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

MDE Version 5.0.1

UMAA Spec Commit 63fb9f9

Version 5.2.1
20 April 2022

Contents

List of Figures

List of Tables

1 Scope

1.1 Identification

This document defines a set of services 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.

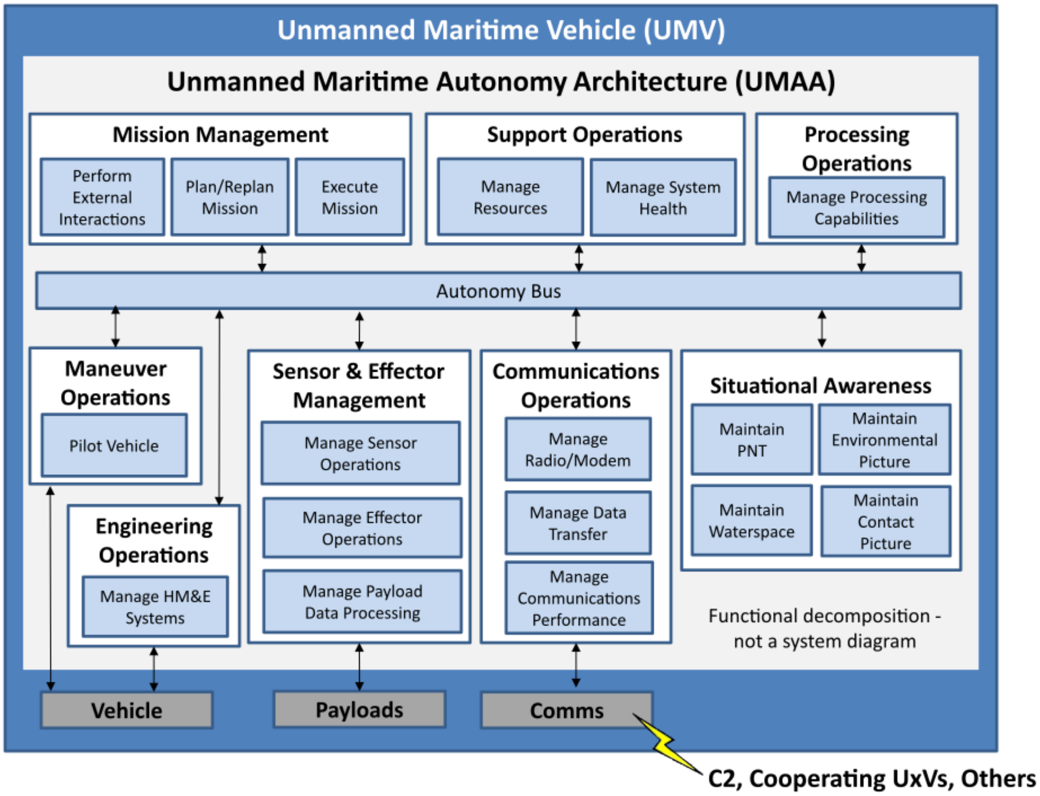

Unmanned Maritime Vehicle (UMV)
Unmanned Maritime Autonomy Architecture (UMAA)
Mission Management
Perform External Interactions
Plan/Replan Mission
Execute Mission
Support Operations
Manage Resources
Manage System Health
Processing Operations
Manage Processing Capabilities
Autonomy Bus
Maneuver Operations
Pilot Vehicle
Sensor & Effector Management
Manage Sensor Operations
Manage Effector Operations
Manage Payload Data Processing
Communications Operations
Manage Radio/Modem
Manage Data Transfer
Manage Communications Performance
Situational Awareness
Maintain PNT
Maintain Environmental Picture
Maintain Waterspace
Maintain Contact Picture
Engineering Operations
Manage HM&E Systems
Functional decomposition - not a system diagram
Vehicle
Payloads
Comms
C2, Cooperating UxVs, Others

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).

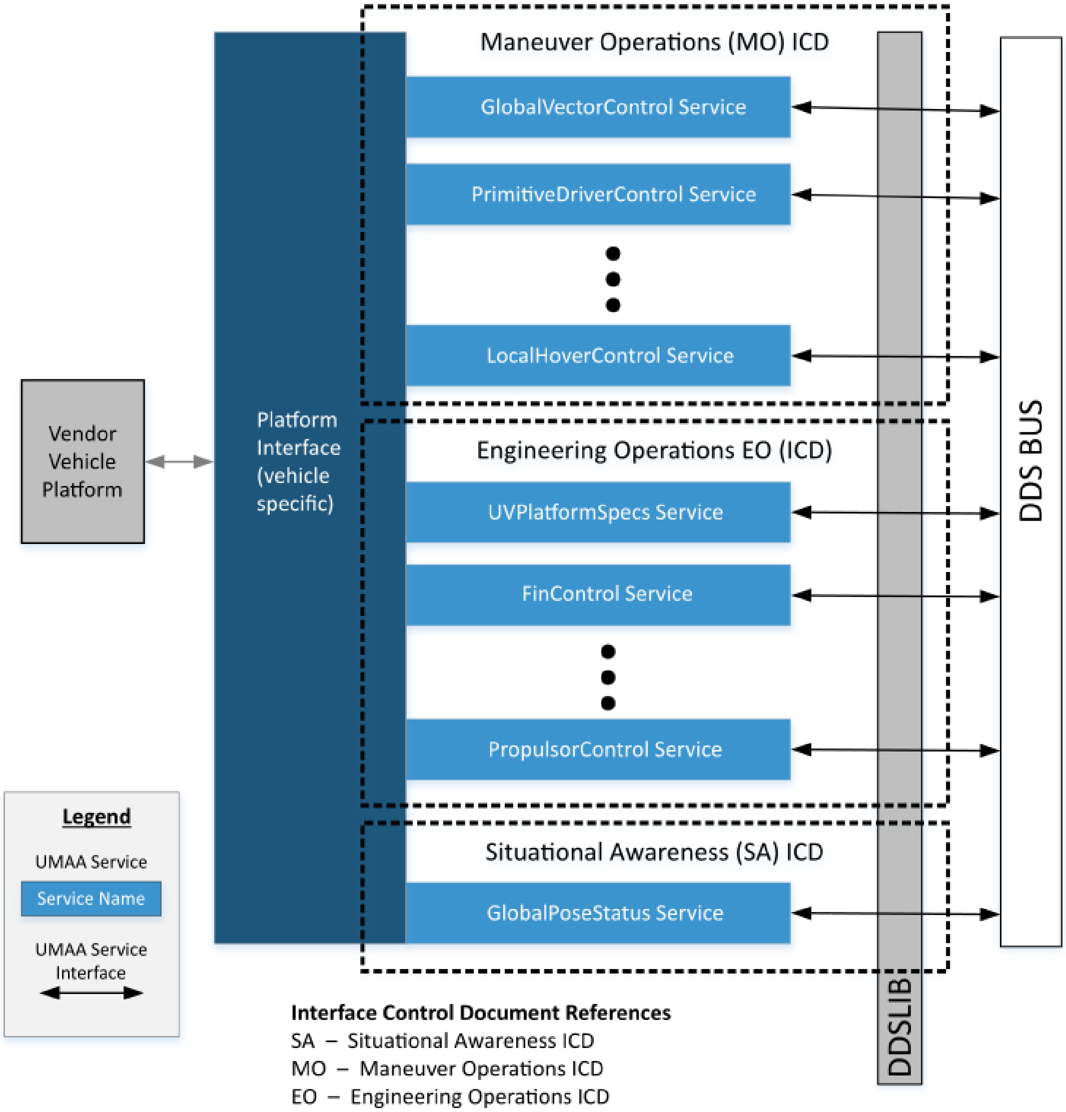

Maneuver Operations (MO) ICD
GlobalVectorControl Service
PrimitiveDriverControl Service
LocalHoverControl Service
Vendor Vehicle Platform
Platform Interface (vehicle specific)
Engineering Operations EO (ICD)
UVPlatformSpecs Service
FinControl Service
PropulsorControl Service
DDS BUS
Situational Awareness (SA) ICD
GlobalPoseStatus Service
DDSLIB
Legend
UMAA Service
Service Name
UMAA Service Interface
Interface Control Document References
SA – Situational Awareness ICD
MO – Maneuver Operations ICD
EO – Engineering Operations ICD

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 - Experimental (MM-EXP) 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^{TM}) modeling tool and are represented as UML^{TM} 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.

