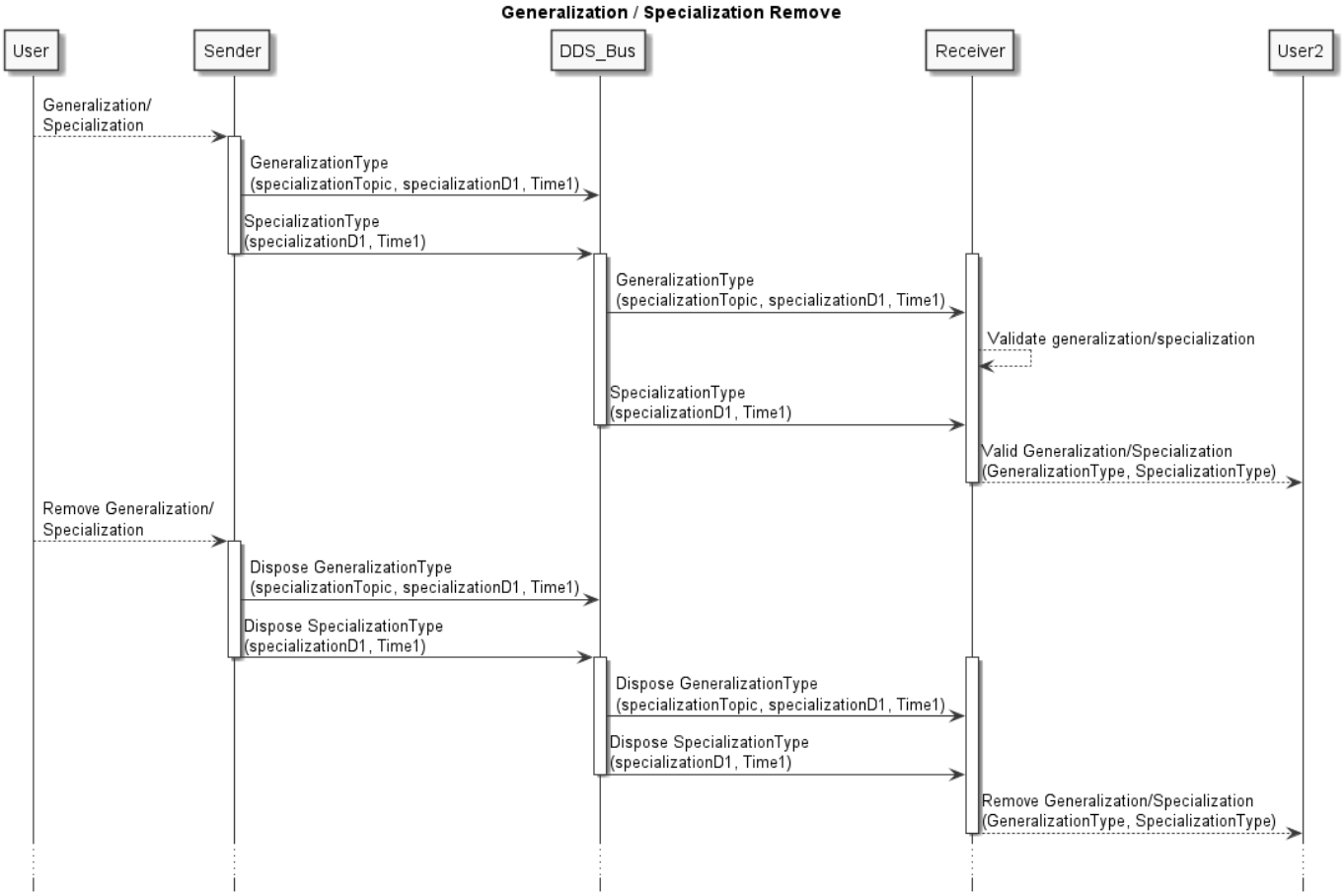

Figure 13: Sequence diagram for removing a generalization/specialization.

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.

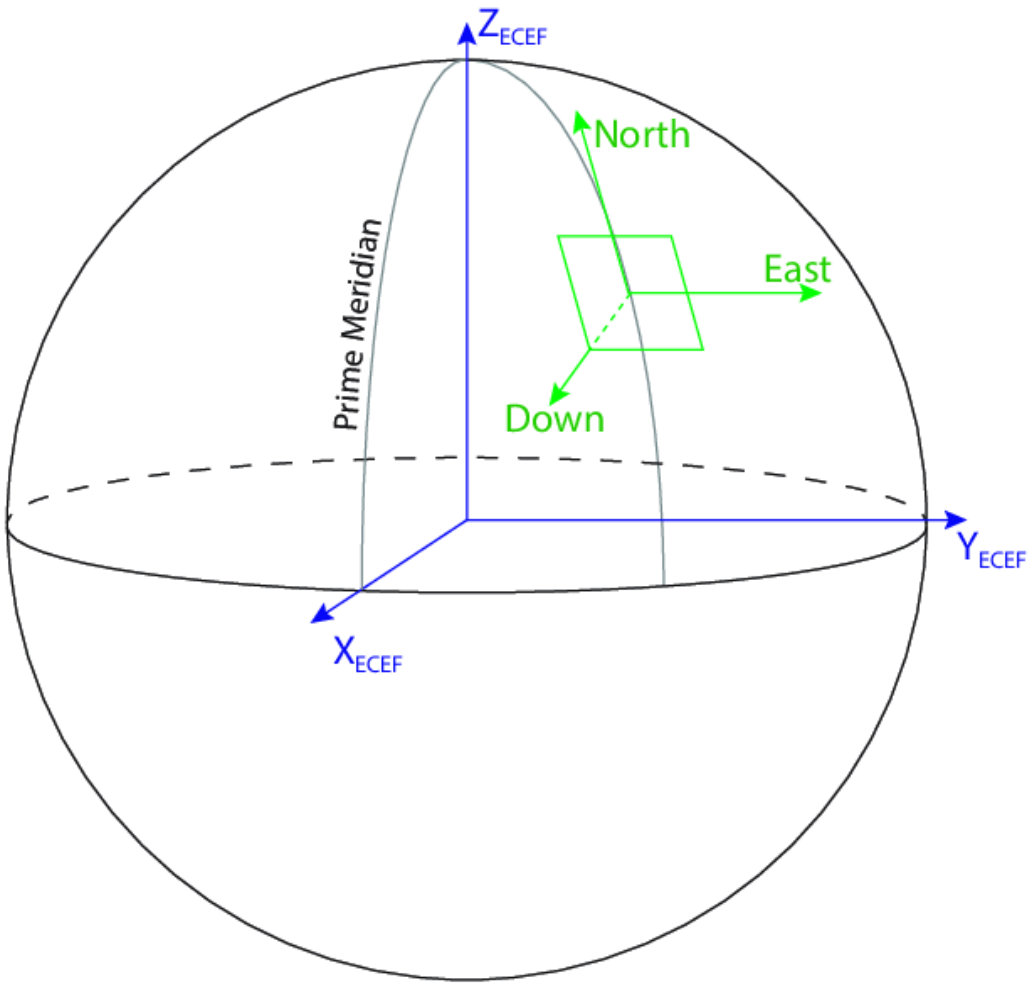

Figure 14: Origins and axes of the Earth-Centered Earth-Fixed (ECEF) and North-East-Down (NED) frames.

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).

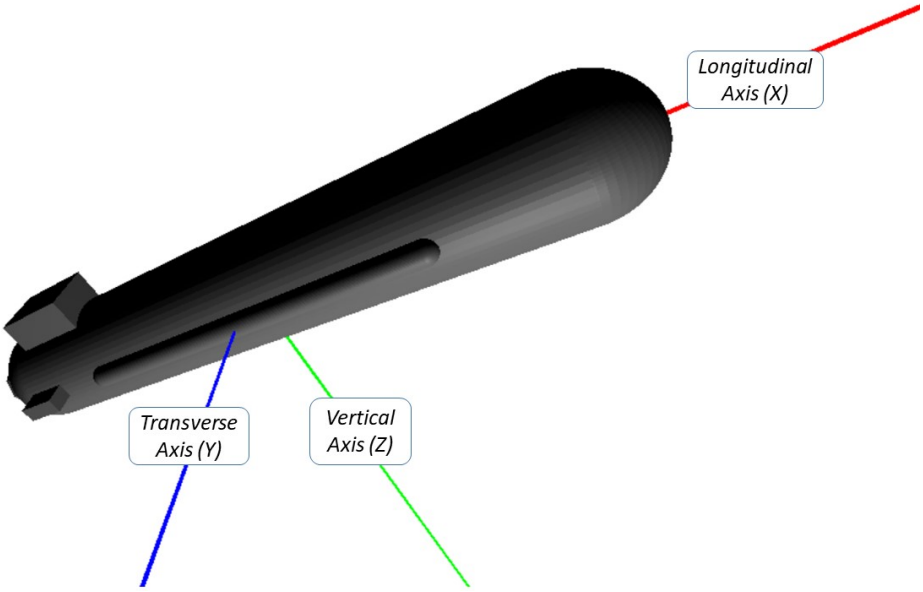

Figure 15: Define the Vehicle Coordinate System

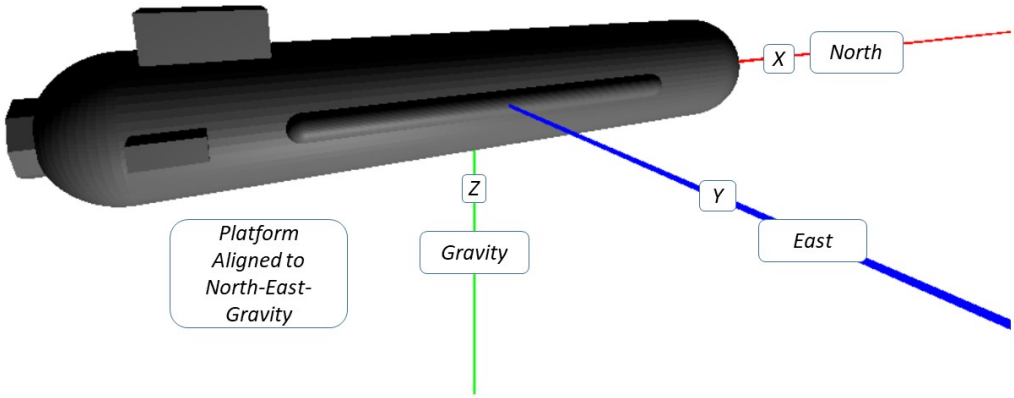

Figure 16: Align the Vehicle with the Reference Frame Axes.

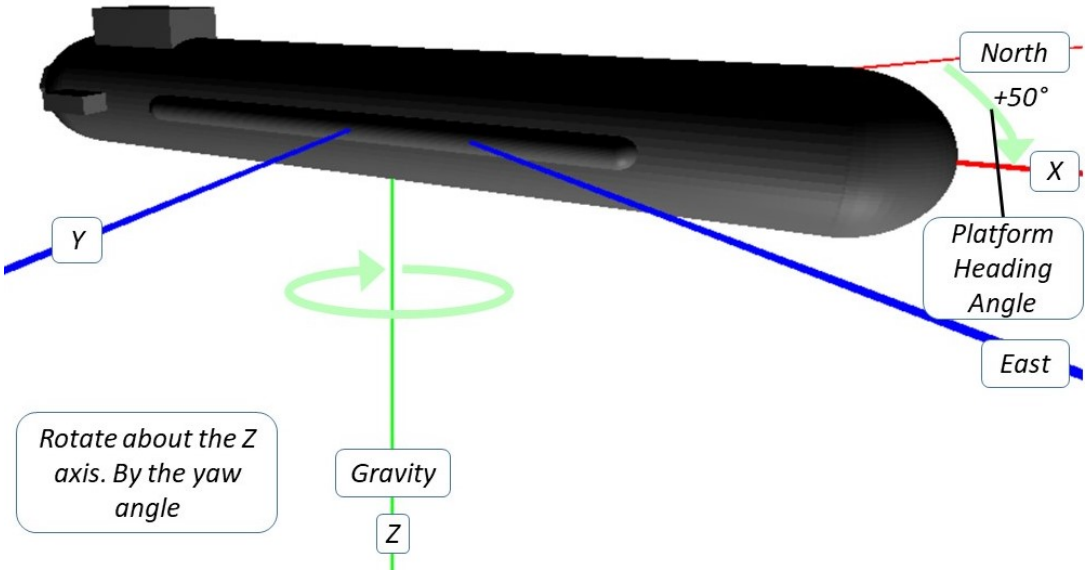

Figure 17: Rotate the Vehicle by the Yaw Angle.

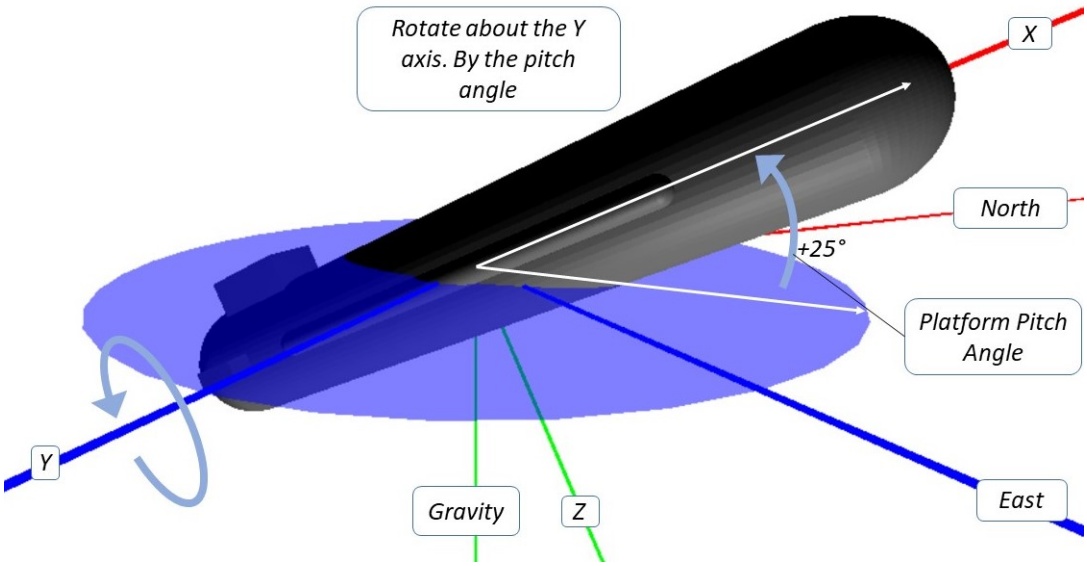

Figure 18: Rotate the Vehicle by the Pitch Angle.

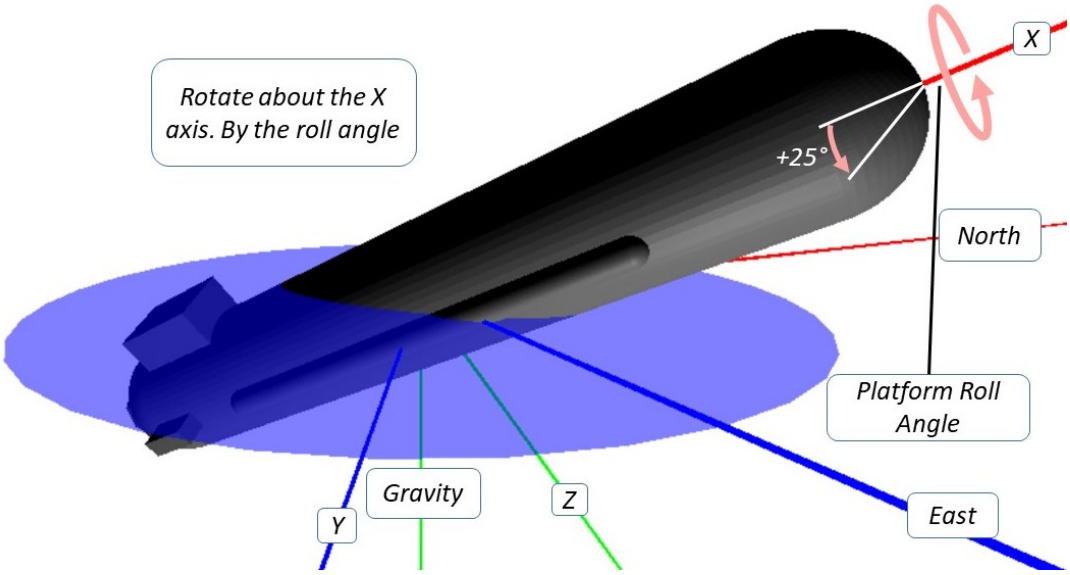

Figure 19: Rotate the Vehicle by the Roll Angle.

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.

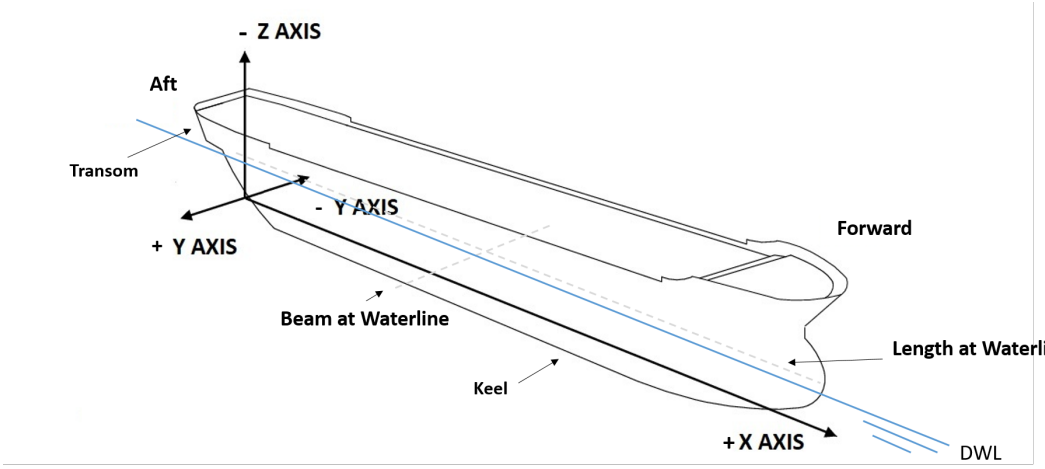

Figure 20: Keel Transom Intersection Origin Location on a USV as Example

- 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.

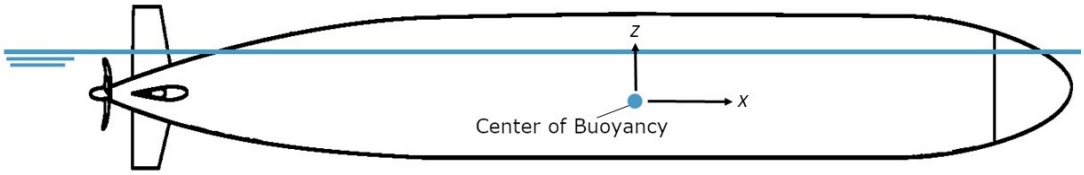

Figure 21: Center of Buoyancy Origin Location on a UUV as Example.

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.

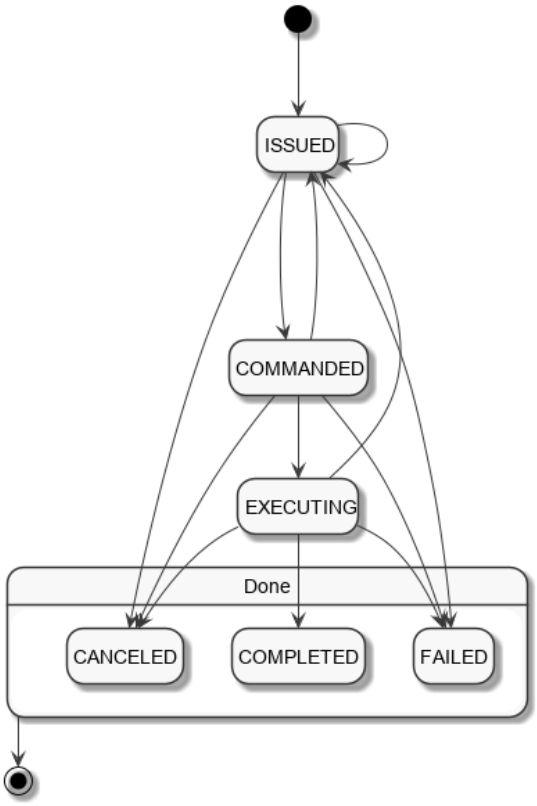

Figure 22: State transitions of the `commandStatus` as commands are processed.

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.