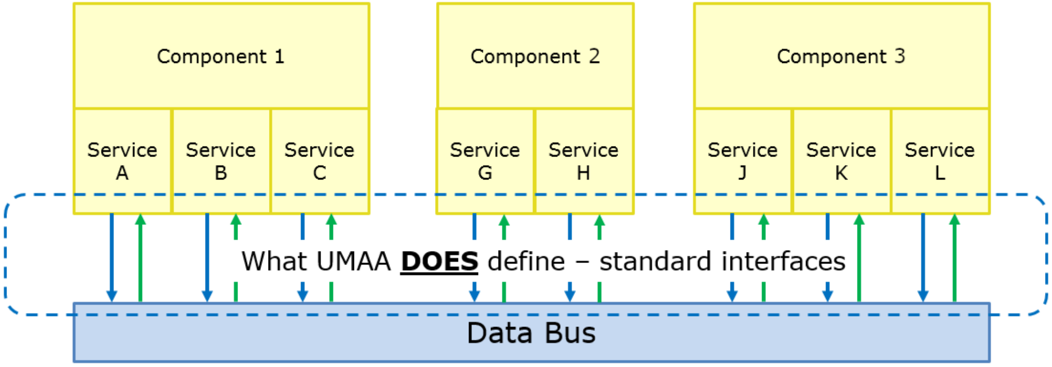


Figure 3: Services and Interfaces Exposed on the UMAA Data Bus.

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^{TM})

The data bus supporting autonomy messaging (as seen in Figure 3) is implemented via DDS^{TM}. 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 query<SBN>[FN]ConfigAck query<SBN>[FN]Config cancel<SBN>[FN]Config query<SBN>[FN]ConfigExecutionStatus	report<SBN>[FN]ConfigCommandStatus report<SBN>[FN]ConfigAck report<SBN>[FN]Config report<SBN>[FN]CancelConfigCommandStatus report<SBN>[FN]ConfigExecutionStatus
Control	set<SBN>[FN] query<SBN>[FN]CommandAck cancel<SBN>[FN]Command query<SBN>[FN]ExecutionStatus	report<SBN>[FN]CommandStatus report<SBN>[FN]CommandAck report<SBN>[FN]CancelCommandStatus report<SBN>[FN]ExecutionStatus
Specs	query<SBN>[FN]Specs	report<SBN>[FN]Specs
Status OR Report	query<SBN>[FN]	report<SBN>[FN]

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

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

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

where,

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

3.5 Namespace Conventions

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

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

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

UMAA::SA::ContactReport

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

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

3.6 Cybersecurity

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

3.7 GUID algorithm

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

3.8 Large Collections

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

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

3.8.1 Necessary QoS

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

3.8.2 Updating Large Collections

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

The set can be updated as a batch (multiple elements in a single "update cycle," as determined by the provider). This allows for a coarse synchronization: data elements that do not match the metadata `updateElementID` and `updateElementTimestamp` can be assumed to be part of an in-progress update cycle. Consumers can choose to immediately act on those data individually or wait until the matching element is received to signal that the complete update cycle has finished and consider the set as a whole. Note that the coarseness of synchronization is service-dependent: in some cases an intermediate view of a collection update may be logically incorrect to act upon.

3.8.3 Specifying an Empty Large Collection

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

3.8.4 Large Set Types

The following details the LargeSetMetadata structure:

Table 4: LargeSetMetadata Structure Definition

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

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

Table 5: Example FooReportTypeItemsSetElement Structure Definition

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

3.8.5 Large List Types

The following details the LargeListMetadata structure:

Table 6: LargeListMetadata Structure Definition

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

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

Table 7: Example FooReportTypeItemsListElement Structure Definition

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

4 Introduction to Coordinate Reference Frames and Position Model

4.1 Vehicle Reference Frame

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

The axes are defined as:

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

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

4.2 Earth-Centered Earth-Fixed Frame

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

4.3 North-East-Down Frame

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

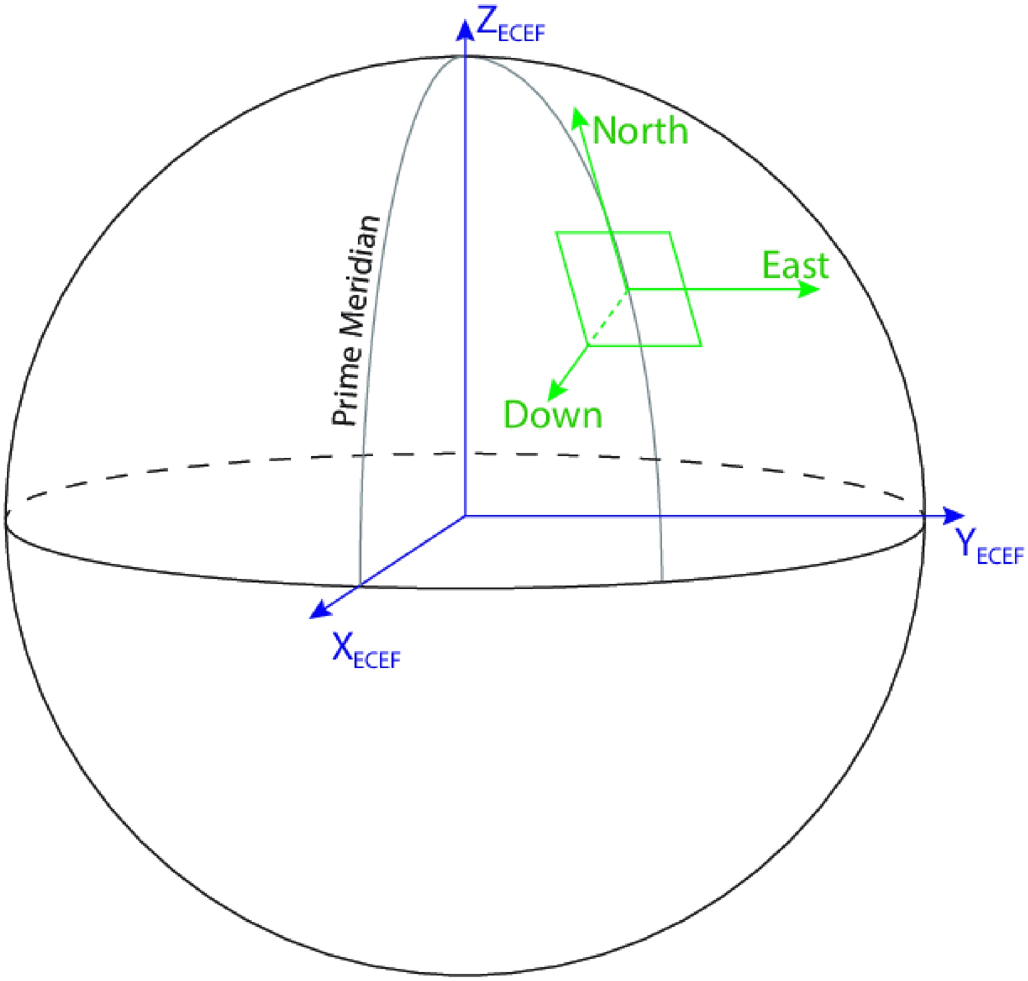


Figure 4: Origins and axes of the Earth-Centered Earth-Fixed (ECEF) and North-East-Down (NED) frames.

4.4 WGS 84

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

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

4.5 Vehicle Orientation

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

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

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

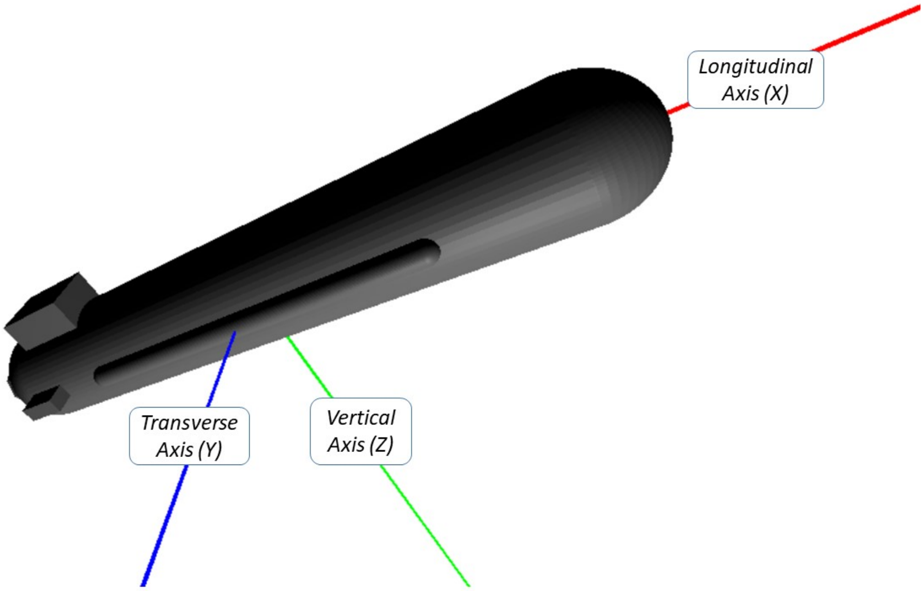


Figure 5: Define the Vehicle Coordinate System

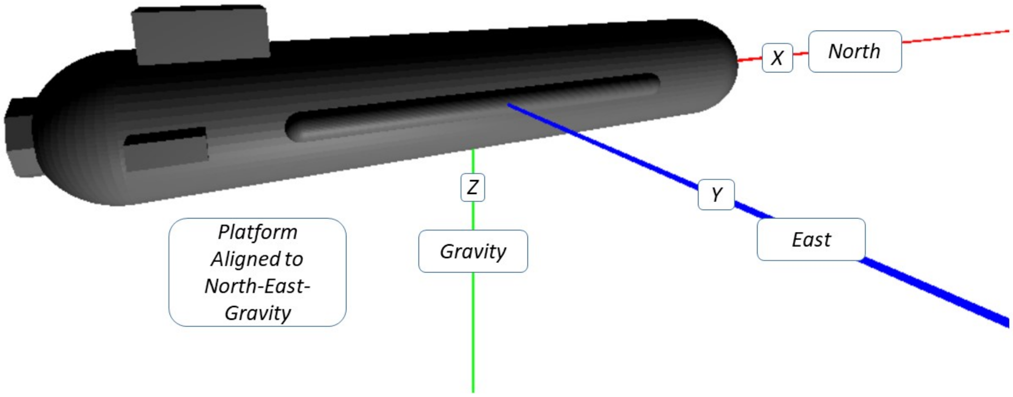


Figure 6: Align the Vehicle with the Reference Frame Axes.

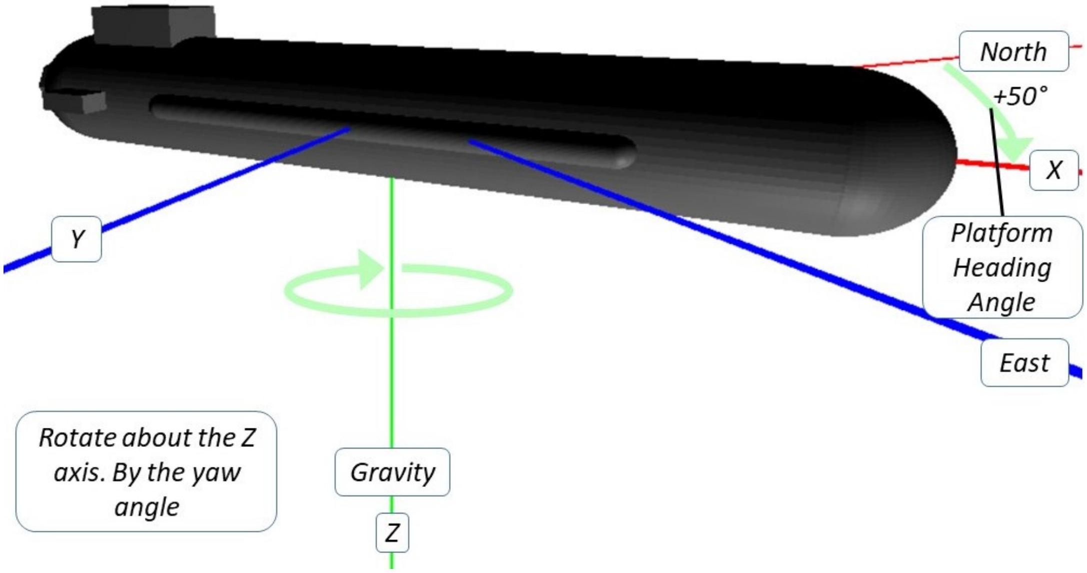


Figure 7: Rotate the Vehicle by the Yaw Angle.

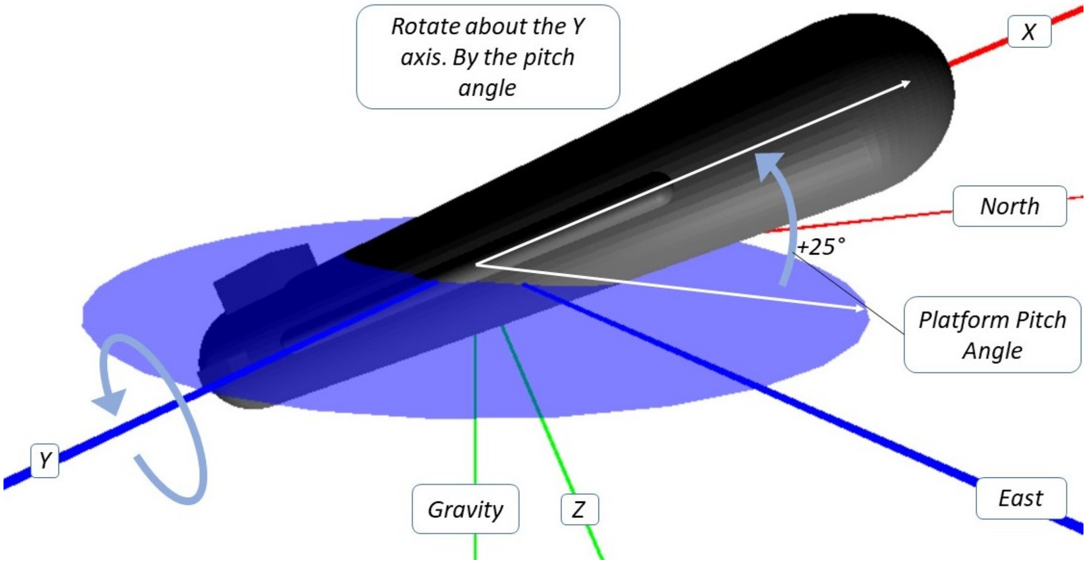


Figure 8: Rotate the Vehicle by the Pitch Angle.

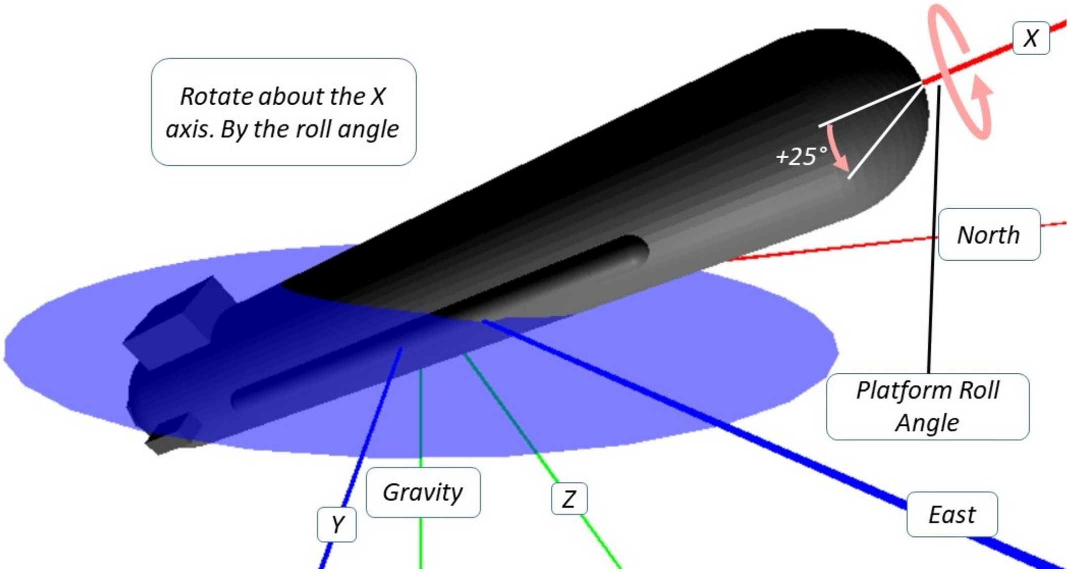


Figure 9: Rotate the Vehicle by the Roll Angle.