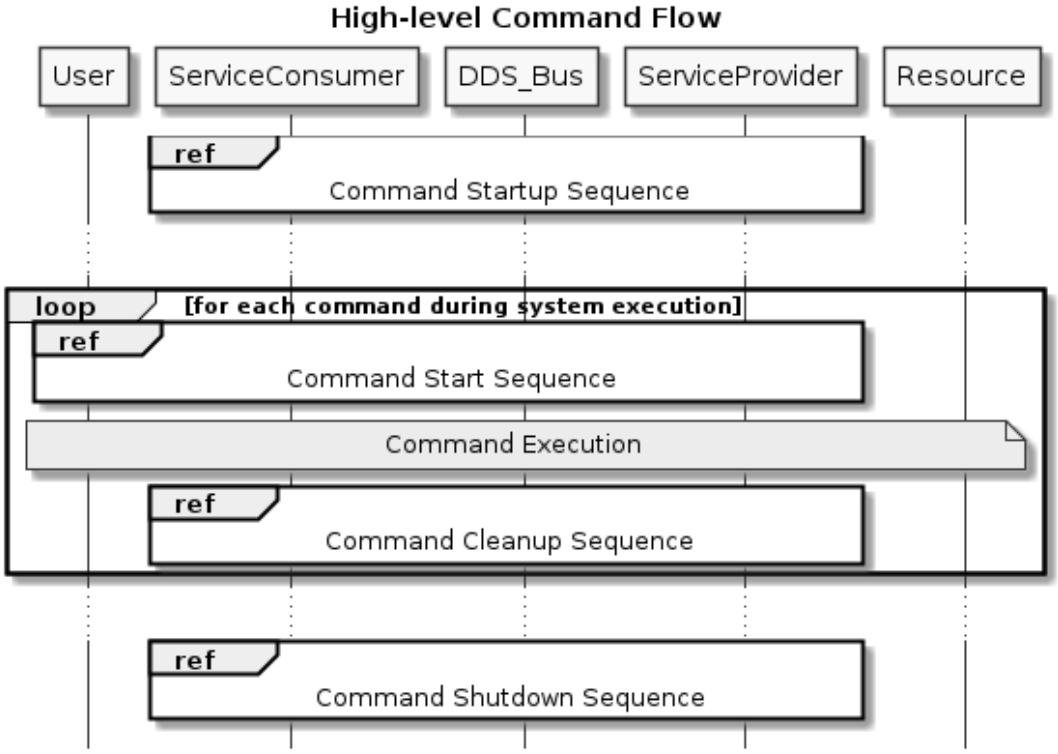

Figure 24: Sequence Diagram for the High-Level Description of a Command Execution.

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.

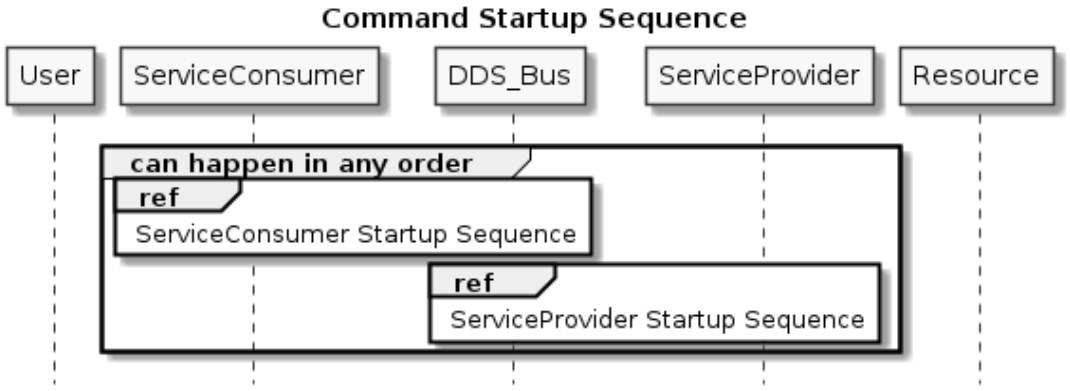

Figure 25: Sequence Diagram for Command Startup.

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.

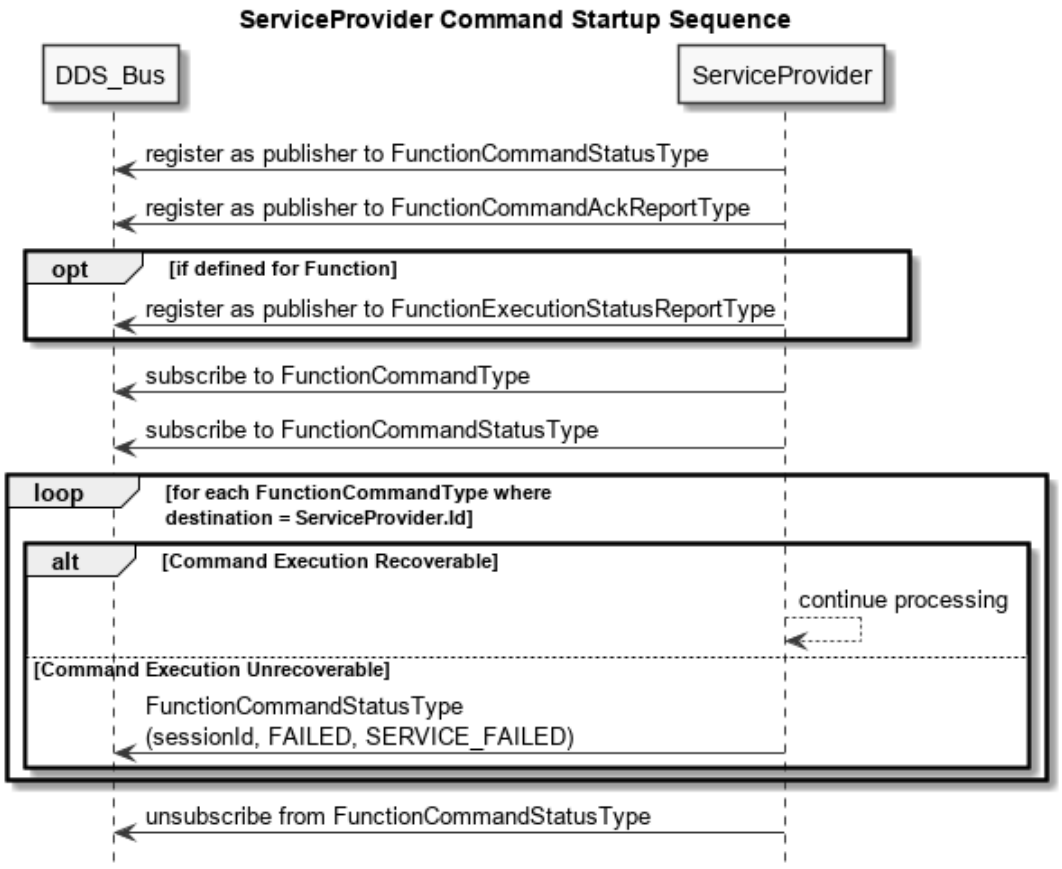

Figure 26: Sequence Diagram for Command Startup for Service Providers.

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.

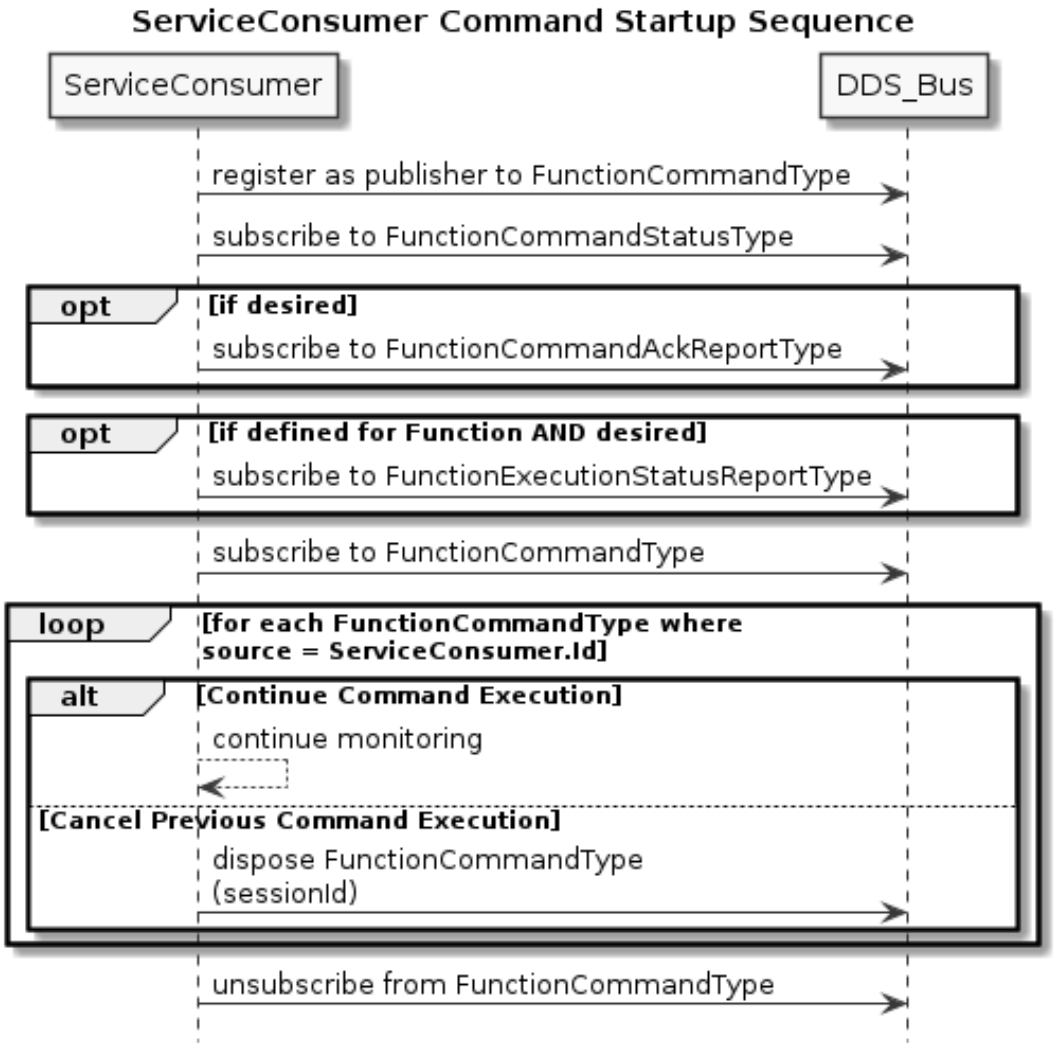

Figure 27: Sequence Diagram for Command Startup for Service Consumers.

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).

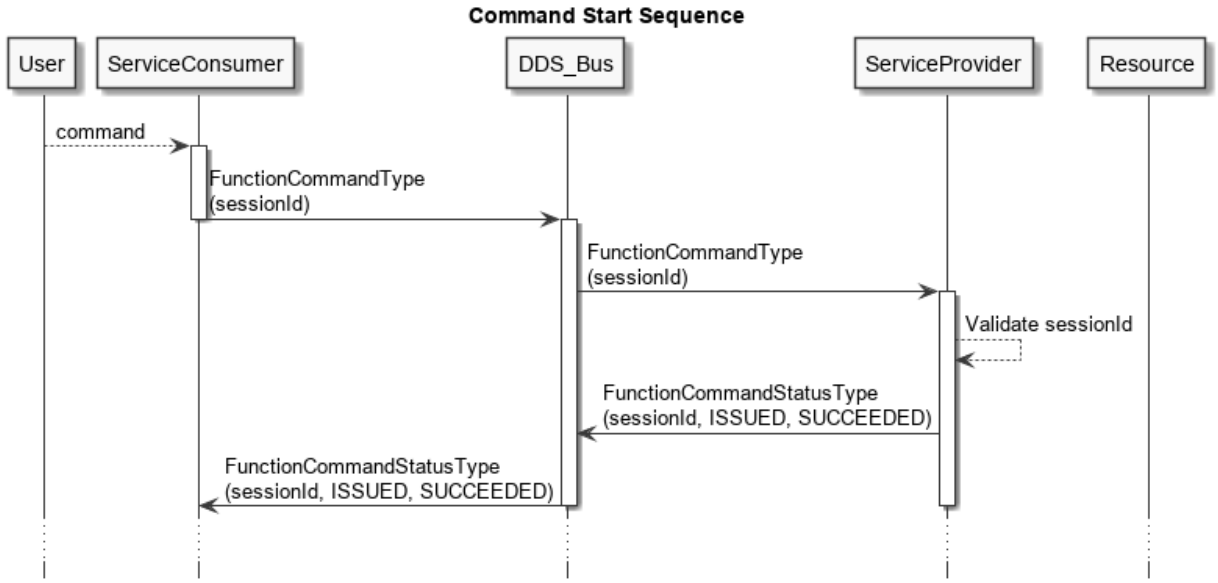

Figure 28: Sequence Diagram for the Start of a Command Execution.

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.

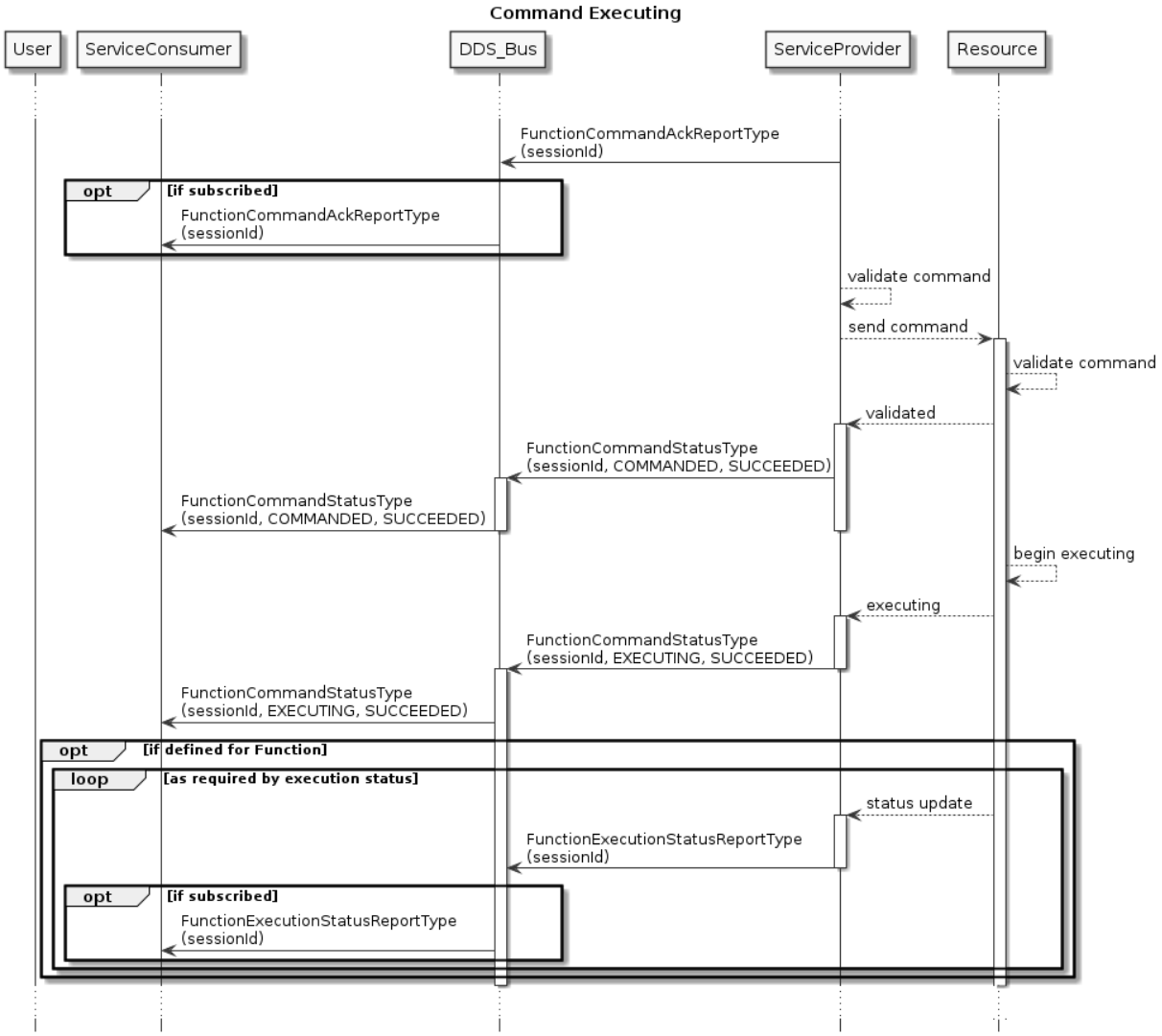

Figure 29: Beginning Sequence Diagram for a Command Execution.

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:

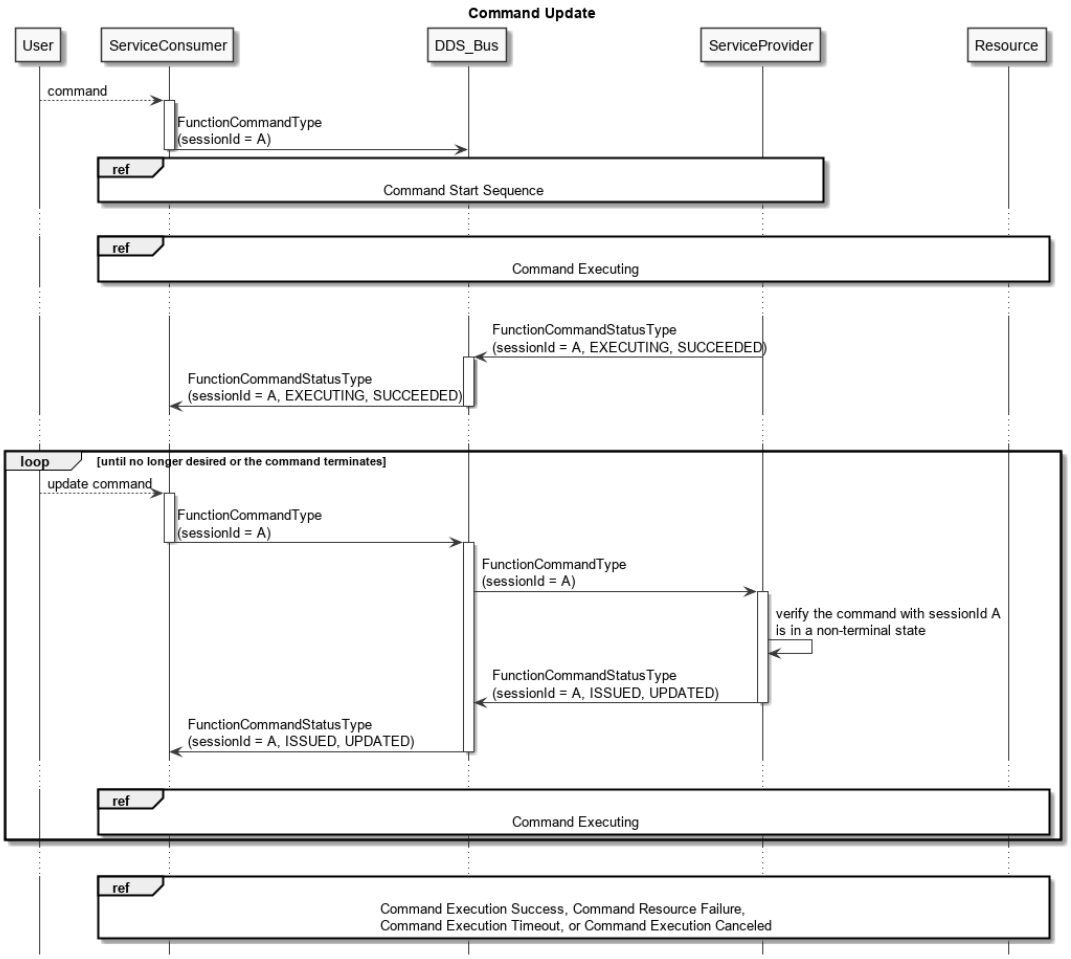

Figure 30: Sequence Diagram for Command Update.

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.

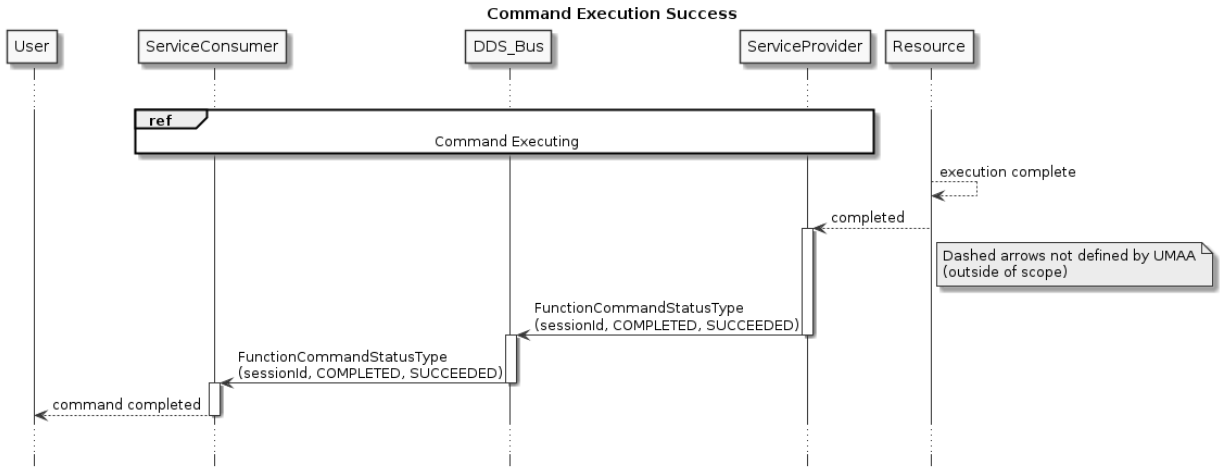

Figure 31: Sequence Diagram for a Command That Completes Successfully.