4.6 Vehicle Coordinate Reference Frame Origin

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

Definitions

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

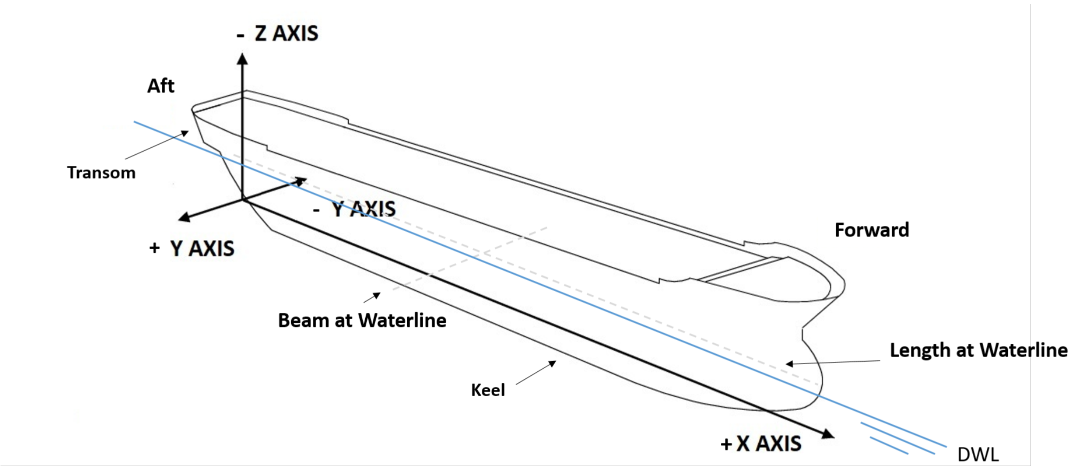


Figure 10: Keel Transom Intersection Origin Location on a USV as Example

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

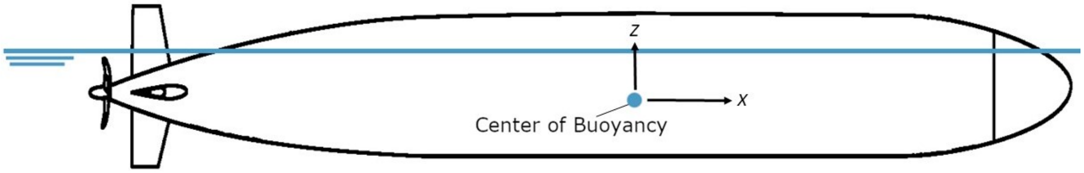


Figure 11: Center of Buoyancy Origin Location on a UUV as Example.

5 Flow Control

5.1 Command / Response

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

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

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

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

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

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

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

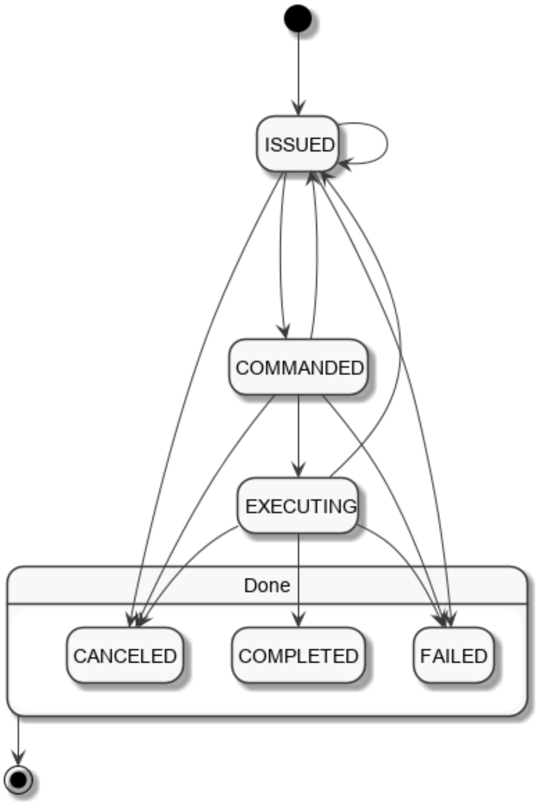


Figure 12: State transitions of the `commandStatus` as commands are processed.

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

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

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

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

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

5.1.1 High-Level Flow

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

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

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

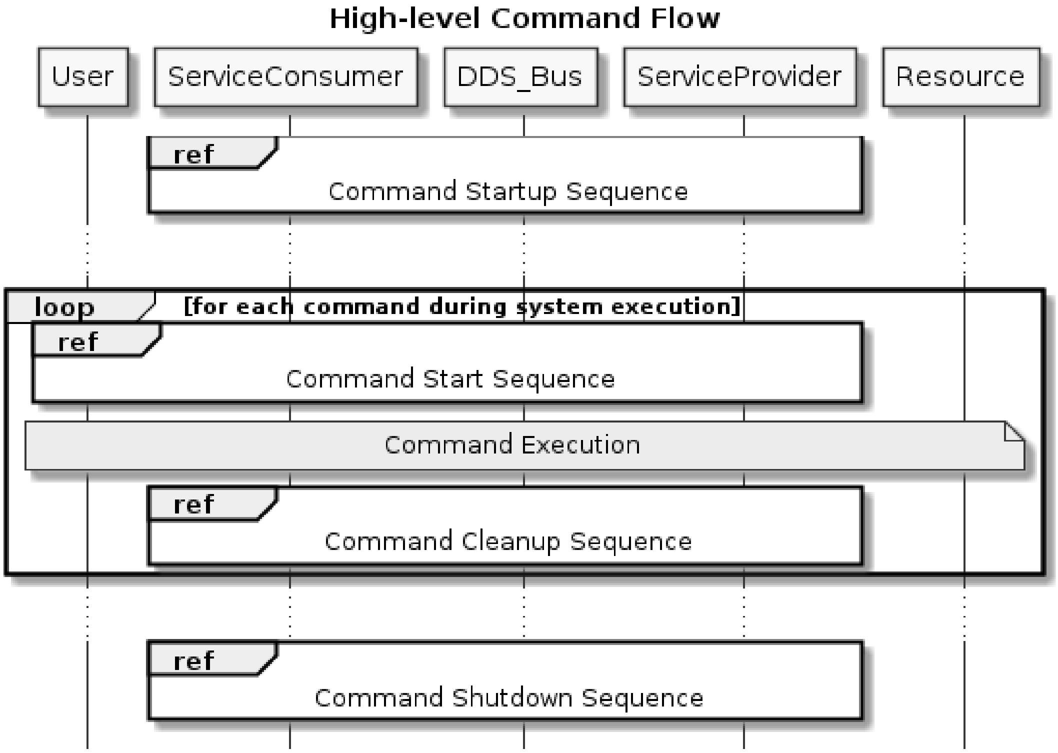


Figure 14: Sequence Diagram for the High-Level Description of a Command Execution.

5.1.2 Command Startup Sequence

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

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

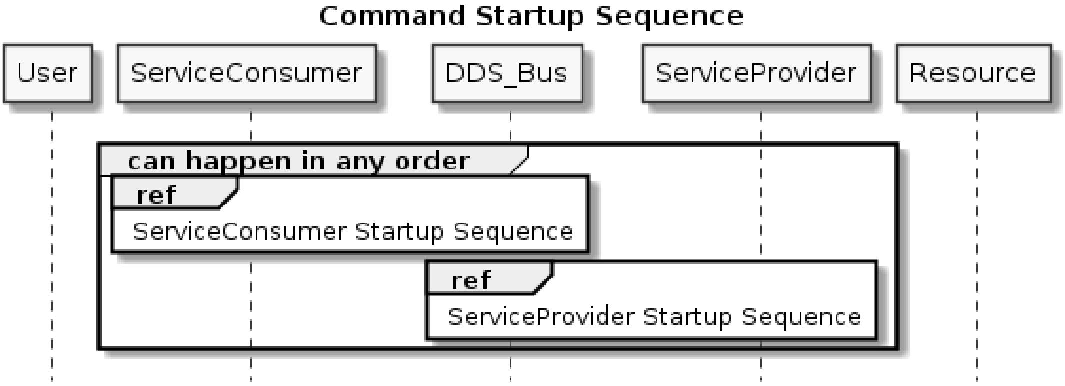


Figure 15: Sequence Diagram for Command Startup.

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

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

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

The Service Provider Startup sequence is shown in Figure 16.

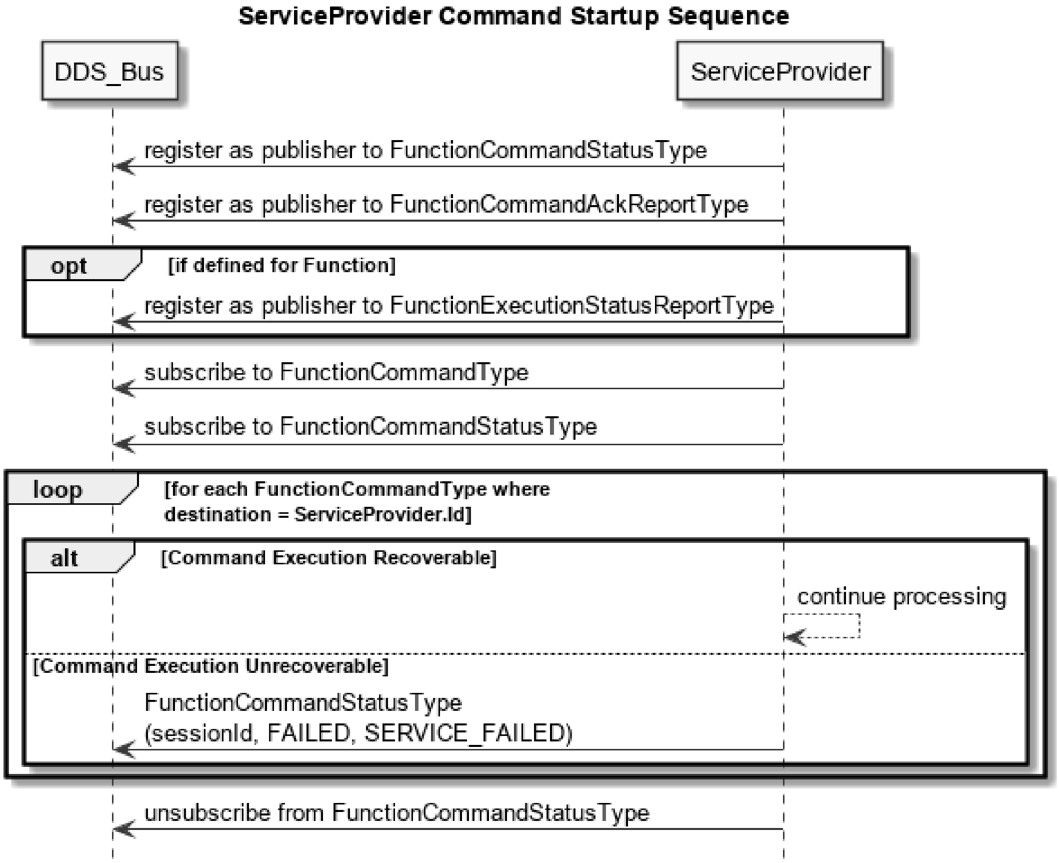

ServiceProvider Command Startup Sequence
DDS_Bus
ServiceProvider
register as publisher to FunctionCommandStatusType
register as publisher to FunctionCommandAckReportType
opt
[if defined for Function]
register as publisher to FunctionExecutionStatusReportType
subscribe to FunctionCommandType
subscribe to FunctionCommandStatusType
loop
[for each FunctionCommandType where destination = ServiceProvider.Id]
alt
[Command Execution Recoverable]
continue processing
[Command Execution Unrecoverable]
FunctionCommandStatusType (sessionId, FAILED, SERVICE_FAILED)
unsubscribe from FunctionCommandStatusType

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

5.1.2.2 Service Consumer Startup Sequence

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

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

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

The Service Consumer Startup sequence is shown in Figure 17.

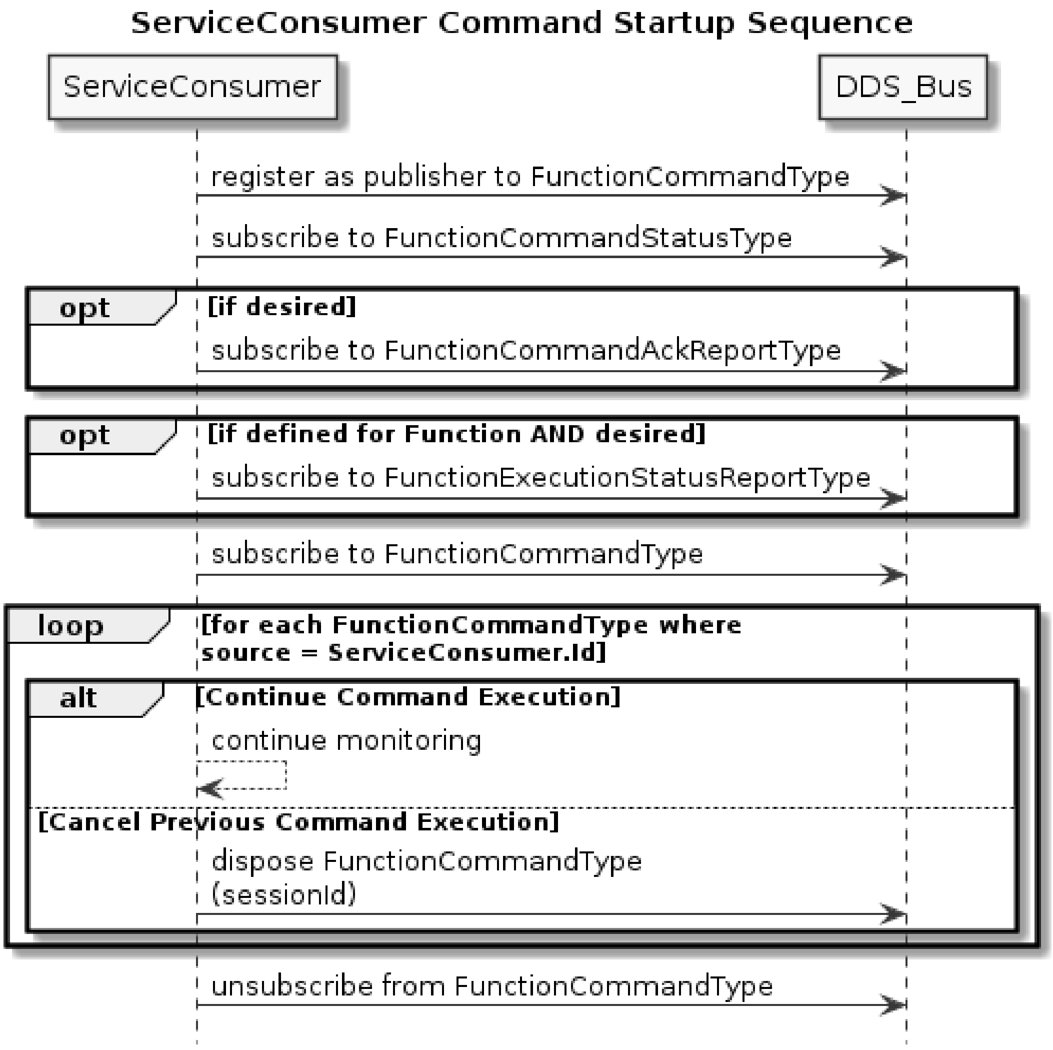

ServiceConsumer Command Startup Sequence
ServiceConsumer
DDS_Bus
register as publisher to FunctionCommandType
subscribe to FunctionCommandStatusType
opt
[if desired]
subscribe to FunctionCommandAckReportType
opt
[if defined for Function AND desired]
subscribe to FunctionExecutionStatusReportType
subscribe to FunctionCommandType
loop
[for each FunctionCommandType where source = ServiceConsumer.Id]
alt
[Continue Command Execution]
continue monitoring
[Cancel Previous Command Execution]
dispose FunctionCommandType (sessionId)
unsubscribe from FunctionCommandType