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.

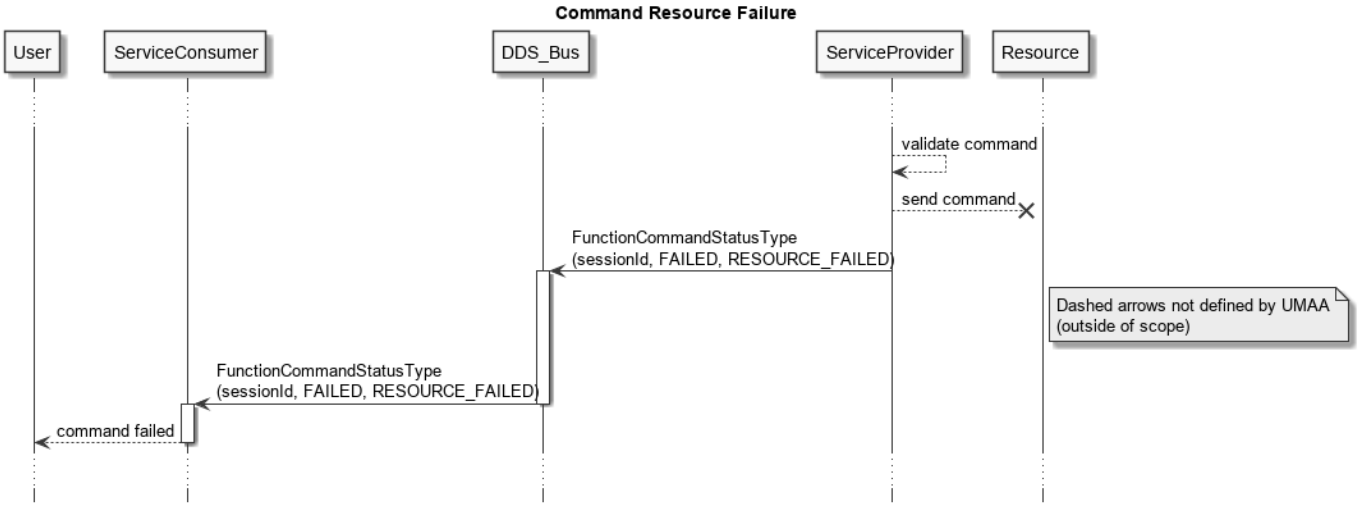

Figure 32: Sequence Diagram for a Command That Fails due to Resource Failure.

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.

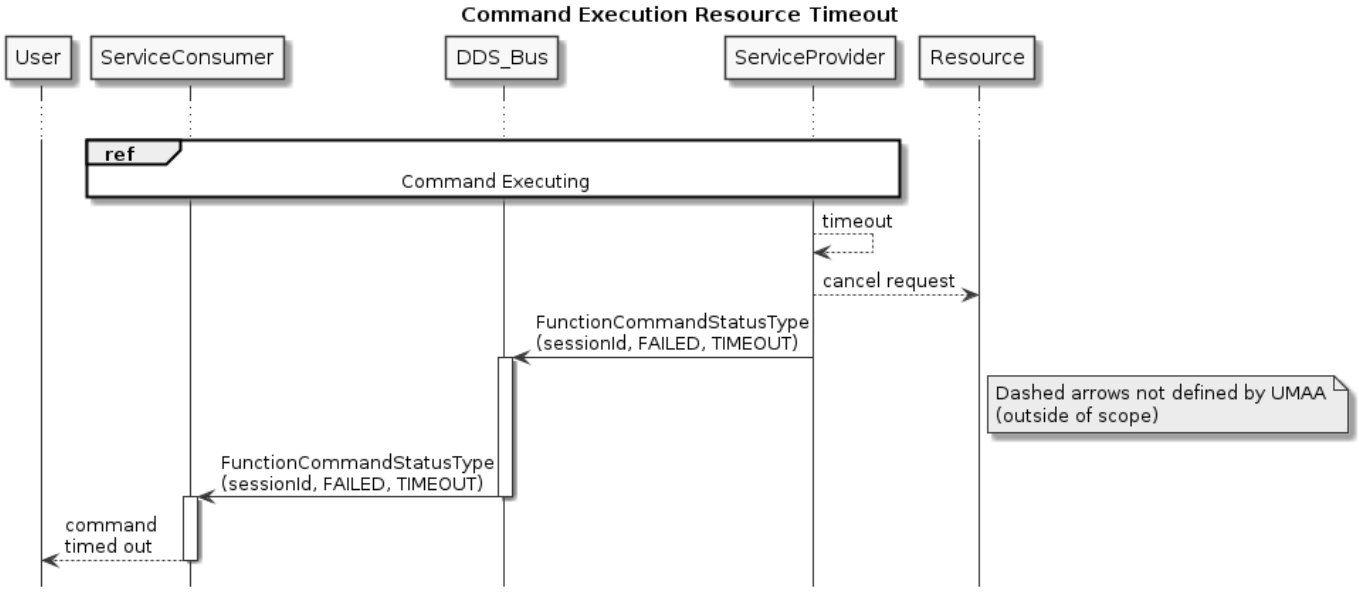

Figure 33: Sequence Diagram for a Command That Times Out Before Completing.

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).

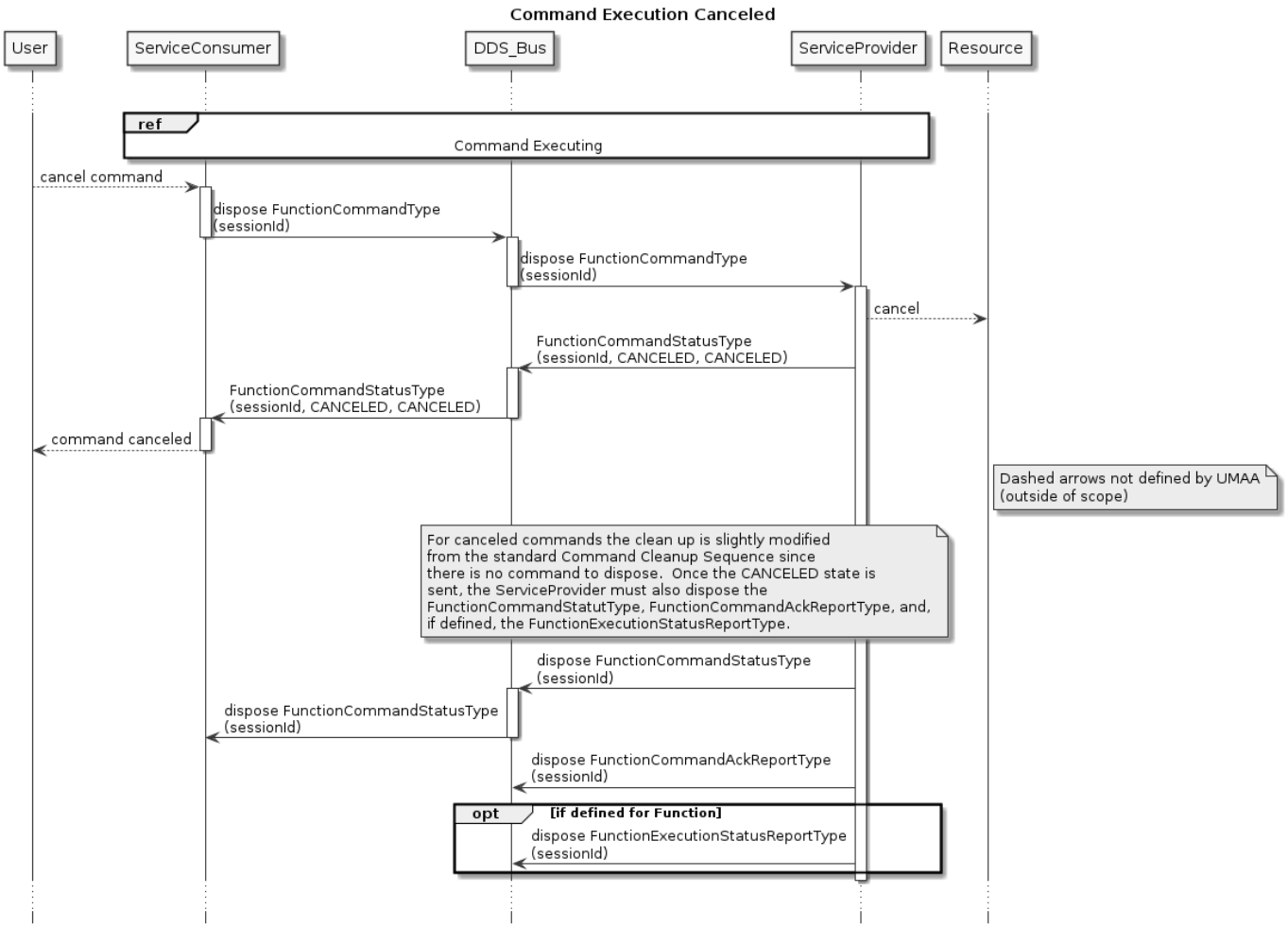

Figure 34: Sequence Diagram for a Command That is Canceled by the Service Consumer Before the Service Provider can Complete It.

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.

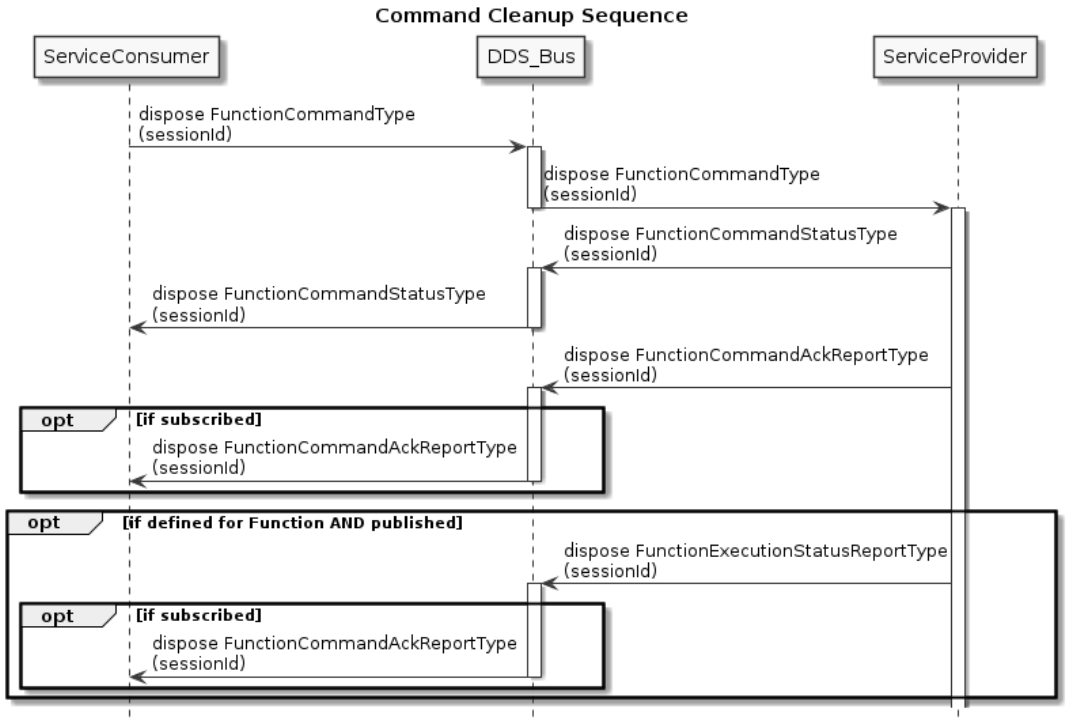

Figure 35: Sequence Diagram Showing Cleanup of the Bus When a Command Has Been Completed and the Service Consumer No Longer Wishes to Maintain the Commanded State.

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.

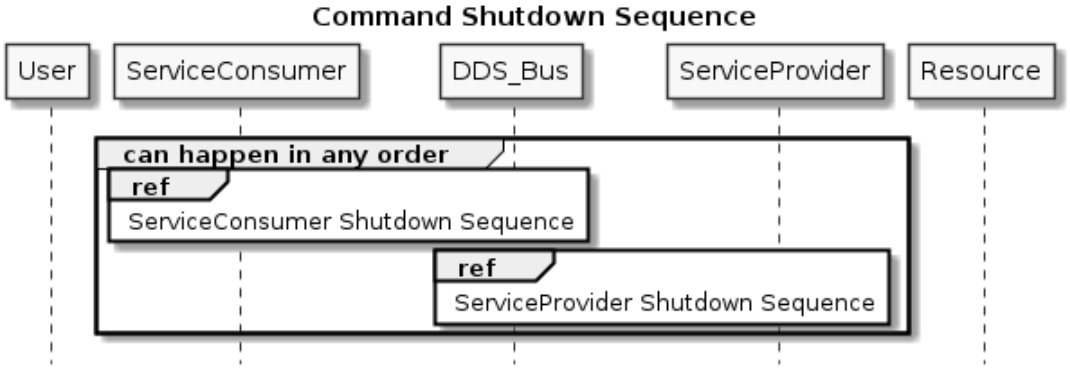

Figure 36: Sequence Diagram for Command Shutdown.

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.

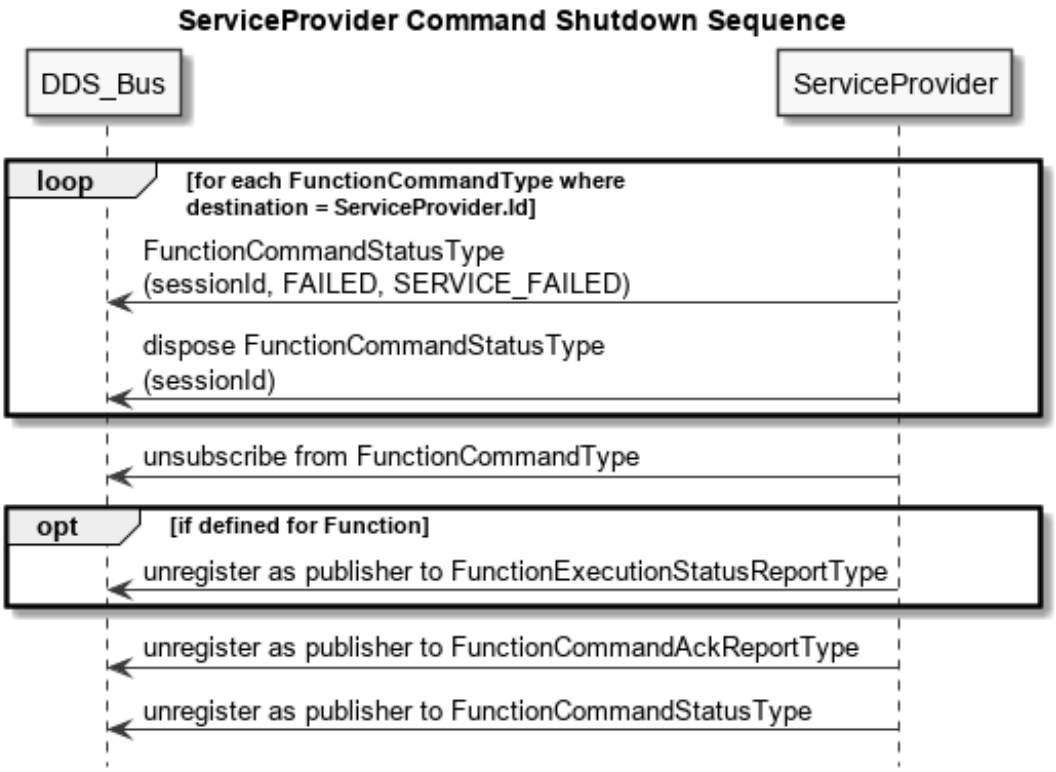

Figure 37: Sequence Diagram for Command Shutdown for Service Providers.

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.

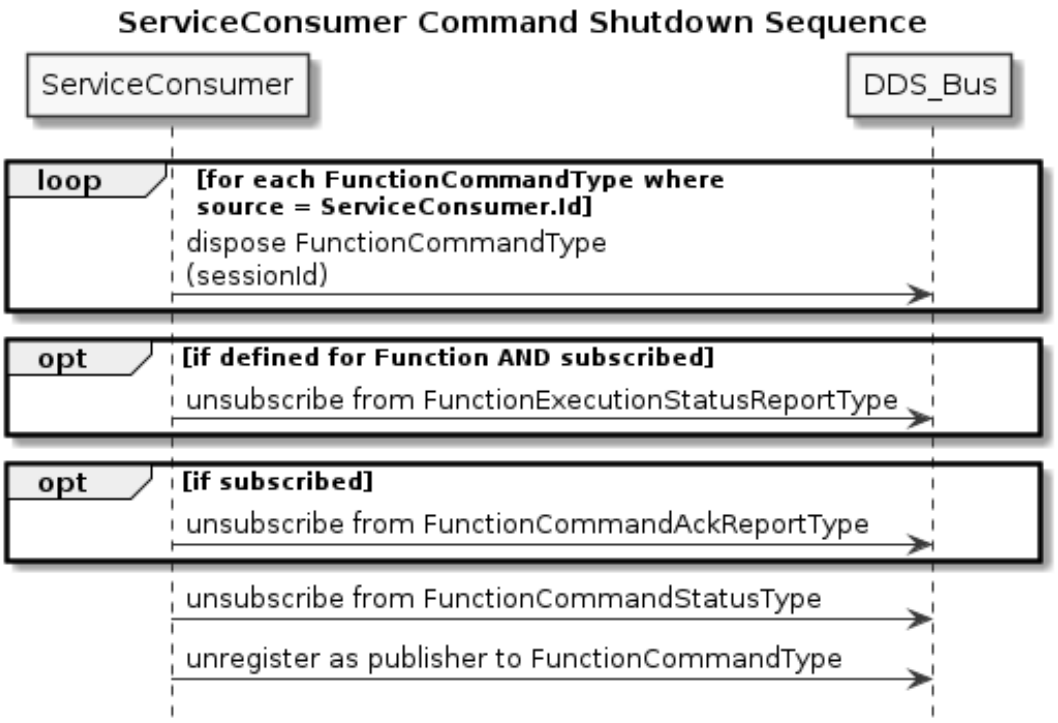

Figure 38: Sequence Diagram for Command Shutdown for Service Consumers.

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).

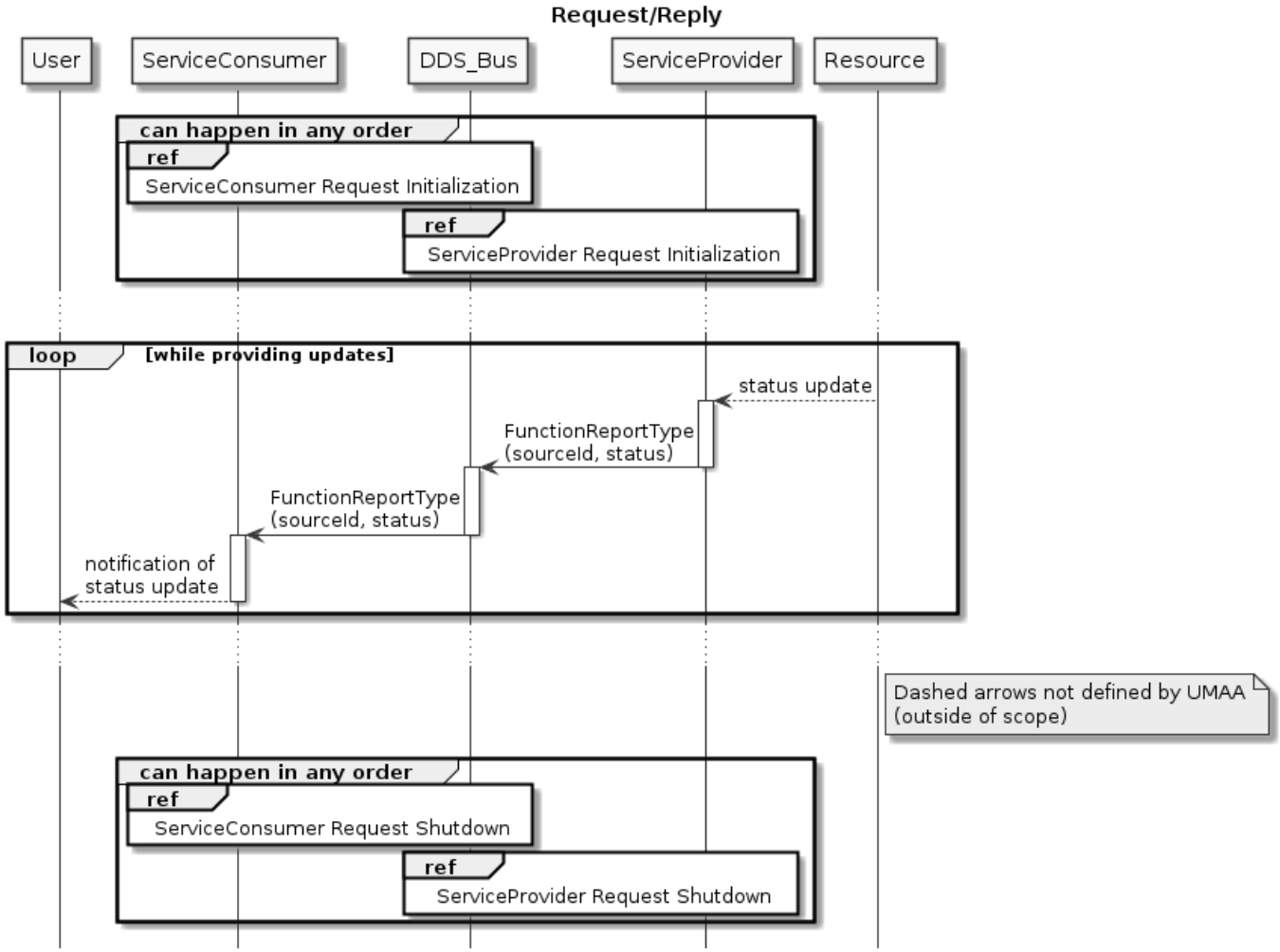

Figure 39: Sequence Diagram for a Request/Reply for Report Data That Does Not Require any Specific Query Data.