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

5.1.3 Command Execution Sequences

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

5.1.4 Command Start Sequence

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

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

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

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

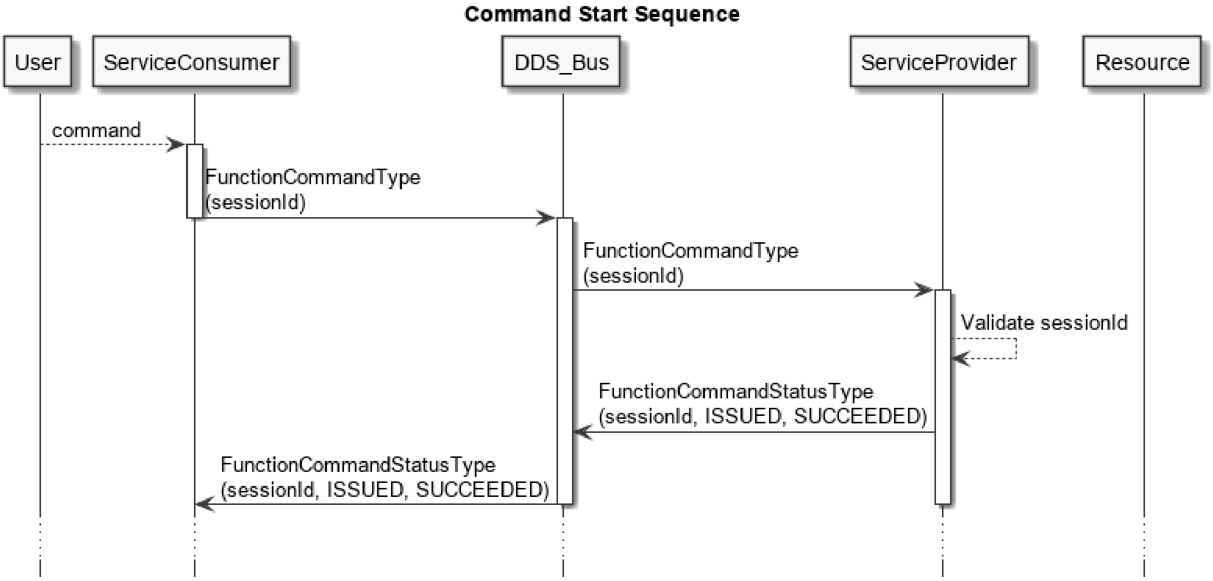

Command Start Sequence
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
command
FunctionCommandType (sessionId)
FunctionCommandType (sessionId)
Validate sessionId
FunctionCommandStatusType (sessionId, ISSUED, SUCCEEDED)
FunctionCommandStatusType (sessionId, ISSUED, SUCCEEDED)

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

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

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

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

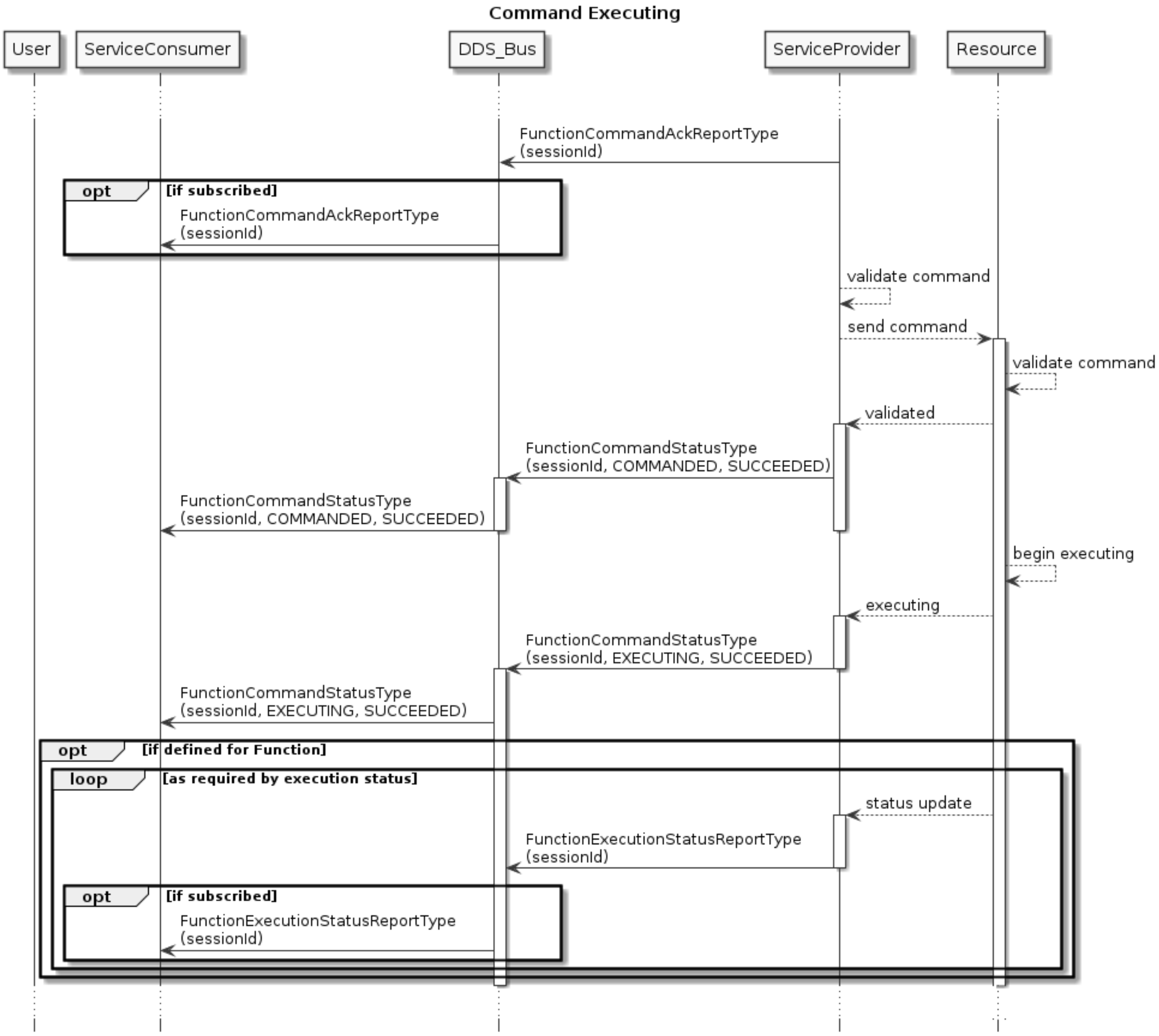

Command Executing
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
FunctionCommandAckReportType (sessionId)
opt [if subscribed]
FunctionCommandAckReportType (sessionId)
validate command
send command
validate command
validated
FunctionCommandStatusType (sessionId, COMMANDED, SUCCEEDED)
FunctionCommandStatusType (sessionId, COMMANDED, SUCCEEDED)
begin executing
executing
FunctionCommandStatusType (sessionId, EXECUTING, SUCCEEDED)
FunctionCommandStatusType (sessionId, EXECUTING, SUCCEEDED)
opt [if defined for Function]
loop [as required by execution status]
status update
FunctionExecutionStatusReportType (sessionId)
opt [if subscribed]
FunctionExecutionStatusReportType (sessionId)

Figure 19: Beginning Sequence Diagram for a Command Execution.

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

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

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

The flow control for command update is detailed below:

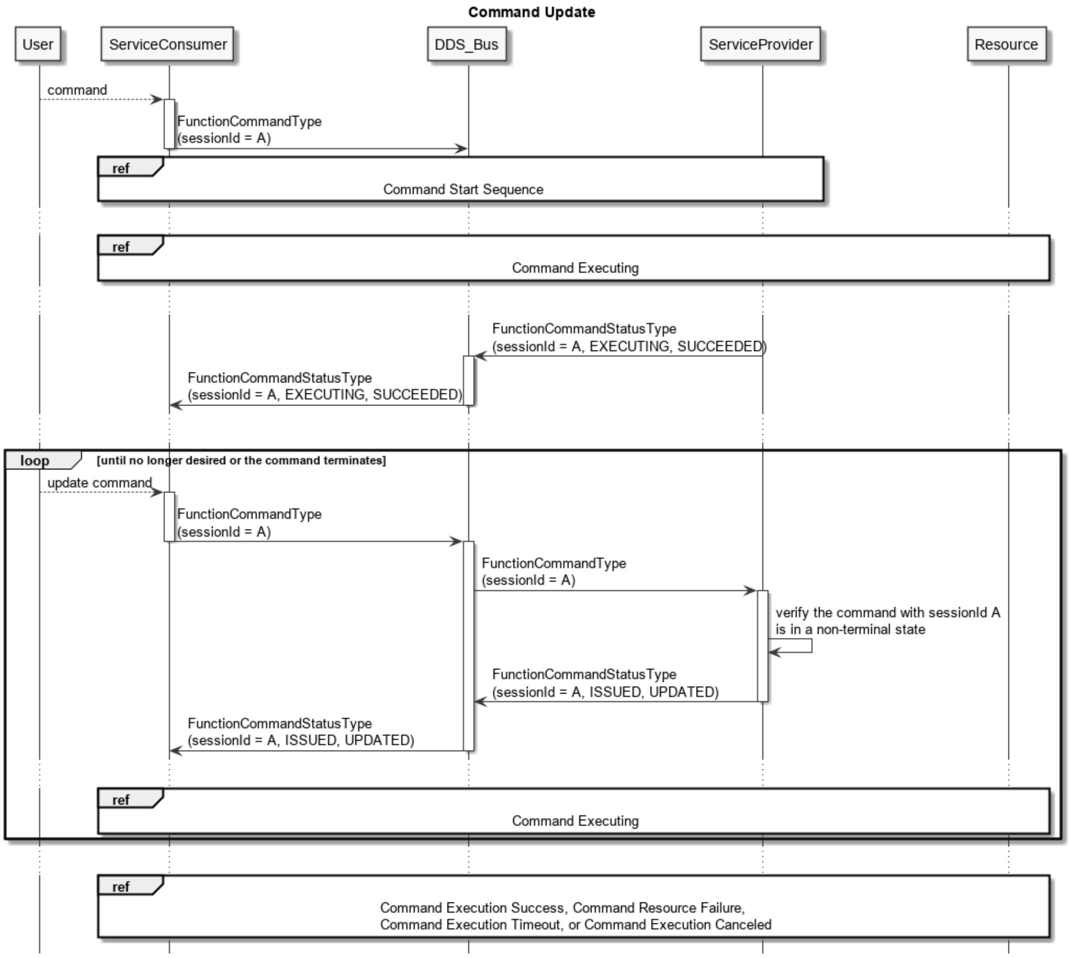

Command Update
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
command
FunctionCommandType (sessionId = A)
ref
Command Start Sequence
ref
Command Executing
FunctionCommandStatusType (sessionId = A, EXECUTING, SUCCEEDED)
FunctionCommandStatusType (sessionId = A, EXECUTING, SUCCEEDED)
loop
[until no longer desired or the command terminates]
update command
FunctionCommandType (sessionId = A)
FunctionCommandType (sessionId = A)
verify the command with sessionId A is in a non-terminal state
FunctionCommandStatusType (sessionId = A, ISSUED, UPDATED)
FunctionCommandStatusType (sessionId = A, ISSUED, UPDATED)
ref
Command Executing
ref
Command Execution Success, Command Resource Failure, Command Execution Timeout, or Command Execution Canceled

Figure 20: Sequence Diagram for Command Update.

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

The Command Execution Success sequence is shown in Figure 21.

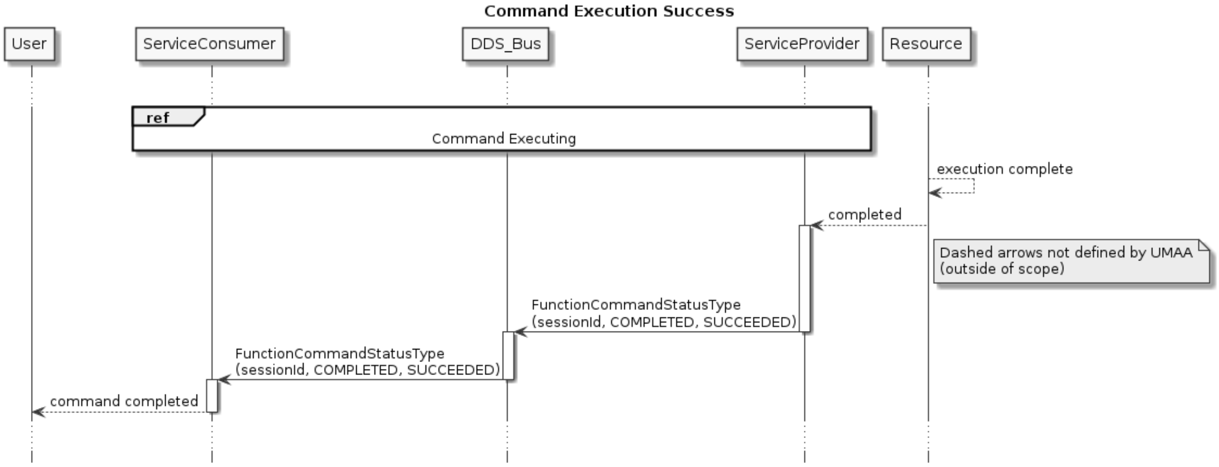

Command Execution Success
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
ref
Command Executing
execution complete
completed
Dashed arrows not defined by UMAA (outside of scope)
FunctionCommandStatusType (sessionId, COMPLETED, SUCCEEDED)
FunctionCommandStatusType (sessionId, COMPLETED, SUCCEEDED)
command completed

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

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

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

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

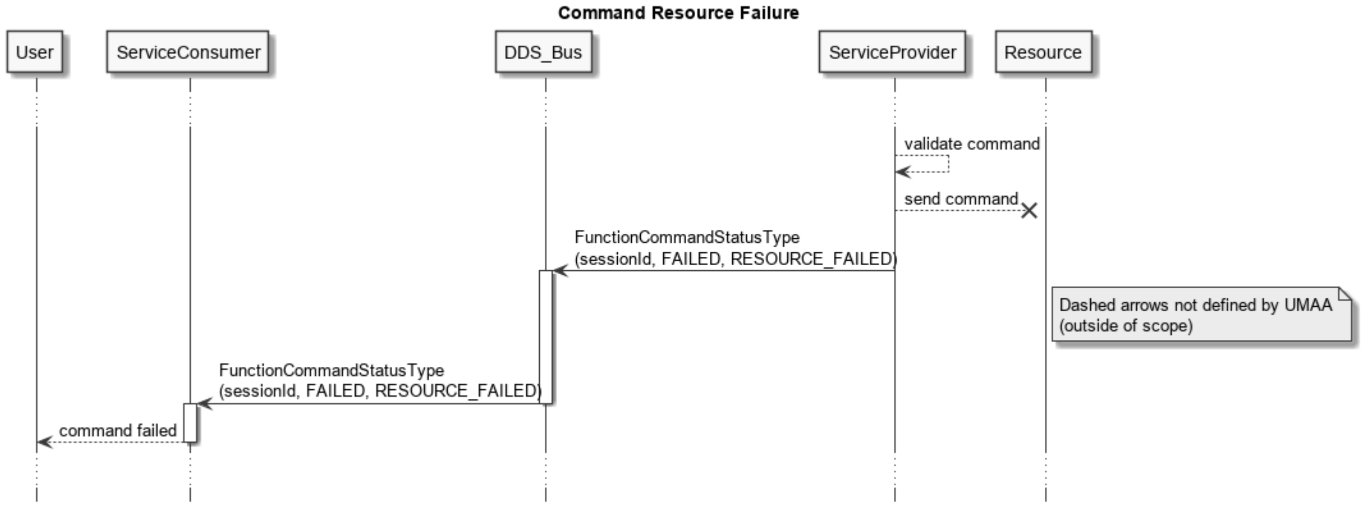

Command Resource Failure
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
validate command
send command
FunctionCommandStatusType
(sessionId, FAILED, RESOURCE_FAILED)
Dashed arrows not defined by UMAA
(outside of scope)
FunctionCommandStatusType
(sessionId, FAILED, RESOURCE_FAILED)
command failed