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.

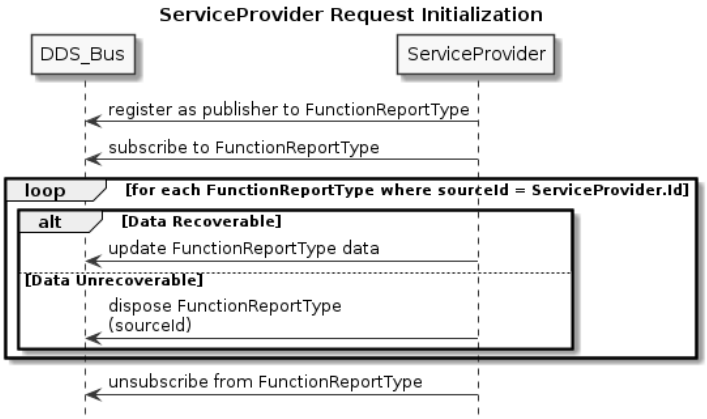

Figure 40: Sequence Diagram for Initialization of a Service Provider to Provide `FunctionReportType`s.

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.

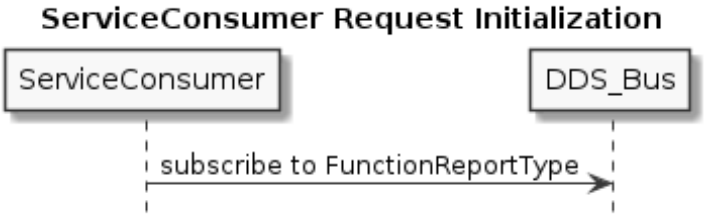

Figure 41: Sequence Diagram for Initialization of a Service Consumer to Request `FunctionReportType`s.

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).

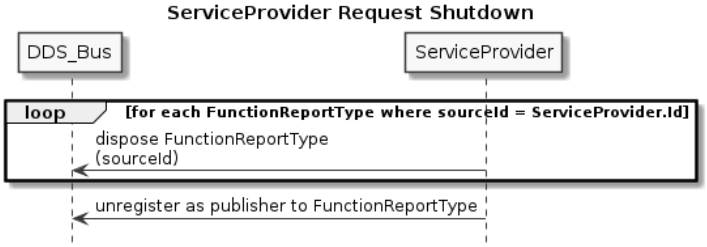

Figure 42: Sequence Diagram for Shutdown of a Service Provider.

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

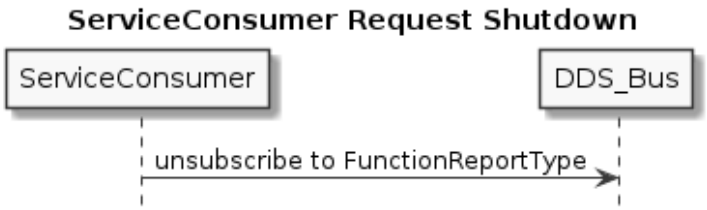

Figure 43: Sequence Diagram for Shutdown of a Service Consumer.

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 Mission Management (MM) Services and Interfaces

6.1 Services and Interfaces

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

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

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

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

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

Operations without DDS Topics

⊕ = 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 ActiveConstraintsControl

The purpose of this service is to provide a set of active constraints. Constraints are specified using conditional statements, where the conditional statements must be kept true while executing any actions.

Table 8: ActiveConstraintsControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setActiveConstraints	reportActiveConstraintsCommandStatus
queryActiveConstraintsCommandAck⊕	reportActiveConstraintsCommandAck
cancelActiveConstraintsCommand⊕	reportActiveConstraintsCancelCommandStatus⊕

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

6.1.1.1 reportActiveConstraintsCommandAck

Description: This operation is used to provide the ActiveConstraints commanded values.

Namespace: UMAA::MM::ActiveConstraintsControl

Topic: ActiveConstraintsCommandAckReportType

Data Type: ActiveConstraintsCommandAckReportType

Table 9: ActiveConstraintsCommandAckReportType Message Definition

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

6.1.1.2 reportActiveConstraintsCommandStatus

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

Namespace: UMAA::MM::ActiveConstraintsControl

Topic: ActiveConstraintsCommandStatusType

Data Type: ActiveConstraintsCommandStatusType

Table 10: ActiveConstraintsCommandStatusType Message Definition

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

6.1.1.3 setActiveConstraints

Description: This operation is used to set the ActiveConstraints command.

Namespace: UMAA::MM::ActiveConstraintsControl

Topic: ActiveConstraintsCommandType

Data Type: ActiveConstraintsCommandType

Table 11: ActiveConstraintsCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
constraintConditionalIDs	sequence<NumericGUID> max size = 256	Provides a reference to each conditional that represents an active constraint.

6.1.2 ConditionalControl

The purpose of this service is to manage conditionals that initiates the execution of an objective and/or enables a constraint.

Table 12: ConditionalControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setConditionalAdd	reportConditionalAddCommandStatus
queryConditionalAddCommandAck⊕	reportConditionalAddCommandAck
cancelConditionalAddCommand⊕	reportConditionalAddCancelCommandStatus⊕
setConditionalDelete	reportConditionalDeleteCommandStatus
queryConditionalDeleteCommandAck⊕	reportConditionalDeleteCommandAck
cancelConditionalDeleteCommand⊕	reportConditionalDeleteCancelCommandStatus⊕

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

6.1.2.1 reportConditionalAddCommandAck

Description: This operation is used to provide the ConditionalAdd commanded values.

Namespace: UMAA::MM::ConditionalControl

Topic: ConditionalAddCommandAckReportType

Data Type: ConditionalAddCommandAckReportType

Table 13: ConditionalAddCommandAckReportType Message Definition

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

6.1.2.2 reportConditionalAddCommandStatus

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

Namespace: UMAA::MM::ConditionalControl

Topic: ConditionalAddCommandStatusType

Data Type: ConditionalAddCommandStatusType

Table 14: ConditionalAddCommandStatusType Message Definition

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

6.1.2.3 reportConditionalDeleteCommandAck

Description: This operation is used to provide the ConditionalDelete commanded values.

Namespace: UMAA::MM::ConditionalControl

Topic: ConditionalDeleteCommandAckReportType

Data Type: ConditionalDeleteCommandAckReportType

Table 15: ConditionalDeleteCommandAckReportType Message Definition

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

6.1.2.4 reportConditionalDeleteCommandStatus

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

Namespace: UMAA::MM::ConditionalControl

Topic: ConditionalDeleteCommandStatusType

Data Type: ConditionalDeleteCommandStatusType

Table 16: ConditionalDeleteCommandStatusType Message Definition

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

6.1.2.5 setConditionalAdd

Description: This operation is used to add a new conditional. If the conditionalID already exists, the operation has no effect and the associated command status must be reported as FAILED.

Namespace: UMAA::MM::ConditionalControl

Topic: ConditionalAddCommandType

Data Type: ConditionalAddCommandType

Table 17: ConditionalAddCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
conditional	ConditionalType	Specifies the conditional to be added.

6.1.2.6 setConditionalDelete

Description: This operation is used to delete an existing conditional.

Namespace: UMAA::MM::ConditionalControl

Topic: ConditionalDeleteCommandType

Data Type: ConditionalDeleteCommandType

Table 18: ConditionalDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
conditionalID†	NumericGUID	Specifies the identifier of the conditional to be deleted. If the identifier is not specified, all conditionals are to be deleted.

6.1.3 ConditionalReport

The purpose of this service is to report the set of conditionals that have successfully been added for use during mission plan execution. Each conditional expresses a condition that can be evaluated to either true or false. They are used in both triggers and constraints. In triggers, conditionals are used as either a state transition trigger or a constraint trigger, which define the condition(s) that activate a state transition or constraint, respectively. In constraints, conditionals are used to define the condition(s) that must be upheld by the system during execution.

Table 19: ConditionalReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryConditional⊕	reportConditional

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

6.1.3.1 reportConditional

Description: This operation is used to provide the current set of conditionals for mission plan execution.

Namespace: UMAA::MM::ConditionalReport

Topic: ConditionalReportType

Data Type: ConditionalReportType

Table 20: ConditionalReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
conditionals→setID	LargeSet<ConditionalType>	Defines the current set of conditionals for mission plan execution. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::ConditionalReport::ConditionalReportTypeConditionalsSetElement.

6.1.4 MissionPlanConstraintControl

The purpose of this service is to provide the capability of adding/deleting constraints to/from the mission plan that must be handled during mission plan execution.

Table 21: MissionPlanConstraintControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanConstraintAdd	reportMissionPlanConstraintAddCommandStatus
queryMissionPlanConstraintAddCommandAck⊕	reportMissionPlanConstraintAddCommandAck
cancelMissionPlanConstraintAddCommand⊕	reportMissionPlanConstraintAddCancelCommandStatus⊕
setMissionPlanConstraintDelete	reportMissionPlanConstraintDeleteCommandStatus
queryMissionPlanConstraintDeleteCommandAck⊕	reportMissionPlanConstraintDeleteCommandAck
cancelMissionPlanConstraintDeleteCommand⊕	reportMissionPlanConstraintDeleteCancelCommandStatus⊕

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

6.1.4.1 reportMissionPlanConstraintAddCommandAck

Description: This operation is used to provide the MissionPlanConstraintAdd commanded values.

Namespace: UMAA::MM::MissionPlanConstraintControl

Topic: MissionPlanConstraintAddCommandAckReportType

Data Type: MissionPlanConstraintAddCommandAckReportType

Table 22: MissionPlanConstraintAddCommandAckReportType Message Definition

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

6.1.4.2 reportMissionPlanConstraintAddCommandStatus

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

Namespace: UMAA::MM::MissionPlanConstraintControl

Topic: MissionPlanConstraintAddCommandStatusType

Data Type: MissionPlanConstraintAddCommandStatusType

Table 23: MissionPlanConstraintAddCommandStatusType Message Definition

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

6.1.4.3 reportMissionPlanConstraintDeleteCommandAck

Description: This operation is used to provide the MissionPlanConstraintDelete commanded values.

Namespace: UMAA::MM::MissionPlanConstraintControl

Topic: MissionPlanConstraintDeleteCommandAckReportType

Data Type: MissionPlanConstraintDeleteCommandAckReportType

Table 24: MissionPlanConstraintDeleteCommandAckReportType Message Definition

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

6.1.4.4 reportMissionPlanConstraintDeleteCommandStatus

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

Namespace: UMAA::MM::MissionPlanConstraintControl

Topic: MissionPlanConstraintDeleteCommandStatusType

Data Type: MissionPlanConstraintDeleteCommandStatusType

Table 25: MissionPlanConstraintDeleteCommandStatusType Message Definition

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

6.1.4.5 setMissionPlanConstraintAdd

Description: This operation adds a new planned constraint that must be handled during mission plan execution.

Namespace: UMAA::MM::MissionPlanConstraintControl

Topic: MissionPlanConstraintAddCommandType

Data Type: MissionPlanConstraintAddCommandType

Table 26: MissionPlanConstraintAddCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
constraint	ConstraintType	Specifies the constraint to be added.

6.1.4.6 setMissionPlanConstraintDelete

Description: This operation deletes an existing planned constraint from the mission plan.

Namespace: UMAA::MM::MissionPlanConstraintControl

Topic: MissionPlanConstraintDeleteCommandType

Data Type: MissionPlanConstraintDeleteCommandType

Table 27: MissionPlanConstraintDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
constraintID†	NumericGUID	Specifies the identifier of the constraint plan that is to be deleted. If the identifier is not specified, all constraints are to be deleted.

6.1.5 MissionPlanExecutionControl

The purpose of this service is to set the desired execution state of a mission plan.

Table 28: MissionPlanExecutionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanExecution	reportMissionPlanExecutionCommandStatus
queryMissionPlanExecutionCommandAck⊕	reportMissionPlanExecutionCommandAck
cancelMissionPlanExecutionCommand⊕	reportMissionPlanExecutionCancelCommandStatus⊕

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

6.1.5.1 reportMissionPlanExecutionCommandAck

Description: This operation is used to provide the MissionPlanExecution commanded values.

Namespace: UMAA::MM::MissionPlanExecutionControl

Topic: MissionPlanExecutionCommandAckReportType

Data Type: MissionPlanExecutionCommandAckReportType

Table 29: MissionPlanExecutionCommandAckReportType Message Definition

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

6.1.5.2 reportMissionPlanExecutionCommandStatus

Description: This operation is used to report the current status of executing a mission plan execution command.

Namespace: UMAA::MM::MissionPlanExecutionControl

Topic: MissionPlanExecutionCommandStatusType

Data Type: MissionPlanExecutionCommandStatusType

Table 30: MissionPlanExecutionCommandStatusType Message Definition

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

6.1.5.3 setMissionPlanExecution

Description: This operation is used to set the current values of a mission plan execution command.

Namespace: UMAA::MM::MissionPlanExecutionControl

Topic: MissionPlanExecutionCommandType

Data Type: MissionPlanExecutionCommandType

Table 31: MissionPlanExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionID†	NumericGUID	Specifies the identifier of the mission plan to command a state. If not included, then commands the state of all mission plans.
state	TaskControlEnumType	The commanded state of the mission plan.

6.1.6 MissionPlanExecutionStatus

The purpose of this service is to provide the current execution state of a mission plan.

Table 32: MissionPlanExecutionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMissionPlanExecution⊕	reportMissionPlanExecution

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

6.1.6.1 reportMissionPlanExecution

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

Namespace: UMAA::MM::MissionPlanExecutionStatus

Topic: MissionPlanExecutionReportType

Data Type: MissionPlanExecutionReportType

Table 33: MissionPlanExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
endTime†	DateTime	Provides the estimated (future time) or actual (past time) end time for the mission associated with missionID.
feedback	StringShortDescription	Provides a reason for the current state of the mission plan (e.g., why the mission plan failed).
missionPlanDescription	StringShortDescription	Provides the description of the mission plan.
name	StringShortDescription	Provides the name of the mission plan.
startTime†	DateTime	Provides the estimated (future time) or actual (past time) start time for the mission plan associated with missionID.
state	TaskStateEnumType	Provides the current state of the mission plan specified by the associated missionID.
missionID*	NumericGUID	An identification of the mission plan.

6.1.7 MissionPlanMissionControl

The purpose of this service is to manage missions for the mission plan.

Table 34: MissionPlanMissionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanMissionAdd	reportMissionPlanMissionAddCommandStatus
queryMissionPlanMissionAddCommandAck⊕	reportMissionPlanMissionAddCommandAck
cancelMissionPlanMissionAddCommand⊕	reportMissionPlanMissionAddCancelCommandStatus⊕
setMissionPlanMissionClear	reportMissionPlanMissionClearCommandStatus
queryMissionPlanMissionClearCommandAck⊕	reportMissionPlanMissionClearCommandAck
cancelMissionPlanMissionClearCommand⊕	reportMissionPlanMissionClearCancelCommandStatus⊕
setMissionPlanMissionDelete	reportMissionPlanMissionDeleteCommandStatus
queryMissionPlanMissionDeleteCommandAck⊕	reportMissionPlanMissionDeleteCommandAck
cancelMissionPlanMissionDeleteCommand⊕	reportMissionPlanMissionDeleteCancelCommandStatus⊕

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

6.1.7.1 reportMissionPlanMissionAddCommandAck

Description: This operation is used to provide the MissionPlanMissionAdd commanded values.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionAddCommandAckReportType

Data Type: MissionPlanMissionAddCommandAckReportType

Table 35: MissionPlanMissionAddCommandAckReportType Message Definition

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

6.1.7.2 reportMissionPlanMissionAddCommandStatus

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionAddCommandStatusType

Data Type: MissionPlanMissionAddCommandStatusType

Table 36: MissionPlanMissionAddCommandStatusType Message Definition

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

6.1.7.3 reportMissionPlanMissionClearCommandAck

Description: This operation is used to provide the MissionPlanMissionClear commanded values.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionClearCommandAckReportType

Data Type: MissionPlanMissionClearCommandAckReportType

Table 37: MissionPlanMissionClearCommandAckReportType Message Definition

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

Attribute Name	Attribute Type	Attribute Description
command	MissionPlanMissionClearCommandType	The source command.

6.1.7.4 reportMissionPlanMissionClearCommandStatus

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionClearCommandStatusType

Data Type: MissionPlanMissionClearCommandStatusType

Table 38: MissionPlanMissionClearCommandStatusType Message Definition

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

6.1.7.5 reportMissionPlanMissionDeleteCommandAck

Description: This operation is used to provide the MissionPlanMissionDelete commanded values.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionDeleteCommandAckReportType

Data Type: MissionPlanMissionDeleteCommandAckReportType

Table 39: MissionPlanMissionDeleteCommandAckReportType Message Definition

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

6.1.7.6 reportMissionPlanMissionDeleteCommandStatus

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionDeleteCommandStatusType

Data Type: MissionPlanMissionDeleteCommandStatusType

Table 40: MissionPlanMissionDeleteCommandStatusType Message Definition

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

6.1.7.7 setMissionPlanMissionAdd

Description: This operation adds a new mission to the mission plan.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionAddCommandType

Data Type: MissionPlanMissionAddCommandType

Table 41: MissionPlanMissionAddCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionPlan	MissionPlanType	Specifies the mission, which consists of task(s) and objective(s), that is to be added to the mission plan.

6.1.7.8 setMissionPlanMissionClear

Description: This operation is used to clear all mission plan-related information.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionClearCommandType

Data Type: MissionPlanMissionClearCommandType

Table 42: MissionPlanMissionClearCommandType Message Definition

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

Attribute Name	Attribute Type	Attribute Description
clearTime	DateTime	The time of the clear. All mission-related information (missions, tasks, objectives, constraints, and conditionals) with a reported timestamp before or equal to the clearTime shall be removed from the DDS bus (when under the control of the provider) or ignored.

6.1.7.9 setMissionPlanMissionDelete

Description: This operation deletes an existing mission from the mission plan.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionDeleteCommandType

Data Type: MissionPlanMissionDeleteCommandType

Table 43: MissionPlanMissionDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
missionID†	NumericGUID	Specifies the identifier of the mission that is to be deleted. If the identifier is not specified, all missions are to be deleted.

6.1.8 MissionPlanObjectiveControl

The purpose of this service is to manage objectives for the mission plan.

Table 44: MissionPlanObjectiveControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanObjectiveAdd	reportMissionPlanObjectiveAddCommandStatus
queryMissionPlanObjectiveAddCommandAck⊕	reportMissionPlanObjectiveAddCommandAck
cancelMissionPlanObjectiveAddCommand⊕	reportMissionPlanObjectiveAddCancelCommandStatus⊕
setMissionPlanObjectiveDelete	reportMissionPlanObjectiveDeleteCommandStatus
queryMissionPlanObjectiveDeleteCommandAck⊕	reportMissionPlanObjectiveDeleteCommandAck
cancelMissionPlanObjectiveDeleteCommand⊕	reportMissionPlanObjectiveDeleteCancelCommandStatus⊕

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

6.1.8.1 reportMissionPlanObjectiveAddCommandAck

Description: This operation is used to provide the MissionPlanObjectiveAdd commanded values.

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveAddCommandAckReportType

Data Type: MissionPlanObjectiveAddCommandAckReportType

Table 45: MissionPlanObjectiveAddCommandAckReportType Message Definition

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

6.1.8.2 reportMissionPlanObjectiveAddCommandStatus

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

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveAddCommandStatusType

Data Type: MissionPlanObjectiveAddCommandStatusType

Table 46: MissionPlanObjectiveAddCommandStatusType Message Definition

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

6.1.8.3 reportMissionPlanObjectiveDeleteCommandAck

Description: This operation is used to provide the MissionPlanObjectiveDelete commanded values.

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveDeleteCommandAckReportType

Data Type: MissionPlanObjectiveDeleteCommandAckReportType

Table 47: MissionPlanObjectiveDeleteCommandAckReportType Message Definition

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

6.1.8.4 reportMissionPlanObjectiveDeleteCommandStatus

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

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveDeleteCommandStatusType

Data Type: MissionPlanObjectiveDeleteCommandStatusType

Table 48: MissionPlanObjectiveDeleteCommandStatusType Message Definition

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

6.1.8.5 setMissionPlanObjectiveAdd

Description: This operation adds a new objective to the mission plan.

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveAddCommandType

Data Type: MissionPlanObjectiveAddCommandType

Table 49: MissionPlanObjectiveAddCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionID	NumericGUID	Specifies the missionID to which the objective should be added.
objective	ObjectiveType	An objective to be added to a task of a mission.
taskID	NumericGUID	Specifies the taskID to which the objective should be added.

6.1.8.6 setMissionPlanObjectiveDelete

Description: This operation deletes an existing objective from the mission plan.

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveDeleteCommandType

Data Type: MissionPlanObjectiveDeleteCommandType

Table 50: MissionPlanObjectiveDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
missionID	NumericGUID	Specifies the identifier of the mission containing the task.
objectiveID†	NumericGUID	Specifies the identifier of the objective that is to be deleted. If the identifier is not specified, all objectives for the given taskID in the given missionID are to be deleted.
taskID	NumericGUID	Specifies the identifier of the task containing the objective to be deleted.

6.1.9 MissionPlanReport

The purpose of this service is to provide one or more mission plan(s).

Table 51: MissionPlanReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMissionPlan⊕	reportMissionPlan

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

6.1.9.1 reportMissionPlan

Description: This operation is used to report the current mission plan.

Namespace: UMAA::MM::MissionPlanReport

Topic: MissionPlanReportType

Data Type: MissionPlanReportType

Table 52: MissionPlanReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
constraints→setID	LargeSet<ConstraintType>	Set of constraints for the mission plan(s). This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::MissionPlanReport::MissionPlanReportTypeConstraintsSetElement.
missionPlan→setID	LargeSet<MissionPlanType>	List of available mission plans. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::MissionPlanReport::MissionPlanReportTypeMissionPlanSetElement.

6.1.10 MissionPlanTaskControl

The purpose of this service is to manage tasks for the mission plan.

Table 53: MissionPlanTaskControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanTaskAdd	reportMissionPlanTaskAddCommandStatus
queryMissionPlanTaskAddCommandAck⊕	reportMissionPlanTaskAddCommandAck
cancelMissionPlanTaskAddCommand⊕	reportMissionPlanTaskAddCancelCommandStatus⊕
setMissionPlanTaskDelete	reportMissionPlanTaskDeleteCommandStatus
queryMissionPlanTaskDeleteCommandAck⊕	reportMissionPlanTaskDeleteCommandAck
cancelMissionPlanTaskDeleteCommand⊕	reportMissionPlanTaskDeleteCancelCommandStatus⊕

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

6.1.10.1 reportMissionPlanTaskAddCommandAck

Description: This operation is used to provide the MissionPlanTaskAdd commanded values.

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskAddCommandAckReportType

Data Type: MissionPlanTaskAddCommandAckReportType

Table 54: MissionPlanTaskAddCommandAckReportType Message Definition

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

6.1.10.2 reportMissionPlanTaskAddCommandStatus

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

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskAddCommandStatusType

Data Type: MissionPlanTaskAddCommandStatusType

Table 55: MissionPlanTaskAddCommandStatusType Message Definition

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

6.1.10.3 reportMissionPlanTaskDeleteCommandAck

Description: This operation is used to provide the MissionPlanTaskDelete commanded values.

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskDeleteCommandAckReportType

Data Type: MissionPlanTaskDeleteCommandAckReportType

Table 56: MissionPlanTaskDeleteCommandAckReportType Message Definition

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

6.1.10.4 reportMissionPlanTaskDeleteCommandStatus

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

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskDeleteCommandStatusType

Data Type: MissionPlanTaskDeleteCommandStatusType

Table 57: MissionPlanTaskDeleteCommandStatusType Message Definition

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

6.1.10.5 setMissionPlanTaskAdd

Description: This operation adds a new task to the mission plan.

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskAddCommandType

Data Type: MissionPlanTaskAddCommandType

Table 58: MissionPlanTaskAddCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionID	NumericGUID	Specifies the missionID to which the task should be added.
taskPlan	TaskPlanType	Specifies the task, which consists of objective(s), that is to be added to the mission plan.

6.1.10.6 setMissionPlanTaskDelete

Description: This operation deletes an existing task from the mission plan.

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskDeleteCommandType

Data Type: MissionPlanTaskDeleteCommandType

Table 59: MissionPlanTaskDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionID	NumericGUID	Specifies the identifier of the mission containing the task to be deleted.
taskID†	NumericGUID	Specifies the identifier of the task that is to be deleted. If the identifier is not specified, all tasks for the given missionID are to be deleted.

6.1.11 ObjectiveExecutionControl

The purpose of this service is to set the desired execution state of an objective.

Table 60: ObjectiveExecutionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setObjectiveExecution	reportObjectiveExecutionCommandStatus
queryObjectiveExecutionCommandAck⊕	reportObjectiveExecutionCommandAck
cancelObjectiveExecutionCommand⊕	reportObjectiveExecutionCancelCommandStatus⊕

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

6.1.11.1 reportObjectiveExecutionCommandAck

Description: This operation is used to provide the ObjectiveExecution commanded values.

Namespace: UMAA::MM::ObjectiveExecutionControl

Topic: ObjectiveExecutionCommandAckReportType

Data Type: ObjectiveExecutionCommandAckReportType

Table 61: ObjectiveExecutionCommandAckReportType Message Definition

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

6.1.11.2 reportObjectiveExecutionCommandStatus

Description: This operation is used to report the current status of executing the objective execution command.

Namespace: UMAA::MM::ObjectiveExecutionControl

Topic: ObjectiveExecutionCommandStatusType

Data Type: ObjectiveExecutionCommandStatusType

Table 62: ObjectiveExecutionCommandStatusType Message Definition

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

6.1.11.3 setObjectiveExecution

Description: This operation is used to set the current values of an objective execution command within a task plan of a mission plan.

Namespace: UMAA::MM::ObjectiveExecutionControl

Topic: ObjectiveExecutionCommandType

Data Type: ObjectiveExecutionCommandType

Table 63: ObjectiveExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
missionID	NumericGUID	Specifies the objective's mission identifier.
objectiveID†	NumericGUID	Specifies the identifier of the objective to command a state. If not included, then commands the state of all objectives within the task plan.
state	TaskControlEnumType	The commanded state of the objective.
taskID	NumericGUID	Specifies the objective's task identifier.

6.1.12 ObjectiveExecutionStatus

The purpose of this service is to provide the current execution state of an objective.

Table 64: ObjectiveExecutionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryObjectiveExecution⊕	reportObjectiveExecution

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

6.1.12.1 reportObjectiveExecution

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

Namespace: UMAA::MM::ObjectiveExecutionStatus

Topic: ObjectiveExecutionReportType

Data Type: ObjectiveExecutionReportType

Table 65: ObjectiveExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
childObjectiveIDs	sequence<NumericGUID> max size = 256	The current child objective IDs associated with this objective.
endTime†	DateTime	Provides the estimated (future time) or actual (past time) end time for the objective associated with missionID, taskID, objectiveID.
feedback	StringShortDescription	Provides a reason for the current state of the objective (e.g., why the objective failed).
startTime†	DateTime	Provides the estimated (future time) or actual (past time) start time for the objective associated with missionID, taskID, objectiveID.
state	TaskStateEnumType	Provides the current state of the objective specified by the associated objective.
missionID*	NumericGUID	An identification of the mission.
objectiveID*	NumericGUID	Identifies the associated objective within a task plan of a mission plan.
taskID*	NumericGUID	An identification of the associated task plan within the mission plan.

6.1.13 TaskPlanExecutionControl

The purpose of this service is to set the desired execution state of the task plan.

Table 66: TaskPlanExecutionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setTaskPlanExecution	reportTaskPlanExecutionCommandStatus
queryTaskPlanExecutionCommandAck⊕	reportTaskPlanExecutionCommandAck
cancelTaskPlanExecutionCommand⊕	reportTaskPlanExecutionCancelCommandStatus⊕

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

6.1.13.1 reportTaskPlanExecutionCommandAck

Description: This operation is used to provide the TaskPlanExecution commanded values.

Namespace: UMAA::MM::TaskPlanExecutionControl

Topic: TaskPlanExecutionCommandAckReportType

Data Type: TaskPlanExecutionCommandAckReportType

Table 67: TaskPlanExecutionCommandAckReportType Message Definition

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

6.1.13.2 reportTaskPlanExecutionCommandStatus

Description: This operation is used to report the current status of executing the task plan execution command.

Namespace: UMAA::MM::TaskPlanExecutionControl

Topic: TaskPlanExecutionCommandStatusType

Data Type: TaskPlanExecutionCommandStatusType

Table 68: TaskPlanExecutionCommandStatusType Message Definition

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

6.1.13.3 setTaskPlanExecution

Description: This operation is used to set the current values of a task plan execution command for a mission plan.

Namespace: UMAA::MM::TaskPlanExecutionControl

Topic: TaskPlanExecutionCommandType

Data Type: TaskPlanExecutionCommandType

Table 69: TaskPlanExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionID	NumericGUID	Specifies the task plan's mission identifier.
state	TaskControlEnumType	The commanded state of the task plan.
taskID†	NumericGUID	Specifies the identifier of the task plan to command a state. If not included, then commands the state of all task plans within the mission plan.

6.1.14 TaskPlanExecutionStatus

The purpose of this service is to provide the current execution state of the task plan.

Table 70: TaskPlanExecutionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryTaskPlanExecution⊕	reportTaskPlanExecution

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

6.1.14.1 reportTaskPlanExecution

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

Namespace: UMAA::MM::TaskPlanExecutionStatus

Topic: TaskPlanExecutionReportType

Data Type: TaskPlanExecutionReportType

Table 71: TaskPlanExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
endTime†	DateTime	Specifies the estimated (future time) or actual (past time) end time for the task plan associated with missionID, taskID.
feedback	StringShortDescription	Provides a reason for the current state of the task plan (e.g., why the task plan failed).
startTime†	DateTime	Specifies the estimated (future time) or actual (past time) start time for the task plan associated with missionID, taskID.
state	TaskStateEnumType	Specifies the current state of the task plan specified by the associated missionID and taskID.
missionID*	NumericGUID	An identification of the mission plan.
taskID*	NumericGUID	An identification of the task plan within the mission plan.

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 72: 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 73: 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 74: 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 75: 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 76: 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 77: 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 78: 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 79: 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 80: 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 81: 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 82: 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 83: 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 84: 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 85: 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 86: 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 87: 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 88: 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 89: 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 90: 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 91: 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 92: 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 93: 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 94: 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 95: 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 96: 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 97: 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 98: 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 99: 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 100: 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 AnnulusSectorRequirementType

Namespace: UMAA::MM::BaseType::AnnulusSectorRequirementType

Description: A requirement that specifies the area of the annulus sector.

Table 101: AnnulusSectorRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
annulusSectorTolerance†	AnnulusSectorToleranceType	Specifies the tolerance for the annulus sector.
maxRange	Distance	Maximum range of the annulus sector.
minRange	Distance	Minimum range of the annulus sector.
sector	BearingSectorVariantType	Specifies the bearing sector.

6.2.31 AnnulusSectorToleranceType

Namespace: UMAA::MM::BaseType::AnnulusSectorToleranceType

Description: Defines the annulus sector tolerance.

Table 102: AnnulusSectorToleranceType 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 amount of error in position allowed from the annulus sector.

6.2.32 AreaRandomWalkObjectiveType

Namespace: UMAA::MM::BaseType::AreaRandomWalkObjectiveType

Description: The goal of the area random walk objective is to execute a random walk maneuver within a given area. This structure is used to specify the area where the random walk must be conducted. The area random walk objective is achieved by having the vehicle execute random vectors at a specified elevation (or current elevation if not specified) while maintaining the vehicle location within a defined area. The area is defined by specifying the vertices of a polygon, and the random vectors within this area can be configured by specifying min/max speeds and min/max time on course. Area and elevation include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::AreaRandomWalkObjectiveType

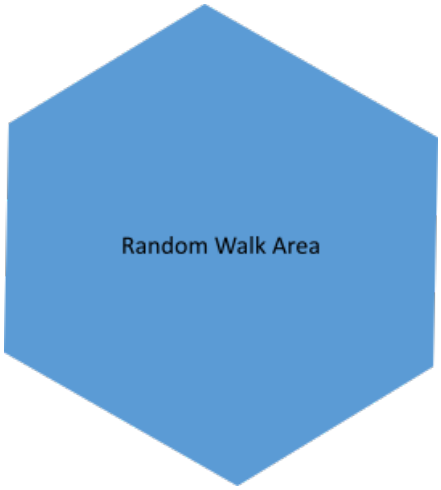

Figure 44: An Area Random Walk

Table 103: AreaRandomWalkObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
area	PolygonAreaRequirementType	Defines the area the vehicle must stay in while executing the random walk maneuver.
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the random walk portion of the objective. If not specified, duration is not used to determine when the random walk maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the random walk. If not specified, the maneuver is performed at the current elevation.
maxSpeed	SpeedVariantType	Defines the maximum vehicle speed on a given vector.
maxTimeOnCourse	DurationSeconds	Defines the maximum time spent on a given vector.
minSpeed	SpeedVariantType	Defines the minimum vehicle speed on a given vector.
minTimeOnCourse	DurationSeconds	Defines the minimum time spent on a given vector.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the area before transitioning to the random walk maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the random walk area location before transitioning to the random walk maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.

Attribute Name	Attribute Type	Attribute Description
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.33 BearingSectorGuideCourseVariantType

Namespace: UMAA::Common::Orientation::BearingSectorGuideCourseVariantType

Description: This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing. The bearing sector is defined relative to a guide (e.g., a contact) location with respect to the guide's course.

Table 104: BearingSectorGuideCourseVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endBearing	HeadingTarget	Provides the end bearing of the bearing sector. The endBearing is defined relative to a guide location with respect to the guide's course.
startBearing	HeadingTarget	Provides the start bearing of the bearing sector. The startBearing is defined relative to a guide location with respect to the guide's course.

6.2.34 BearingSectorMagneticNorthVariantType

Namespace: UMAA::Common::Orientation::BearingSectorMagneticNorthVariantType

Description: This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing. The bearing sector is defined relative to a guide (e.g., contact) location with respect to magnetic north.

Table 105: BearingSectorMagneticNorthVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endBearing	HeadingMagneticNorth	Provides the end bearing of the bearing sector. The endBearing is defined relative to a guide location with respect to magnetic north.
startBearing	HeadingMagneticNorth	Provides the start bearing of the bearing sector. The startBearing is defined relative to a guide location with respect to magnetic north.

6.2.35 BearingSectorTrueNorthVariantType

Namespace: UMAA::Common::Orientation::BearingSectorTrueNorthVariantType

Description: This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing. The bearing sector is defined relative to a guide (e.g., contact) location with respect to true north.

Table 106: BearingSectorTrueNorthVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endBearing	HeadingTrueNorthAngle	Provides the end bearing of the bearing sector. The endBearing is defined relative to a guide location with respect to true north.
startBearing	HeadingTrueNorthAngle	Provides the start bearing of the bearing sector. The startBearing is defined relative to a guide location with respect to true north.

6.2.36 BearingSectorVariantType

Namespace: UMAA::Common::Orientation::BearingSectorVariantType

Description: Union Type. This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing.

Table 107: BearingSectorVariantType Union(s)

Type Name	Type Description
BearingSectorGuideCourseVariantType	This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing. The bearing sector is defined relative to a guide (e.g., a contact) location with respect to the guide's course.
BearingSectorMagneticNorthVariantType	This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing. The bearing sector is defined relative to a guide (e.g., contact) location with respect to magnetic north.
BearingSectorTrueNorthVariantType	This structure defines a bearing sector, which is defined to be the sector created by rotating in a positive sense from the startBearing to the endBearing. The bearing sector is defined relative to a guide (e.g., contact) location with respect to true north.

6.2.37 CircleObjectiveType

Namespace: UMAA::MM::BaseType::CircleObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for following the circle pattern. The circle objective is achieved by having the vehicle execute the circle pattern maneuver at a specified elevation (or current elevation if not specified) as specified by the center position and radius, with the defined speed, trackTolerance, and turnDirection. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::CircleObjectiveType

Table 108: CircleObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the remaining pattern portion of the objective. If not specified, duration is not used to determine when the circle maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the circle maneuver. If not specified, the maneuver is performed at the current elevation.
loops†	SizeReal	Defines the number of loops around the circle pattern to execute. If not specified, the loops attribute is not used to determine when the circle maneuver is complete.
position†	GeoPosition2D	Defines the reference position for the circle pattern. If not specified, the reference position is the current vehicle position.
radius	Distance	Defines the radius for the circle pattern.
speed	SpeedRequirementVariantType	Defines the vehicle speed to maintain while executing the circle maneuver.
trackTolerance	DistanceRequirementType	Defines the maximum allowable cross track error while executing the circle maneuver.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the circle pattern location before transitioning to the circle maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the circle pattern location before transitioning to the circle maneuver.
turnDirection	WaterTurnDirectionEnumType	Defines the turn direction while executing the circle maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.38 ConditionalType

Namespace: UMAA::MM::Conditional::ConditionalType

Description: This structure defines common attributes across all conditionals.

Table 109: ConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalID	NumericGUID	Defines a unique identifier for the conditional.
name	StringShortDescription	Defines a short name for the conditional.
specializationID	NumericGUID	ID to capture specializations of ConditionalType.
specializationTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization.
specializationTopic	StringShortDescription	Topic to capture specializations of ConditionalType.

Table 110: ConditionalType Specialization(s)

Type Name	Type Description
ConstraintViolatedConditionalType	This structure defines a constraint violated conditional. The conditional is true when the conditional provided in the ConditionalReportType message, as specified by constraintConditionalID, is determined to be false.
DepthConditionalType	This structure defines a depth conditional. The conditional is true when the current depth, provided in GlobalPoseReportType, has the relationship specified in conditionalOp to the specified depth.
DepthRateConditionalType	This structure defines a depth rate conditional. The conditional is true when the current down speed (depthRate), provided in VelocityReportType, has the relationship specified in conditionalOp to the specified depthRate.
EmitterPresetConditionalType	This structure defines an emitter preset level conditional. The conditional is true when the current emitter preset levelID, provided in EmitterPresetReportType, is equal to the specified emitter preset levelID.
ExpConditionalType	This structure is used to define the an experimental conditional by specifying key/value pairs.
HeadingSectorConditionalType	This structure defines a heading sector conditional. The conditional is true when all heading sectors in the set are determined to be true; each heading sector is true when the vehicle yaw, provided by GlobalPoseReportType, is either inside or outside the defined sector as indicated by the headingSectorKind.
LogicalANDConditionalType	This structure defines a logical AND operator for a set of conditionals. The conditional is true when both conditionals referenced by conditionalID1 and conditionalID2 evaluate to true.
LogicalNOTConditionalType	This structure defines a logical NOT operator for a conditional. The conditional is true when the conditional referenced by notConditionalID evaluates to false.
LogicalORConditionalType	This structure defines a logical OR operator for a set of conditionals. The conditional is true when at least one of the conditionals referenced by conditionalID1 and conditionalID2 evaluate to true.
MissionStateConditionalType	This structure defines a mission state conditional. The conditional is true when the current state of the specified mission, provided by MissionPlanExecutionReportType, is equal to the defined missionState.
ObjectiveStateConditionalType	This structure defines an objective state conditional. The conditional is true when the current state of the specified objective, provided by ObjectiveExecutionReportType, is equal to the defined objectiveState.
PitchRateConditionalType	This structure defines a pitch rate conditional. The conditional is true when the current pitchRate, provided in VelocityReportType, has the relationship specified in conditionalOp to the specified pitchRate.

Type Name	Type Description
RelativeSpeedConditionalType	This structure defines a relative speed conditional. The conditional is true when the current speedThroughWater, provided in SpeedReportType, has the relationship specified in conditionalOp to the specified speed.
RollRateConditionalType	This structure defines a roll rate conditional. The conditional is true when the current rollRate, provided in VelocityReportType, has the relationship specified in conditionalOp to the specified rollRate.
SpeedConditionalType	This structure defines a speed conditional. The conditional is true when the current speedOverGround, provided in SpeedReportType, has the relationship specified in conditionalOp to the specified speed.
TaskStateConditionalType	This structure defines a task state conditional. The conditional is true when the current state of the specified task, provided by TaskPlanExecutionReportType, is equal to the defined taskState.
TimeConditionalType	This structure defines a time conditional. The conditional is true when current time has the specified relationship to the specified time.
WaterZoneConditionalType	This structure defines a water zone conditional. The conditional is true when all zones in the set are determined to be true; each zone is true when the vehicle location, provided by GlobalPoseReportType, is either inside or outside the defined volume as indicated by the zoneKind.
YawRateConditionalType	This structure defines a yaw rate conditional. The conditional is true when the current yawRate, provided in VelocityReportType, has the relationship specified in conditionalOp to the specified yawRate.

6.2.39 ConstraintType

Namespace: UMAA::MM::Constraint::ConstraintType

Description: This structure defines common attributes across all Constraints.

Table 111: ConstraintType Structure Definition

Attribute Name	Attribute Type	Attribute Description
constraintConditionalID	NumericGUID	Defines a unique identifier for the conditional.
constraintID	NumericGUID	Defines a unique identifier for the constraint.
name	StringShortDescription	Defines a short name for the constraint.
triggerConditionalID†	NumericGUID	Defines a unique identifier of the trigger that enables the constraint. If it is not defined, the trigger conditional is assumed to always be true.

6.2.40 ConstraintViolatedConditionalType

Namespace: UMAA::MM::Conditional::ConstraintViolatedConditionalType

Description: This structure defines a constraint violated conditional. The conditional is true when the conditional provided in the ConditionalReportType message, as specified by constraintConditionalID, is determined to be false. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::ConstraintViolatedConditionalType

Table 112: ConstraintViolatedConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
constraintConditionalID	NumericGUID	Defines the unique identifier of the constraint conditional.
duration†	DurationSeconds	Specifies how long the constraint needs to be violated before becoming true in order to allow the system to react to new constraints. This duration resets after achieving the constraint.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.41 DateTimeRequirementType

Namespace: UMAA::Common::Time::DateTimeRequirementType

Description: Defines a date time requirement.

Table 113: DateTimeRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
time	DateTime	Specifies the required time.
timeTolerance†	DateTimeToleranceType	Specifies the time tolerance.

6.2.42 DateTimeToleranceType

Namespace: UMAA::Common::Time::DateTimeToleranceType

Description: Defines the date time tolerance.

Table 114: DateTimeToleranceType 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	DateTime	Specifies the lower limit of allowable values.
upperlimit	DateTime	Specifies the upper limit of allowable values.

6.2.43 DepthConditionalType

Namespace: UMAA::MM::Conditional::DepthConditionalType

Description: This structure defines a depth conditional. The conditional is true when the current depth, provided in GlobalPoseReportType, has the relationship specified in conditionalOp to the specified depth. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::DepthConditionalType

Table 115: DepthConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current depth to the specified depth that must be met in order for the conditional to be true.
depth	DistanceBSL	Defines the value to compare with the current depth.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.44 DepthRateConditionalType

Namespace: UMAA::MM::Conditional::DepthRateConditionalType

Description: This structure defines a depth rate conditional. The conditional is true when the current down speed (depthRate), provided in VelocityReportType, has the relationship specified in conditionalOp to the specified depthRate. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::DepthRateConditionalType

Table 116: DepthRateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current down speed to the specified down speed that must be met in order for the conditional to be true.
depthRate	DownSpeed	Defines the value to compare with the current down speed.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.45 DepthRateRequirementType

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

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

Table 117: 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.

6.2.46 DepthRateRequirementVariantType

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

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

Table 118: DepthRateRequirementVariantType Structure Definition

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

6.2.47 DepthRateToleranceType

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

Description: Defines the depth rate tolerance.

Table 119: 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.

6.2.48 DepthRequirementType

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

Description: Defines the depth below sea level.

Table 120: 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.

6.2.49 DepthRequirementVariantType

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

Description: The depth below sea level.

Table 121: DepthRequirementVariantType Structure Definition

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

6.2.50 DepthToleranceType

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

Description: Defines the depth below sea level tolerance.

Table 122: 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.
upperlimit	DistanceBSL	Specifies the upper limit of allowable values for the depth below sea level.

6.2.51 DepthVariantType

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

Description: The depth below sea level.

Table 123: DepthVariantType Structure Definition

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

6.2.52 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 124: 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).

6.2.53 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 125: 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).

6.2.54 DirectionCurrentVariantType

Namespace: UMAA::Common::Orientation::DirectionCurrentVariantType

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 126: DirectionCurrentVariantType 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).

6.2.55 DirectionMagneticNorthRequirement

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

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

Table 127: 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.

6.2.56 DirectionMagneticNorthRequirementVariantType

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

Description: Specifies the direction with respect to magnetic north.

Table 128: DirectionMagneticNorthRequirementVariantType Structure Definition

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

6.2.57 DirectionMagneticNorthVariantType

Namespace: UMAA::Common::Orientation::DirectionMagneticNorthVariantType

Description: Specifies the direction with respect to magnetic north.

Table 129: DirectionMagneticNorthVariantType Structure Definition

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

6.2.58 DirectionRequirementVariantType

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

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

Table 130: 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.

6.2.59 DirectionToleranceType

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

Description: An angle tolerance associated with a direction.

Table 131: 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.

6.2.60 DirectionTrueNorthRequirement

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

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

Table 132: 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.

6.2.61 DirectionTrueNorthRequirementVariantType

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

Description: Specifies the direction with respect to true north.

Table 133: DirectionTrueNorthRequirementVariantType Structure Definition

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

6.2.62 DirectionTrueNorthVariantType

Namespace: UMAA::Common::Orientation::DirectionTrueNorthVariantType

Description: Specifies the direction with respect to true north.

Table 134: DirectionTrueNorthVariantType Structure Definition

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

6.2.63 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 135: DirectionTurnRateRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
directionRate	TurnRate	Specifies a change in direction as a function of time.

Attribute Name	Attribute Type	Attribute Description
directionRateTolerance†	DirectionTurnRateToleranceType	Specifies the direction turn rate tolerance.

6.2.64 DirectionTurnRateRequirementVariantType

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

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

Table 136: 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.

6.2.65 DirectionTurnRateToleranceType

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

Description: Defines the direction turn rate tolerance.

Table 137: 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.

6.2.66 DirectionVariantType

Namespace: UMAA::Common::Orientation::DirectionVariantType

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

Table 138: DirectionVariantType Union(s)

Type Name	Type Description
DirectionCurrentVariantType	Specifies the direction with respect to the current, where 0 is defined to be with the direction of the current (i.e. downstream).
DirectionMagneticNorthVariantType	Specifies the direction with respect to magnetic north.
DirectionTrueNorthVariantType	Specifies the direction with respect to true north.
DirectionWindVariantType	Specifies the direction with respect to the direction of the wind, where 0 is defined to be the direction into the wind.

6.2.67 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 139: 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.

6.2.68 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 140: 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.

6.2.69 DirectionWindVariantType

Namespace: UMAA::Common::Orientation::DirectionWindVariantType

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

Table 141: DirectionWindVariantType 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.

6.2.70 DistanceRequirementType

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

Description: Defines a distance requirement.

Table 142: DistanceRequirementType Structure Definition

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

6.2.71 DistanceToleranceType

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

Description: Defines the distance tolerance.

Table 143: 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.

6.2.72 DriftObjectiveType

Namespace: UMAA::MM::BaseType::DriftObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for drifting. The drift objective is achieved by having the vehicle, under a reduced power mode, maintain its position within the circle at a defined elevation (or current elevation if not defined) as specified by the reference position and driftRadius. If a position is not specified, then the current vehicle position is used as the reference position for drifting. DriftRadius and elevation include optional tolerances. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::DriftObjectiveType

Table 144: DriftObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
driftRadius	DistanceRequirementType	Defines the drift radius that specifies the maximum distance from the reference position the vehicle is allowed to drift.
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the drifting portion of the objective. If not specified, duration is not used to determine when drifting is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while drifting. If not specified, the maneuver is performed at the current elevation.
position†	GeoPosition2D	Defines the reference position for drifting. If not specified, the reference position is the current vehicle position.
speed	SpeedVariantType	Defines the desired vehicle speed when maneuvering within the area defined by driftRadius.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the drift location before transitioning to drifting. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the drift location before transitioning to drifting.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.73 ElevationRequirementVariantType

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

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

Table 145: 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.

6.2.74 ElevationVariantType

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

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

Table 146: 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.

6.2.75 EllipseVariantType

Namespace: UMAA::MM::BaseType::EllipseVariantType

Description: Defines an ellipse shape.

Table 147: EllipseVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
centerPosition	GeoPosition2D	Describes a reference point for the ellipse.
direction	HeadingTrueNorthAngle	Specifies the direction for the ellipse.
semiMajorRadius	Distance	Specifies the radius of the ellipse along the major axis.
semiMinorRadius	Distance	Specifies the radius of the ellipse along the minor axis.

6.2.76 EmitterPresetConditionalType

Namespace: UMAA::MM::Conditional::EmitterPresetConditionalType

Description: This structure defines an emitter preset level conditional. The conditional is true when the current emitter preset levelID, provided in EmitterPresetReportType, is equal to the specified emitter preset levelID. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::EmitterPresetConditionalType

Table 148: EmitterPresetConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
levelID	NumericGUID	Defines a unique identifier of the emitter preset level to compare with the current emitter preset level.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.77 EngineRPMSpeedRequirement

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

Description: Defines the engine rpm.

Table 149: 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.

6.2.78 EngineRPMSpeedRequirementVariantType

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

Description: Defines the engine RPM.

Table 150: EngineRPMSpeedRequirementVariantType Structure Definition

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

6.2.79 EngineRPMSpeedTolerance

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

Description: Defines the speed through engine rpm.

Table 151: 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.

6.2.80 EngineRPMSpeedVariantType

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

Description: Defines the engine RPM.

Table 152: EngineRPMSpeedVariantType Structure Definition

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

6.2.81 ExpBinaryValueType

Namespace: UMAA::MM::BaseType::ExpBinaryValueType

Description: This structure is used to define the binary value in a key/value pair.

Table 153: ExpBinaryValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
binaryValue	BinaryValue	Defines binary data for the value.

6.2.82 ExpBooleanValueType

Namespace: UMAA::MM::BaseType::ExpBooleanValueType

Description: This structure is used to define the boolean value in a key/value pair.

Table 154: ExpBooleanValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
booleanValue	boolean	Defines boolean data for the value.

6.2.83 ExpByteValueType

Namespace: UMAA::MM::BaseType::ExpByteValueType

Description: This structure is used to define the byte value in a key/value pair.

Table 155: ExpByteValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
byteValue	ByteValue	Defines byte data for the value.

6.2.84 ExpCharValueType

Namespace: UMAA::MM::BaseType::ExpCharValueType

Description: This structure is used to define the char value in a key/value pair.

Table 156: ExpCharValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
charValue	CharValue	Defines char data for the value.

6.2.85 ExpConditionalType

Namespace: UMAA::MM::Conditional::ExpConditionalType

Description: This structure is used to define the an experimental conditional by specifying key/value pairs. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::ExpConditionalType

Table 157: ExpConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
expConditionalName	StringShortDescription	Defines a short name for the experimental conditional.
keyValues	sequence<KeyValueType> max size = 170	Defines a set of key/value pairs for the experimental conditional.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.86 ExpDateTimeValueType

Namespace: UMAA::MM::BaseType::ExpDateTimeValueType

Description: This structure is used to define the DateTime value in a key/value pair.

Table 158: ExpDateTimeValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
dateTimeValue	DateTime	Defines DateTime data for the value.

6.2.87 ExpDoubleValueType

Namespace: UMAA::MM::BaseType::ExpDoubleValueType

Description: This structure is used to define the double value in a key/value pair.

Table 159: ExpDoubleValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
doubleValue	DoubleValue	Defines double data for the value.

6.2.88 ExpIntegerValueType

Namespace: UMAA::MM::BaseType::ExpIntegerValueType

Description: This structure is used to define the integer value in a key/value pair.

Table 160: ExpIntegerValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
integerValue	IntegerValue	Defines integer data for the value.

6.2.89 ExpLongLongValueType

Namespace: UMAA::MM::BaseType::ExpLongLongValueType

Description: This structure is used to define the long long value in a key/value pair.

Table 161: ExpLongLongValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
longlongValue	LargeCount	Defines long long data for the value.

6.2.90 ExpObjectiveType

Namespace: UMAA::MM::BaseType::ExpObjectiveType

Description: This structure is used to define the goal of an experimental objective by specifying key/value pairs. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::ExpObjectiveType

Table 162: ExpObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
expObjectiveDescription	StringShortDescription	Defines a short name for the experimental objective.

Attribute Name	Attribute Type	Attribute Description
keyValues	sequence<KeyValueType> max size = 170	Defines a set of key/value pairs for the experimental objective.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.91 ExpStringValueType

Namespace: UMAA::MM::BaseType::ExpStringValueType

Description: This structure is used to define the string value in a key/value pair.

Table 163: ExpStringValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
stringValue	StringValue	Defines string data for the value.

6.2.92 ExpValueType

Namespace: UMAA::MM::BaseType::ExpValueType

Description: Union Type. This structure is used to define the value in a key/value pair.

Table 164: ExpValueType Union(s)

Type Name	Type Description
ExpBinaryValueType	This structure is used to define the binary value in a key/value pair.
ExpBooleanValueType	This structure is used to define the boolean value in a key/value pair.
ExpByteValueType	This structure is used to define the byte value in a key/value pair.
ExpCharValueType	This structure is used to define the char value in a key/value pair.
ExpDateTimeValueType	This structure is used to define the DateTime value in a key/value pair.
ExpDoubleValueType	This structure is used to define the double value in a key/value pair.
ExpIntegerValueType	This structure is used to define the integer value in a key/value pair.
ExpLongLongValueType	This structure is used to define the long long value in a key/value pair.
ExpStringValueType	This structure is used to define the string value in a key/value pair.

6.2.93 Figure8ObjectiveType

Namespace: UMAA::MM::BaseType::Figure8ObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for following the figure 8 pattern. The figure 8 objective is achieved by having the vehicle execute the figure 8 maneuver at a defined elevation (or current elevation if not defined) as specified by the reference position, length, radius, and orientation, with the defined speed, trackTolerance, and turnDirection. If a reference position is not specified, then the current vehicle position is used as the reference position for the figure 8 pattern. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::Figure8ObjectiveType

Table 165: Figure8ObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the remaining pattern portion of the objective. If not specified, duration is not used to determine when the figure 8 maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the figure 8 maneuver. If not specified, the maneuver is performed at the current elevation.
length	Distance	Defines the length between the semicircles at either end for the figure 8 pattern.
loops†	SizeReal	Defines the number of loops around the figure 8 pattern to execute. If not specified, the loops attribute is not used to determine when the figure 8 maneuver is complete.
orientation	DirectionVariantType	Defines the orientation of the figure 8 pattern, measured perpendicular to the length axis.
position†	GeoPosition2D	Defines the reference position for the figure 8 pattern. If not specified, the reference position is the current vehicle position.
radius	Distance	Defines the radius of the semicircles for the figure 8 pattern.
speed	SpeedRequirementVariantType	Defines the vehicle speed to maintain while executing the figure 8 maneuver.
trackTolerance	DistanceRequirementType	Defines the maximum allowable cross track error while executing the figure 8 maneuver.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the figure 8 pattern location before transitioning to the figure 8 maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the figure 8 pattern location before transitioning to the figure 8 maneuver.

Attribute Name	Attribute Type	Attribute Description
turnDirection	WaterTurnDirectionEnumType	Defines the turn direction while executing the figure 8 maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.94 FreeFloatObjectiveType

Namespace: UMAA::MM::BaseType::FreeFloatObjectiveType

Description: The goal of the free float objective is to terminate all vehicle propulsion so that the vehicle is in a free float condition. The free float objective is achieved when all vehicle propulsion has been terminated. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::FreeFloatObjectiveType

Table 166: FreeFloatObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
duration†	DurationSeconds	Defines the desired duration to free float; if not specified, runs indefinitely until it is interrupted (e.g., another objective takes precedence, it is canceled, etc.).
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.95 GeoPosition2D

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

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

Table 167: 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.

6.2.96 GeoPosition2DRequirement

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

Description: Defines a position requirement.

Table 168: GeoPosition2DRequirement Structure Definition

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

6.2.97 GeoPosition2DTolerance

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

Description: Defines a position tolerance.

Table 169: 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.

6.2.98 GroundSpeedRequirement

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

Description: Defines the speed over ground.

Table 170: 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.

6.2.99 GroundSpeedRequirementVariantType

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

Description: Defines the speed over ground.

Table 171: GroundSpeedRequirementVariantType Structure Definition

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

6.2.100 GroundSpeedTolerance

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

Description: Defines the speed over ground tolerance.

Table 172: 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.

6.2.101 GroundSpeedVariantType

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

Description: Defines the speed over ground.

Table 173: GroundSpeedVariantType Structure Definition

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

6.2.102 HeadingSectorConditionalType

Namespace: UMAA::MM::Conditional::HeadingSectorConditionalType

Description: This structure defines a heading sector conditional. The conditional is true when all heading sectors in the set are determined to be true; each heading sector is true when the vehicle yaw, provided by GlobalPoseReportType, is either inside or outside the defined sector as indicated by the headingSectorKind. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::HeadingSectorConditionalType

Table 174: HeadingSectorConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
sector	sequence<HeadingSectorType> max size = 32	Defines the heading sector that is used to compare with the current yaw.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.103 HeadingSectorType

Namespace: UMAA::MM::Conditional::HeadingSectorType

Description: This structure defines a heading sector, a range of headings defined from startHeading to endHeading by rotating in a positive sense, that the vehicle must keep in or keep out.

Table 175: HeadingSectorType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endHeading	YawAngle	Defines the end heading of the defined sector.
headingSectorKind	HeadingSectorKindEnumType	Defines the type of heading sector, i.e., inside, outside.
startHeading	YawAngle	Defines the start heading of the defined sector.

6.2.104 HoverObjectiveType

Namespace: UMAA::MM::BaseType::HoverObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for hovering. The hover objective is achieved by having the vehicle actively maintain its position at a defined elevation (or current elevation if not defined) within the circle as defined by the reference position and hoverRadius, and optionally maintain a specified heading. If a position is not specified, then the current vehicle position is used as the reference position for hovering. If a heading is not specified, then the system is allowed to determine the best heading for hovering. Elevation, heading and hoverRadius include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::HoverObjectiveType

Table 176: HoverObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
controlPriority	HoverKindEnumType	Defines the priority to hover at the specified reference position.
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the hovering portion of the objective. If not specified, duration is not used to determine when hovering is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while hovering. If not specified, the maneuver is performed at the current elevation.
heading†	DirectionRequirementVariantType	Defines the heading for the vehicle to maintain while hovering. If not specified, the system will determine the best heading (e.g. current heading, into the wind/current, etc.) for hovering.
hoverRadius	DistanceRequirementType	Defines the maximum distance the vehicle position is allowed to be from the hover position and still considered to be achieved.
position†	GeoPosition2D	Defines the reference position for hovering. If not specified, the reference position is the current vehicle position.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the hover location before transitioning to hovering. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the hover location before transitioning to hovering.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.105 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 177: 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.

6.2.106 KeyValueType

Namespace: UMAA::MM::BaseType::KeyValueType

Description: This structure is used to define a key/value pair.

Table 178: KeyValueType Structure Definition

Attribute Name	Attribute Type	Attribute Description
key	StringName	Defines an identifier for the data contained in key/value pair.
value	ExpValueType	Defines the data contained in key/value pair.

6.2.107 LogicalANDConditionalType

Namespace: UMAA::MM::Conditional::LogicalANDConditionalType

Description: This structure defines a logical AND operator for a set of conditionals. The conditional is true when both conditionals referenced by conditionalID1 and conditionalID2 evaluate to true. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::LogicalANDConditionalType

Table 179: LogicalANDConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalID1	NumericGUID	Defines the first conditional to which the logical AND operation is applied.
conditionalID2	NumericGUID	Defines the second conditional to which the logical AND operation is applied.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.108 LogicalNOTConditionalType

Namespace: UMAA::MM::Conditional::LogicalNOTConditionalType

Description: This structure defines a logical NOT operator for a conditional. The conditional is true when the conditional referenced by notConditionalID evaluates to false. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::LogicalNOTConditionalType

Table 180: LogicalNOTConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
notConditionalID	NumericGUID	Defines the conditional to which the logical NOT operation is applied.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.109 LogicalORConditionalType

Namespace: UMAA::MM::Conditional::LogicalORConditionalType

Description: This structure defines a logical OR operator for a set of conditionals. The conditional is true when at least one of the conditionals referenced by conditionalID1 and conditionalID2 evaluate to true. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::LogicalORConditionalType

Table 181: LogicalORConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalID1	NumericGUID	Defines the first conditional to which the logical OR operation is applied.
conditionalID2	NumericGUID	Defines the second conditional to which the logical OR operation is applied.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.110 MissionPlanType

Namespace: UMAA::MM::BaseType::MissionPlanType

Description: This structure is used to report current mission plan(s).

Table 182: MissionPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
approvalRequired	boolean	An indication whether approval is required for the specified mission.
missionDescription	StringShortDescription	A description of the mission.
missionID	NumericGUID	Unique identifier for the mission.
missionPriority	Priority	Specifies the desired importance for completing the mission in order to handle the case where a plan cannot be generated to complete all missions. For this case, mission priority is used to determine what mission(s) to drop from the plan. Missions with the lowest priority must be dropped before missions with a higher priority. Mission priority is considered before task priority, meaning a low priority mission with a high priority task would be dropped before a high priority mission with a low priority task. If multiple missions have the lowest priority, then the order in which they are dropped is not defined by their priority and must be determined by some other mechanism.
name	StringShortDescription	A short name for the mission.
stateTrigger	sequence<StateTriggerType> max size = 16	Specifies the conditional statement that when true attempts to change the current state. Each trigger is evaluated individually, meaning if multiple triggers are defined with the same state, then their conditional statements are treated as if they are logically OR'd.
taskPlans→setID	LargeSet<TaskPlanType>	List of task plans associated with the mission. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::BaseType::MissionPlanTypeTaskPlansSetElement.

6.2.111 MissionStateConditionalType

Namespace: UMAA::MM::Conditional::MissionStateConditionalType

Description: This structure defines a mission state conditional. The conditional is true when the current state of the specified mission, provided by MissionPlanExecutionReportType, is equal to the defined missionState. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::MissionStateConditionalType

Table 183: MissionStateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
missionID	NumericGUID	Identifies the mission to be used in the conditional statement.
missionState	TaskStateEnumType	Specifies the state to compare with the current mission state.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.112 ObjectiveStateConditionalType

Namespace: UMAA::MM::Conditional::ObjectiveStateConditionalType

Description: This structure defines an objective state conditional. The conditional is true when the current state of the specified objective, provided by ObjectiveExecutionReportType, is equal to the defined objectiveState. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::ObjectiveStateConditionalType

Table 184: ObjectiveStateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
missionID	NumericGUID	Identifies the mission to be used in the conditional statement.
objectiveID	NumericGUID	Identifies the objective to be used in the conditional statement.
objectiveState	TaskStateEnumType	Specifies the state to compare with the current objective state.
taskID	NumericGUID	Identifies the task to be used in the conditional statement.

Attribute Name	Attribute Type	Attribute Description
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.113 ObjectiveType

Namespace: UMAA::MM::BaseType::ObjectiveType

Description: This is a base structure that all specialization objectives are inherited from. Each specialized objective structure shall be used to define or report its own specialized data.

Table 185: ObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
approvalRequired	boolean	An indication whether approval is required for the specified objective within a mission.
duringConditionID†	NumericGUID	A reference to a conditional that must be kept true during objective execution by completing/executing one or more actions. If not specified, then no duringCondition exists for the objective.
name	StringShortDescription	A short name for the objective.
objectiveDescription	StringShortDescription	A description of the objective.
objectiveID	NumericGUID	Unique identifier for the objective within a mission.
objectivePriority	Priority	Specifies the desired importance for completing the objective in order to handle the case where a plan cannot be generated to complete all objectives. For this case, objective priority is used to determine what objectives(s) to drop from the plan. Objectives with the lowest priority must be dropped before objectives with a higher priority. If multiple objectives have the lowest priority, then the order in which they are dropped is not defined by their priority and must be determined by some other mechanism.
preconditionID†	NumericGUID	A reference to a conditional that must be made true prior to executing the objective by completing one or more actions. If not specified, then no precondition exists for the objective.
preferredResourceID	sequence<IdentifierType> max size = 16	If defined, specifies a list of preferred resource(s) to execute the objective in order of preference. A preferred resource must be used if a valid plan can be generated. Otherwise, if a valid plan cannot be generated then another resource may be used (i.e., the inability to use a preferred resource when alternatives are available is not a cause for failure).

Attribute Name	Attribute Type	Attribute Description
stateTrigger	sequence<StateTriggerType> max size = 16	Specifies the conditional statement that when true attempts to change the current state. Each trigger is evaluated individually, meaning if multiple triggers are defined with the same state, then their conditional statements are treated as if they are logically OR'd.
specializationID	NumericGUID	ID to capture specializations of ObjectiveType.
specializationTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization.
specializationTopic	StringShortDescription	Topic to capture specializations of ObjectiveType.

Table 186: ObjectiveType Specialization(s)

Type Name	Type Description
FreeFloatObjectiveType	The goal of the free float objective is to terminate all vehicle propulsion so that the vehicle is in a free float condition. The free float objective is achieved when all vehicle propulsion has been terminated.
AreaRandomWalkObjectiveType	The goal of the area random walk objective is to execute a random walk maneuver within a given area. This structure is used to specify the area where the random walk must be conducted. The area random walk objective is achieved by having the vehicle execute random vectors at a specified elevation (or current elevation if not specified) while maintaining the vehicle location within a defined area. The area is defined by specifying the vertices of a polygon, and the random vectors within this area can be configured by specifying min/max speeds and min/max time on course. Area and elevation include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action.
CircleObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for following the circle pattern. The circle objective is achieved by having the vehicle execute the circle pattern maneuver at a specified elevation (or current elevation if not specified) as specified by the center position and radius, with the defined speed, trackTolerance, and turnDirection. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely.

Type Name	Type Description
DriftObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for drifting. The drift objective is achieved by having the vehicle, under a reduced power mode, maintain its position within the circle at a defined elevation (or current elevation if not defined) as specified by the reference position and driftRadius. If a position is not specified, then the current vehicle position is used as the reference position for drifting. DriftRadius and elevation include optional tolerances. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action.
ExpObjectiveType	This structure is used to define the goal of an experimental objective by specifying key/value pairs.
Figure8ObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for following the figure 8 pattern. The figure 8 objective is achieved by having the vehicle execute the figure 8 maneuver at a defined elevation (or current elevation if not defined) as specified by the reference position, length, radius, and orientation, with the defined speed, trackTolerance, and turnDirection. If a reference position is not specified, then the current vehicle position is used as the reference position for the figure 8 pattern. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely, or until interrupted by some other action.
HoverObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for hovering. The hover objective is achieved by having the vehicle actively maintain its position at a defined elevation (or current elevation if not defined) within the circle as defined by the reference position and hoverRadius, and optionally maintain a specified heading. If a position is not specified, then the current vehicle position is used as the reference position for hovering. If a heading is not specified, then the system is allowed to determine the best heading for hovering. Elevation, heading and hoverRadius include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action.

Type Name	Type Description
RacetrackObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for following the racetrack pattern. The racetrack objective is achieved by having the vehicle execute the racetrack maneuver at a defined elevation (or current elevation if not defined) as specified by the reference position, length, radius and orientation, with the defined speed, trackTolerance and turnDirection. If position is not specified, then the current vehicle position is used as the reference position for the racetrack pattern. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely, or until interrupted by some other action.
RegularPolygonObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for following the polygon pattern. The regular polygon objective is achieved by having the vehicle execute the regular polygon maneuver at a defined elevation (or current elevation if not defined) as specified by the reference position, diameter, and orientation, with the defined speed, trackTolerance, and turnDirection. If position is not specified, then the current vehicle position is used as the reference position for the regular polygon pattern. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely, or until interrupted by some other action.

Type Name	Type Description
RouteObjectiveType	This structure is used to report an element that describes a clearly defined goal specifying the action(s) required for following a route. The route objective is achieved by having the vehicle achieve each waypoint in order and is complete when the final waypoint is achieved. Each waypoint is specified by a position along with a captureRadius, and each waypoint optionally includes a specified attitude and elevation. If the attitude or elevation is not specified, then any attitude or elevation at the waypoint is acceptable, respectively. The desired effect when a waypoint is achieved is for the vehicle's position to be within a distance less than or equal to the captureRadius. The captureRadius includes an optional tolerance. If specified and the system is unable to achieve the captureRadius, then the waypoint can be considered completed if the vehicle's position is within a distance less than or equal to the specified tolerance. If the system is unable to complete the waypoint within the specified tolerance, then the objective is considered to have failed. If the captureRadius tolerance is not specified and the system is not able to achieve the captureRadius, then it is a "best effort" to achieve the waypoint, and is therefore not considered a cause for the objective to fail. If attitude or elevation is specified, then the desired effect when a waypoint is achieved includes the vehicle's attitude aligning with this specified attitude or obtaining the specified elevation, respectively, at the waypoint position along with the positional effect described above. Both attitude and elevation include an optional tolerance. If the system is unable to achieve the specified attitude or elevation within their tolerance, then the objective is considered to have failed. If a tolerance is not defined for attitude or elevation, then it is a "best effort" to achieve the specified attitude or elevation, respectively, and is therefore not considered a cause for the objective to fail. Additionally, there is an optional trackTolerance. If trackTolerance is specified, then the defined path between waypoints is specified to be the track line created by the waypoints with a cross track error less than or equal to the trackTolerance. The trackTolerance includes an optional tolerance. If specified, the vehicle is allowed to operate within this tolerance without having the objective fail. Otherwise, if the tolerance is violated, then the objective is considered to have failed. If the trackTolerance tolerance is not specified, then it is a "best effort" to maintain the specified trackTolerance, and is therefore not considered a cause for the objective to fail. If the trackTolerance is not specified, then path between waypoints is not specified and therefore left for the system to determine the best path.
ScreenRandomWalkObjectiveType	The goal of the screen random walk objective is to execute a random walk maneuver within a specified area that is relative to a guide (e.g., a high value unit (HVU)). This structure is used to specify the guide and the random walk area relative to the guide where the screening must be conducted. The screen random walk objective is achieved by having the vehicle execute random vectors at a defined elevation (or current elevation if not defined) while maintaining the vehicle position within a defined area relative to a guide. The area is defined by specifying a start and end bearing along with a min and max range. The random vectors can be configured by specifying min/max speeds and min/max time on course. Area and elevation include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action.

Type Name	Type Description
StationkeepObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for station keeping. The station keep objective is achieved by maintaining the vehicle location at a defined elevation (or current elevation if not defined) within a defined area relative to a guide. The area is defined by specifying a start and end bearing along with a min and max range. Area and elevation include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action.
VectorObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for the vector objective to achieve the specified speed, direction of travel, and altitude or depth of the vehicle. The vector objective is achieved by having the vehicle execute the vector maneuver at a defined elevation (or current elevation if not defined) as specified by direction, directionMode, and speed. The vector can be configured by setting an optional depthChangePitch. Direction, elevation, and speed include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action.

6.2.114 Orientation3DNEDRequirement

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

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

Table 187: 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.

6.2.115 PitchRateConditionalType

Namespace: UMAA::MM::Conditional::PitchRateConditionalType

Description: This structure defines a pitch rate conditional. The conditional is true when the current pitchRate, provided in VelocityReportType, has the relationship specified in conditionalOp to the specified pitchRate. This type is a

specialization of ConditionalType.

Topic: UMAA::MM::Conditional::PitchRateConditionalType

Table 188: PitchRateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current pitch rate to the specified pitch rate that must be met in order for the conditional to be true.
pitchRate	PitchRate	Defines the value to compare with the current pitchRate.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.116 PitchYNEDRequirement

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

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

Table 189: 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.

6.2.117 PitchYNEDTolerance

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

Description: A down or up angle tolerance.

Table 190: 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.

6.2.118 PitchYNEDType

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

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

Table 191: PitchYNEDType Structure Definition

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

6.2.119 Polygon

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

Description: Specifies an area defined by a polygon.

Table 192: Polygon Structure Definition

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

6.2.120 PolygonAreaRequirementType

Namespace: UMAA::MM::BaseType::PolygonAreaRequirementType

Description: A requirement that specifies the area of a polygon.

Table 193: PolygonAreaRequirementType Structure Definition

Attribute Name	Attribute Type	Attribute Description
area	Polygon	Specifies the area enclosed by the simple polygon.
areaTolerance†	PolygonAreaToleranceType	Specifies the tolerance for the polygon area.

6.2.121 PolygonAreaToleranceType

Namespace: UMAA::MM::BaseType::PolygonAreaToleranceType

Description: Defines the polygon area tolerance.

Table 194: PolygonAreaToleranceType 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 amount of error in position allowed from the polygon area.

6.2.122 PolygonVariantType

Namespace: UMAA::MM::BaseType::PolygonVariantType

Description: Defines a polygon shape.

Table 195: PolygonVariantType Structure Definition

Attribute Name	Attribute Type	Attribute Description
lineKind	LineSegmentEnumType	Describes the type of line used to represent the polygon.
referencePoints	sequence<GeoPosition2D> max size = 128	Describes reference points for the polygon.

6.2.123 RacetrackObjectiveType

Namespace: UMAA::MM::BaseType::RacetrackObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for following the racetrack pattern. The racetrack objective is achieved by having the vehicle execute the racetrack maneuver at a defined elevation (or current elevation if not defined) as specified by the reference position, length, radius and orientation, with the defined speed, trackTolerance and turnDirection. If position is not specified, then the current vehicle position is used as the reference position for the racetrack pattern. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::RacetrackObjectiveType

Table 196: RacetrackObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the remaining pattern portion of the objective. If not specified, duration is not used to determine when the racetrack maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the racetrack maneuver. If not specified, the maneuver is performed at the current elevation.
length	Distance	Defines the length between the semicircles at either end for the racetrack pattern.
loops†	SizeReal	Defines the number of loops around the racetrack pattern to execute; if not specified, the loops attribute is not used to determine when the racetrack maneuver is complete.
orientation	DirectionVariantType	Defines the orientation of the racetrack, measured perpendicular to the length axis.
position†	GeoPosition2D	Defines the reference position for the racetrack pattern. If not specified, the reference position is the current vehicle position.
radius	Distance	Defines the radius of the semicircles for the racetrack pattern.
speed	SpeedRequirementVariantType	Defines the vehicle speed to maintain while executing the racetrack maneuver.
trackTolerance	DistanceRequirementType	Defines the maximum allowable cross track error while executing the racetrack maneuver.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the racetrack pattern location before transitioning to the racetrack maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the racetrack pattern location before transitioning to the racetrack maneuver.

Attribute Name	Attribute Type	Attribute Description
turnDirection	WaterTurnDirectionEnumType	Defines the turn direction while executing the racetrack maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.124 RegularPolygonObjectiveType

Namespace: UMAA::MM::BaseType::RegularPolygonObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for following the polygon pattern. The regular polygon objective is achieved by having the vehicle execute the regular polygon maneuver at a defined elevation (or current elevation if not defined) as specified by the reference position, diameter, and orientation, with the defined speed, trackTolerance, and turnDirection. If position is not specified, then the current vehicle position is used as the reference position for the regular polygon pattern. Elevation, speed, and trackTolerance include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If both the duration attribute and the loops attribute are defined, complete the action(s) at whichever attribute condition completes first. If neither the duration attribute nor the loops attribute is specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::RegularPolygonObjectiveType

Table 197: RegularPolygonObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
diameter	Distance	Defines the diameter of a circumscribed circle around the polygon for the regular polygon pattern.
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the remaining pattern portion of the objective. If not specified, duration is not used to determine when the regular polygon maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the regular polygon maneuver. If not specified, the maneuver is performed at the current elevation.
loops†	SizeReal	Defines the number of loops around the regular polygon pattern to execute; if not specified, the loops attribute is not used to determine when the regular polygon maneuver is complete.
numberSides	SidesCount	Defines the number of sides for the regular polygon pattern.
orientation	DirectionVariantType	Defines the orientation of the regular polygon pattern, measured from the reference position of the polygon to one point on the polygon.

Attribute Name	Attribute Type	Attribute Description
position†	GeoPosition2D	Defines the reference position for the regular polygon pattern. If not specified, the reference position is the current vehicle position.
speed	SpeedRequirementVariantType	Defines the vehicle speed to maintain while executing the regular polygon maneuver.
trackTolerance	DistanceRequirementType	Defines the maximum allowable cross track error while executing the regular polygon maneuver.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the regular polygon pattern location before transitioning to the regular polygon maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the regular polygon pattern location before transitioning to the regular polygon maneuver.
turnDirection	WaterTurnDirectionEnumType	Defines the turn direction while executing the regular polygon maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.125 RelativeSpeedConditionalType

Namespace: UMAA::MM::Conditional::RelativeSpeedConditionalType

Description: This structure defines a relative speed conditional. The conditional is true when the current speedThroughWater, provided in SpeedReportType, has the relationship specified in conditionalOp to the specified speed. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::RelativeSpeedConditionalType

Table 198: RelativeSpeedConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current speed to the specified speed that must be met in order for the conditional to be true.
speed	SpeedLocalWaterMass	Defines the value to compare with the current speedThroughWater.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.126 RollRateConditionalType

Namespace: UMAA::MM::Conditional::RollRateConditionalType

Description: This structure defines a roll rate conditional. The conditional is true when the current rollRate, provided in VelocityReportType, has the relationship specified in conditionalOp to the specified rollRate. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::RollRateConditionalType

Table 199: RollRateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current roll rate to the specified roll rate that must be met in order for the conditional to be true.
rollRate	RollRate	Defines the value to compare with the current rollRate.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.127 RollXNEDRequirement

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

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

Table 200: 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.

6.2.128 RollXNEDTolerance

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

Description: A rotational tolerance.

Table 201: 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.

6.2.129 RollXNEDType

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

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

Table 202: RollXNEDType Structure Definition

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

6.2.130 RouteObjectiveType

Namespace: UMAA::MM::BaseType::RouteObjectiveType

Description: This structure is used to report an element that describes a clearly defined goal specifying the action(s) required for following a route. The route objective is achieved by having the vehicle achieve each waypoint in order and is complete when the final waypoint is achieved. Each waypoint is specified by a position along with a captureRadius, and each waypoint optionally includes a specified attitude and elevation. If the attitude or elevation is not specified, then any attitude or elevation at the waypoint is acceptable, respectively. The desired effect when a waypoint is achieved is for the vehicle's position to be within a distance less than or equal to the captureRadius. The captureRadius includes an optional tolerance. If specified and the system is unable to achieve the captureRadius, then the waypoint can be considered completed if the vehicle's position is within a distance less than or equal to the specified tolerance. If the system is unable to complete the waypoint within the specified tolerance, then the objective is considered to have failed. If the captureRadius tolerance is not specified and the system is not able to achieve the captureRadius, then it is a "best effort" to achieve the waypoint, and is therefore not considered a cause for the objective to fail. If attitude or elevation is specified, then the desired effect when a waypoint is achieved includes the vehicle's attitude aligning with this specified attitude or obtaining the specified elevation, respectively, at the waypoint position along with the positional effect described above. Both attitude and elevation include an optional tolerance. If the system is unable to achieve the specified attitude or elevation within their tolerance, then the objective is considered to have failed. If a tolerance is not defined for attitude or elevation, then it is a "best effort" to achieve the specified attitude or elevation, respectively, and is therefore not considered a cause for the objective to fail. Additionally, there is an optional trackTolerance. If trackTolerance is specified, then the defined path between waypoints is specified to be the track line created by the waypoints with a cross track error less than or equal to the trackTolerance. The trackTolerance includes an optional tolerance. If specified, the vehicle is allowed to operate within this tolerance without having the objective fail. Otherwise, if the tolerance is violated, then the objective is considered to have failed. If the trackTolerance tolerance is

not specified, then it is a "best effort" to maintain the specified trackTolerance, and is therefore not considered a cause for the objective to fail. If the trackTolerance is not specified, then path between waypoints is not specified and therefore left for the system to determine the best path. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::RouteObjectiveType

Table 203: RouteObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
routeDescription	StringShortDescription	Description of a route.
waypoints→listID	LargeList<WaypointType>	Specifies the route the vehicle is to travel. This attribute is implemented as a large list, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::BaseType::RouteObjectiveTypeWaypointsListElement.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.131 ScreenRandomWalkObjectiveType

Namespace: UMAA::MM::BaseType::ScreenRandomWalkObjectiveType

Description: The goal of the screen random walk objective is to execute a random walk maneuver within a specified area that is relative to a guide (e.g., a high value unit (HVU)). This structure is used to specify the guide and the random walk area relative to the guide where the screening must be conducted. The screen random walk objective is achieved by having the vehicle execute random vectors at a defined elevation (or current elevation if not defined) while maintaining the vehicle position within a defined area relative to a guide. The area is defined by specifying a start and end bearing along with a min and max range. The random vectors can be configured by specifying min/max speeds and min/max time on course. Area and elevation include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::ScreenRandomWalkObjectiveType

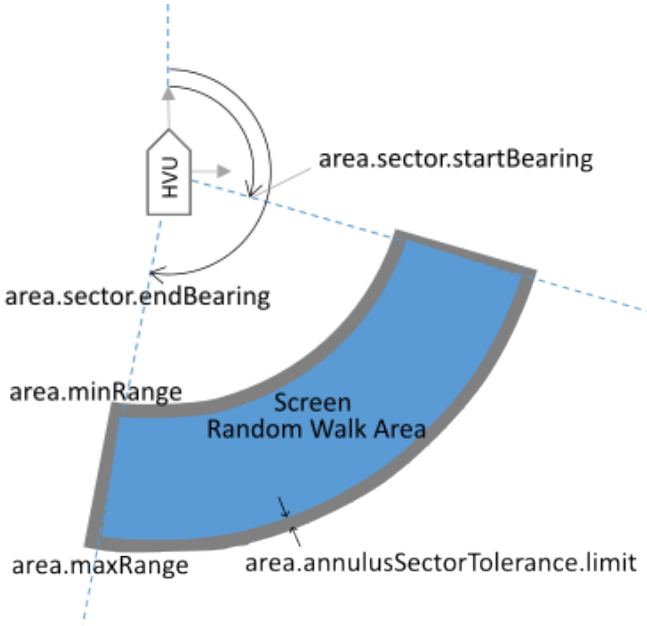

Figure 45: A Screen Random Walk

Table 204: ScreenRandomWalkObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
area	AnnulusSectorRequirementType	Defines the area the vehicle must stay in while executing the random walk maneuver.
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the random walk portion of the objective. If not specified, then duration is not used to determine when the random walk maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the random walk. If not specified, the maneuver is performed at the current elevation.
guideID	NumericGUID	Defines the ID of the guide relative to which the area for conducting the random walk is defined.
maxSpeed	SpeedVariantType	Defines the maximum vehicle speed on a given vector.
maxTimeOnCourse	DurationSeconds	Defines the maximum time spent on a given vector.
minSpeed	SpeedVariantType	Defines the minimum vehicle speed on a given vector.
minTimeOnCourse	DurationSeconds	Defines the minimum time spent on a given vector.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the random walk location before transitioning to the random walk maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the random walk location before transitioning to the random walk maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.132 ShapeVariantType

Namespace: UMAA::MM::BaseType::ShapeVariantType

Description: Union Type. This structure is used to describe the shape variant.

Table 205: ShapeVariantType Union(s)

Type Name	Type Description
EllipseVariantType	Defines an ellipse shape.
PolygonVariantType	Defines a polygon shape.

6.2.133 SpeedConditionalType

Namespace: UMAA::MM::Conditional::SpeedConditionalType

Description: This structure defines a speed conditional. The conditional is true when the current speedOverGround, provided in SpeedReportType, has the relationship specified in conditionalOp to the specified speed. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::SpeedConditionalType

Table 206: SpeedConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current speed to the specified speed that must be met in order for the conditional to be true.
speed	GroundSpeed	Defines the value to compare with the current speedOverGround.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.134 SpeedRequirementVariantType

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

Description: Union Type. Speed of the vehicle.

Table 207: 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.

6.2.135 SpeedVariantType

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

Description: Union Type. Speed of the vehicle.

Table 208: 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.

6.2.136 StateTriggerType

Namespace: UMAA::MM::BaseType::StateTriggerType

Description: This structure is used to specify a mechanism that attempts to initiate a planned state for a Mission Plan, Task Plan, or Objective when its defined conditional expression is determined to transition to logically true.

Table 209: StateTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalID	NumericGUID	Uniquely identifies the conditional.
count†	Count	Specifies the number of times the trigger can be used. If not included, assumed to be infinite.
state	TriggerStateEnumType	Specifies the state of the trigger.

6.2.137 StationkeepObjectiveType

Namespace: UMAA::MM::BaseType::StationkeepObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for station keeping. The station keep objective is achieved by maintaining the vehicle location at a defined elevation (or current elevation if not defined) within a defined area relative to a guide. The area is defined by specifying a start and end bearing along with a min and max range. Area and elevation include optional tolerances for their values. If specified, the vehicle is allowed to operate within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::StationkeepObjectiveType

Table 210: StationkeepObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
area	AnnulusSectorRequirementType	Defines the area the vehicle must stay in while executing the station keep maneuver.
duration†	DurationSeconds	After the transit portion of the objective is complete, defines the duration to execute the station keep portion of the objective. If not specified, then duration is not used to determine when the station keep maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the station keep maneuver. If not specified, the maneuver is performed at the current elevation.
guideID	NumericGUID	Defines the ID of the guide relative to which the area for conducting station keep is defined.
guideLostFailureDelay†	DurationSeconds	After the station keep objective is achieved (either initially or after an update), defines the amount of time to delay transitioning to a failed state when the system is unable to determine the guide's location. This measured time is reset each time the guide is tracked. If not defined, then a system configured delay time is used.
transitElevation†	ElevationVariantType	Defines the elevation used while transiting to the station keep location before transitioning to the station keep maneuver. If not specified, transit at the current elevation.
transitSpeed	SpeedVariantType	Defines the speed used while transiting to the station keep location before transitioning to the station keep maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.138 TaskPlanType

Namespace: UMAA::MM::BaseType::TaskPlanType

Description: This structure is used to define the attributes to specify a task plan. A task plan is a collection of logically related set of objectives.

Table 211: TaskPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
approvalRequired	boolean	An indication of whether approval is required for the specified task within a mission.
name	StringShortDescription	A short name for the task.
objectives→setID	LargeSet<ObjectiveType>	List of objectives associated with the task. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::BaseType::TaskPlanTypeObjectivesSetElement.
stateTrigger	sequence<StateTriggerType> max size = 16	Specifies the conditional statement that when true attempts to change the current state. Each trigger is evaluated individually, meaning if multiple triggers are defined with the same state, then their conditional statements are treated as if they are logically OR'd.
taskDescription	StringShortDescription	A description of the task.
taskID	NumericGUID	Unique identifier for the task within a mission.
taskPriority	Priority	Specifies the desired importance for completing the task in order to handle the case where a plan cannot be generated to complete all tasks. For this case, task priority is used to determine what task(s) to drop from the plan. Tasks with the lowest priority must be dropped before tasks with a higher priority. Task priority is considered before objective priority, meaning a low priority task with a high priority objective would be dropped before a high priority task with a low priority objective. If multiple tasks have the lowest priority, then the order in which they are dropped is not defined by their priority and must be determined by some other mechanism.

6.2.139 TaskStateConditionalType

Namespace: UMAA::MM::Conditional::TaskStateConditionalType

Description: This structure defines a task state conditional. The conditional is true when the current state of the specified task, provided by TaskPlanExecutionReportType, is equal to the defined taskState. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::TaskStateConditionalType

Table 212: TaskStateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
missionID	NumericGUID	Identifies the mission to be used in the conditional statement.
taskID	NumericGUID	Identifies the task to be used in the conditional statement.
taskState	TaskStateEnumType	Specifies the state to compare with the current task state.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.140 TimeConditionalType

Namespace: UMAA::MM::Conditional::TimeConditionalType

Description: This structure defines a time conditional. The conditional is true when current time has the specified relationship to the specified time. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::TimeConditionalType

Table 213: TimeConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current time to the specified time that must be met in order for the conditional to be true.
time	DateTime	Defines the time to which the current time will be compared.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.141 VectorObjectiveType

Namespace: UMAA::MM::BaseType::VectorObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for the vector objective to achieve the specified speed, direction of travel, and altitude or depth of the vehicle. The vector objective is achieved by having the vehicle execute the vector maneuver at a defined elevation (or current elevation if not defined) as specified by direction, directionMode, and speed. The vector can be configured by setting an optional depthChangePitch. Direction, elevation, and speed include optional tolerances for their values. If specified, the vehicle is allowed to operate

within these tolerances without having the objective fail. Otherwise, if any tolerances are violated after the objective is initially achieved, then the objective 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 objective to fail. If the duration attribute is not specified, the action(s) should continue indefinitely, or until interrupted by some other action. This type is a specialization of ObjectiveType.

Topic: UMAA::MM::BaseType::VectorObjectiveType

Table 214: VectorObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
depthChangePitch†	PitchYNEDType	Defines the desired angle of the vehicle when traversing to the requested elevation for UUVs.
direction	DirectionRequirementVariantType	Defines the vehicle direction to maintain while executing the vector maneuver.
directionMode	DirectionModeEnumType	Defines the direction mode while executing the vector maneuver.
duration†	DurationSeconds	Defines the duration to execute the global vector. If not specified, then duration is not used to determine when the vector maneuver is complete.
elevation†	ElevationRequirementVariantType	Defines the vehicle elevation to maintain while executing the vector maneuver. If not specified, the maneuver is performed at the current elevation.
speed	SpeedRequirementVariantType	Defines the vehicle speed to maintain while executing the vector maneuver.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ObjectiveType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ObjectiveType generalization.

6.2.142 VehicleSpeedModeRequirementVariantType

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

Description: Defines the speed mode.

Table 215: VehicleSpeedModeRequirementVariantType Structure Definition

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

6.2.143 VehicleSpeedModeVariantType

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

Description: Defines the speed mode.

Table 216: VehicleSpeedModeVariantType Structure Definition

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

6.2.144 WaterSpeedRequirement

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

Description: Defines the speed through water.

Table 217: 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.

6.2.145 WaterSpeedRequirementVariantType

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

Description: Defines the speed through water.

Table 218: WaterSpeedRequirementVariantType Structure Definition

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

6.2.146 WaterSpeedTolerance

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

Description: Defines the speed through water tolerance.

Table 219: 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.

6.2.147 WaterSpeedVariantType

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

Description: Defines the speed through water.

Table 220: WaterSpeedVariantType Structure Definition

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

6.2.148 WaterZoneConditionalType

Namespace: UMAA::MM::Conditional::WaterZoneConditionalType

Description: This structure defines a water zone conditional. The conditional is true when all zones in the set are determined to be true; each zone is true when the vehicle location, provided by GlobalPoseReportType, is either inside or outside the defined volume as indicated by the zoneKind. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::WaterZoneConditionalType

Table 221: WaterZoneConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
ceiling	ElevationVariantType	Describes the plane relative to the mean sea level that intersects the highest point or plane of the polygon.
floor	ElevationVariantType	Describes the plane relative to the mean sea level that intersects the lowest point or plane of the polygon.
zone	sequence<ShapeVariantType> max size = 16	Defines the zone that is used to compare with the current vehicle location.

Attribute Name	Attribute Type	Attribute Description
zoneKind	WaterZoneKindEnumType	Defines the type of zone, i.e., inside/outside
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.149 WaypointType

Namespace: UMAA::MM::BaseType::WaypointType

Description: This structure is used to define attributes of a waypoint including position, depth, and speed.

Table 222: WaypointType Structure Definition

Attribute Name	Attribute Type	Attribute Description
attitude†	Orientation3DNEDRequirement	Defines the attitude at the waypoint the vehicle must achieve. If not included, then the vehicle's attitude at the waypoint is left for the system to determine.
captureRadius	DistanceRequirementType	Defines a capture radius for the waypoint, which is the maximum distance the vehicle position is allowed to be from the waypoint position for it to be considered achieved.
elevation†	ElevationRequirementVariantType	Defines the elevation at the waypoint the vehicle must achieve. If not included, then the vehicle's elevation at the waypoint is left for the system to determine.
name†	StringShortDescription	A short name for the waypoint.
position	GeoPosition2D	Defines the position of the waypoint the vehicle must achieve.
speed†	SpeedVariantType	Defines the vehicle speed to maintain while executing the route. If not included, then the vehicle speed is left for the system to determine (e.g., in order to meet defined time constraint).
trackTolerance†	DistanceRequirementType	Defines the maximum allowable cross track error, where the previous waypoint (or the vehicle's current position when execution of the route objective begins for the first waypoint) is used to define a track line. If not included, then the path between waypoints is left for the system to determine.
waypointID	NumericGUID	A unique identification of the waypoint.

6.2.150 YawRateConditionalType

Namespace: UMAA::MM::Conditional::YawRateConditionalType

Description: This structure defines a yaw rate conditional. The conditional is true when the current yawRate, provided in VelocityReportType, has the relationship specified in conditionalOp to the specified yawRate. This type is a specialization of ConditionalType.

Topic: UMAA::MM::Conditional::YawRateConditionalType

Table 223: YawRateConditionalType Structure Definition

Attribute Name	Attribute Type	Attribute Description
conditionalOp	ConditionalOperatorEnumType	Defines the relationship of the current yaw rate to the specified yaw rate that must be met in order for the conditional to be true.
yawRate	YawRate	Defines the value to compare with the current yaw rate.
specializationReferenceTimestamp	DateTime	This field identifies the timestamp that signals the end of an atomic update to the instance of the specialization. NOTE: Ties this element back to the specializationID in ConditionalType generalization.
specializationReferenceID*	NumericGUID	NOTE: Ties this element back to the ConditionalType generalization.

6.2.151 YawZNEDRequirement

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

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

Table 224: 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.

6.2.152 YawZNEDTolerance

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

Description: A directional tolerance.

Table 225: 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.

6.2.153 YawZNEDType

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

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

Table 226: 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 227: 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 ConditionalOperatorEnumType

Namespace: UMAA::Common::MaritimeEnumeration::ConditionalOperatorEnumType

Description: A mutually exclusive set of values that defines a mathematical inequality.

Table 228: ConditionalOperatorEnumType Enumeration

Enumeration Value	Description
GREATER_THAN	One value is greater than another value.
GREATER_THAN_OR_EQUAL_TO	One value is greater than or equal to another value.
LESS_THAN	One value is less than another value.
LESS_THAN_OR_EQUAL_TO	One value is less than or equal to another value.

6.3.3 ContingencyBehaviorEnumType

Namespace: UMAA::Common::MaritimeEnumeration::ContingencyBehaviorEnumType

Description: A mutually exclusive set of values that defines the behavior of the vehicle used in case of emergency during the mission.

Table 229: ContingencyBehaviorEnumType Enumeration

Enumeration Value	Description
CONTINUE	Continue the mission
FINISH	Finish the mission
HOME	Return to home
LOITER	Loiter
NONE	None
VEHICLE_SPECIFIC	None of the above (specific to the vehicle)

6.3.4 DirectionModeEnumType

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

Description: Specifies whether direction is a course or heading.

Table 230: 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 HeadingSectorKindEnumType

Namespace: UMAA::Common::MaritimeEnumeration::HeadingSectorKindEnumType

Description: A mutually exclusive set of values that defines the heading sector kind.

Table 231: HeadingSectorKindEnumType Enumeration

Enumeration Value	Description
INSIDE	The heading sector kind is inside.
OUTSIDE	The heading sector kind is outside.

6.3.6 HoverKindEnumType

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

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

Table 232: HoverKindEnumType Enumeration

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

6.3.7 LineSegmentEnumType

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

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

Table 233: LineSegmentEnumType Enumeration

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

6.3.8 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 234: 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.

6.3.9 TaskControlEnumType

Namespace: UMAA::Common::MaritimeEnumeration::TaskControlEnumType

Description: An enumeration that is used to command the state of the mission plan, mission task, or mission objective.

Table 235: TaskControlEnumType Enumeration

Enumeration Value	Description
CANCEL	Cancel the mission plan, mission task, or mission objective.
EXECUTION_APPROVED	Approve the execution of the mission plan, mission task, or mission objective.
EXECUTION_NOT_APPROVED	Reject the execution of the mission plan, mission task, or mission objective.
PAUSE	Pause the execution of the approved mission plan, mission task, or mission objective.
PLAN	Plan the mission plan, mission task, or mission objective.
QUEUE	Queue the mission plan, mission task, or mission objective for execution.
RESTART	Restart the execution of the mission plan, mission task, or mission objective.
RESUME	Resume the execution of the mission plan, mission task, or mission objective.

6.3.10 TaskStateEnumType

Namespace: UMAA::Common::MaritimeEnumeration::TaskStateEnumType

Description: An enumeration that is used to report the state of the mission plan, a mission task, or mission objective.

Table 236: TaskStateEnumType Enumeration

Enumeration Value	Description
AWAITING_EXECUTION_APPROVAL	The mission plan, mission task, or mission objective is awaiting execution approval.
CANCELED	The mission plan, mission task, or mission objective has been cancelled.
CANCELING	The mission plan, mission task, or mission objective is in the process of being cancelled.
COMPLETED	The mission plan, mission task, or mission objective been completed. Collection tasks are considered complete when the resulting product is processed and disseminated. All other tasks are complete once the vehicle transitions from the executing state (vehicle releases weapon, stops jamming, etc.).
EXECUTING	The mission plan, mission task, or mission objective has begun execution (slews sensor and begins collect, begins to prepare weapons for release, starts jamming, etc.). This state defines the point of no return for a mission plan, mission task, or mission objective. Once transitioning to this state, the mission plan, mission task, or mission objective can no longer be reallocated to another UxS or vehicle unless it transitions to the FAILED state.
EXECUTION_APPROVED	The mission plan, mission task, or mission objective been approved for execution.
FAILED	The mission plan, mission task, or mission objective has failed. The UxS node has determined that no vehicles within the UxS can achieve the mission plan, mission task, or mission objective.

Enumeration Value	Description
NOT_PLANNED	The mission plan, mission task, or mission objective has not been planned.
NOT_QUEUED	The mission plan, mission task, or mission objective has not been queued for execution.
PAUSED	Used to pause the execution of an approved mission plan, approved mission task, or approved mission objective.
PAUSING	The mission plan, mission task, or mission objective is in the process of being paused.
PLANNED	The mission plan, mission task, or mission objective has been planned, indicating that it is part of an approved and active detailed mission plan.
PLANNING	The mission plan, mission task, or mission objective is still in the planning state.
QUEUED	The mission plan, mission task, or mission objective has been queued for execution.
QUEUING	The mission plan, mission task, or mission objective is being queued (e.g., uploading to vehicle) for execution.
RESTARTING	The mission plan, mission task, or mission objective is in the process of being restarted.
RESUMING	The mission plan, mission task, or mission objective is in the process of being resumed.

6.3.11 TriggerStateEnumType

Namespace: UMAA::Common::MaritimeEnumeration::TriggerStateEnumType

Description: Defines a mutually exclusive set of values that defines the trigger state.

Table 237: TriggerStateEnumType Enumeration

Enumeration Value	Description
CANCEL	A canceling state.
PAUSE	A pausing state.
PLAN	A planning state.
QUEUE	A queueing state.
RESTART	A restarting state.
RESUME	A resuming state.

6.3.12 VehicleSpeedModeEnumType

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

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

Table 238: VehicleSpeedModeEnumType Enumeration

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

6.3.13 WaterTurnDirectionEnumType

Namespace: UMAA::Common::MaritimeEnumeration::WaterTurnDirectionEnumType

Description: A mutually exclusive set of values that define the types of turn directions applied by the vehicle during turns.

Table 239: WaterTurnDirectionEnumType Enumeration

Enumeration Value	Description
LEFT_TURN	The vehicle will make left turns.
RIGHT_TURN	The vehicle will make right turns.

6.3.14 WaterZoneKindEnumType

Namespace: UMAA::Common::MaritimeEnumeration::WaterZoneKindEnumType

Description: Defines a mutually exclusive set of water zone kinds.

Table 240: WaterZoneKindEnumType Enumeration

Enumeration Value	Description
INSIDE	Defines a zone that the vehicle is required to stay inside.
OUTSIDE	Defines a zone that the vehicle is required to stay outside.

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 241: 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.
BinaryValue	octet[256]		Describes a binary value type.
BooleanEnumType	boolean		A mutually exclusive set of values that defines the truth values of logical algebra.
ByteValue	octet	maxInclusive=255 minInclusive=0	Describes a byte value type.
CharValue	char	maxInclusive=255 minInclusive=0	Describes a char value type.
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.
DoubleValue	double		Describes a double value type.

Type Name	Primitive Type	Range of Values	Description
DownSpeed	double	axisDirection=down axisUnit=MeterPerSecond maximumValue=299792458 minimumValue=-299792458 rangeMeaning=exact resolution=0.001 units=MeterPerSecond	Used for measuring speed and increases in magnitude as speed toward the center of the Earth increases.
DurationHours	double	maxInclusive=10505 minInclusive=0 units=Hour referenceFrame=Counting	Represents a time duration in hours.
DurationSeconds	double	maxInclusive=37817280 minInclusive=0 units=Seconds referenceFrame=Counting	Represents a time duration in seconds.
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.

Type Name	Primitive Type	Range of Values	Description
HeadingMagneticNorth	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 units=Radian referenceFrame=MagneticNorth	Heading as an angle specified with respect to Magnetic North.
HeadingTarget	double	maxInclusive=6.28318530718 minInclusive=-6.28318530718 units=Radian referenceFrame=Target	Describes heading with respect to the target.
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.
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.
IntegerValue	long		Describes an integer value type.
LargeCollectionSize	long	maxInclusive=2147483647 minInclusive=0	Specifies the size of a Large Collection.
LargeCount	uint64	maxInclusive=18446744073709551615 minInclusive=0 referenceFrame=Counting	Represents a count of elements.
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.
PitchRate	double	maxInclusive=32.767 minInclusive=-32.767 units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the platform's pitch angle.

Type Name	Primitive Type	Range of Values	Description
Priority	long	maxInclusive=255 minInclusive=0	Represents the priority as a positive integer. Low numbers represent low priority while higher numbers represent high priority.
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.
RollRate	double	maxInclusive=32.767 minInclusive=-32.767 units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the platform's roll angle.
SidesCount	long	maxInclusive=255 minInclusive=3	Represents the number of sides a polygon has using a positive integer.
SizeReal	double	units=None referenceFrame=Counting	Realizes SizeType: an entity that describes the magnitude or number of a measurable or countable entity.
Speed	double	maxInclusive=299792458 minInclusive=0 units=MeterPerSecond referenceFrame=Counting	This type stores speed in meters/s.
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.
StringName	string	length=64	Describes a 64 char string.
StringShortDescription	string	length=1023	Represents a short format description.
StringValue	string	length=256	Describes a 256 char string.
TurnRate	double	maxInclusive=32.767 minInclusive=-32.767 units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the heading angle of a platform.

Type Name	Primitive Type	Range of Values	Description
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.
YawRate	double	maxInclusive=32.767 minInclusive=-32.767 units=RadianPerSecond referenceFrame=Counting	Specifies the rate of change of the platform's yaw angle.

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