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

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

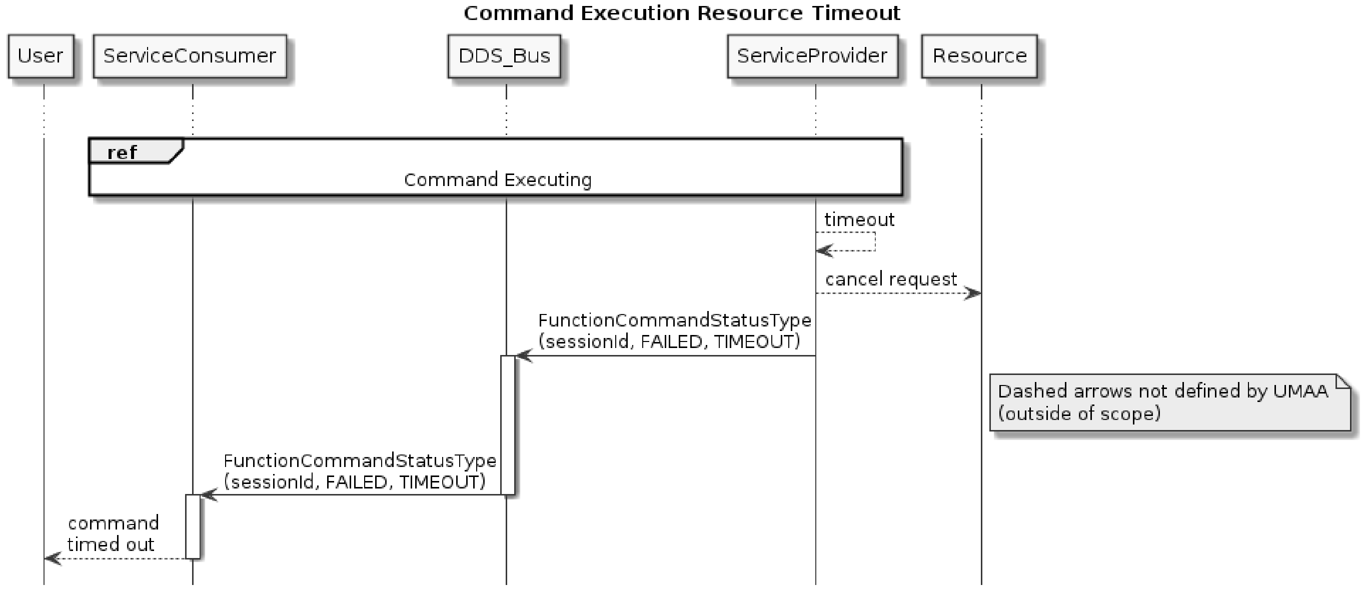

Command Execution Resource Timeout
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
ref
Command Executing
timeout
cancel request
FunctionCommandStatusType
(sessionId, FAILED, TIMEOUT)
Dashed arrows not defined by UMAA
(outside of scope)
FunctionCommandStatusType
(sessionId, FAILED, TIMEOUT)
command
timed out

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

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

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

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

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

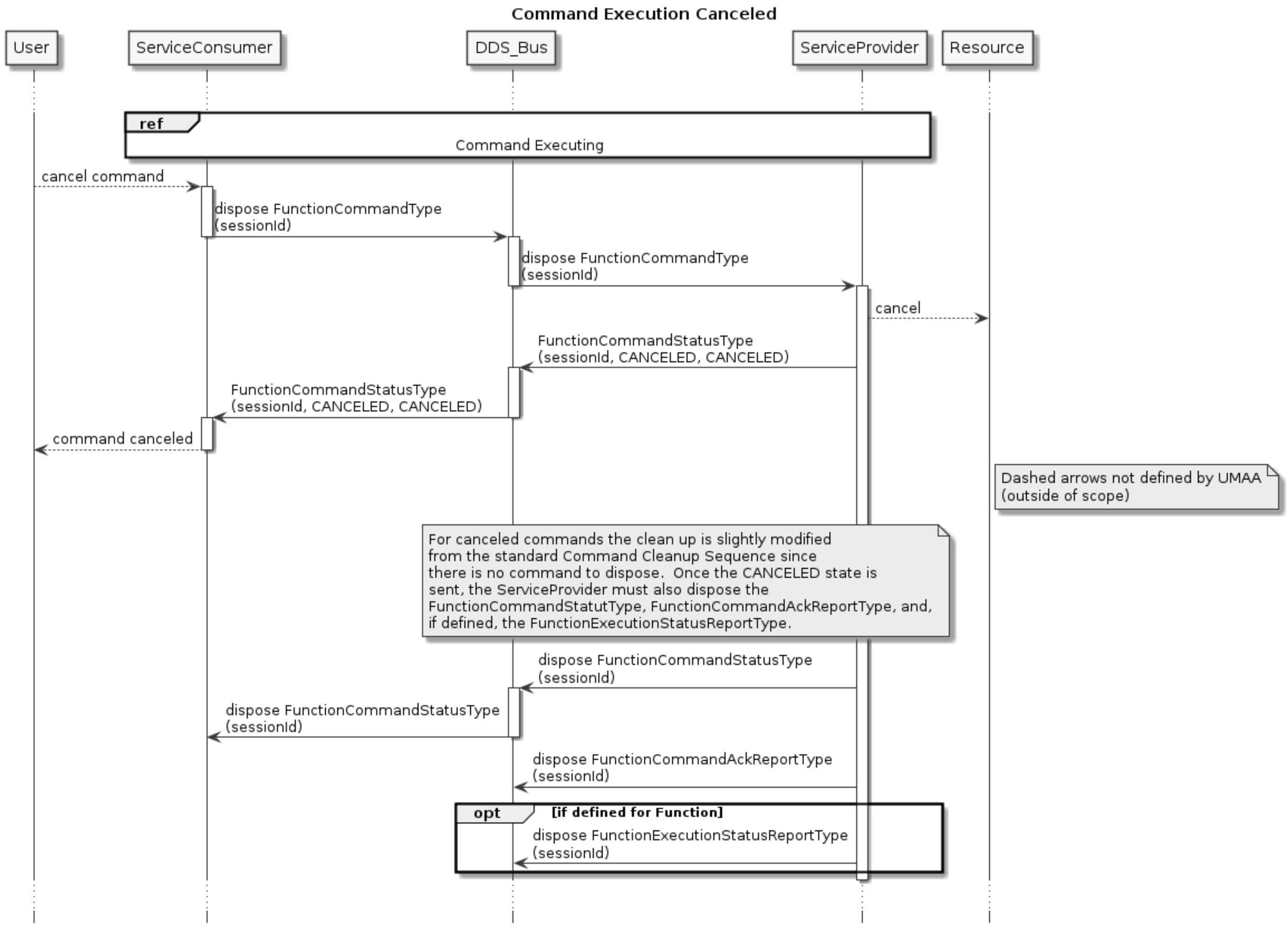

Command Execution Canceled
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
ref
Command Executing
cancel command
dispose FunctionCommandType (sessionId)
dispose FunctionCommandType (sessionId)
cancel
FunctionCommandStatusType (sessionId, CANCELED, CANCELED)
FunctionCommandStatusType (sessionId, CANCELED, CANCELED)
command canceled
Dashed arrows not defined by UMAA (outside of scope)
For canceled commands the clean up is slightly modified from the standard Command Cleanup Sequence since there is no command to dispose. Once the CANCELED state is sent, the ServiceProvider must also dispose the FunctionCommandStatutType, FunctionCommandAckReportType, and, if defined, the FunctionExecutionStatusReportType.
dispose FunctionCommandStatusType (sessionId)
dispose FunctionCommandStatusType (sessionId)
dispose FunctionCommandAckReportType (sessionId)
opt
[if defined for Function]
dispose FunctionExecutionStatusReportType (sessionId)

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

5.1.5 Command Cleanup

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

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

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

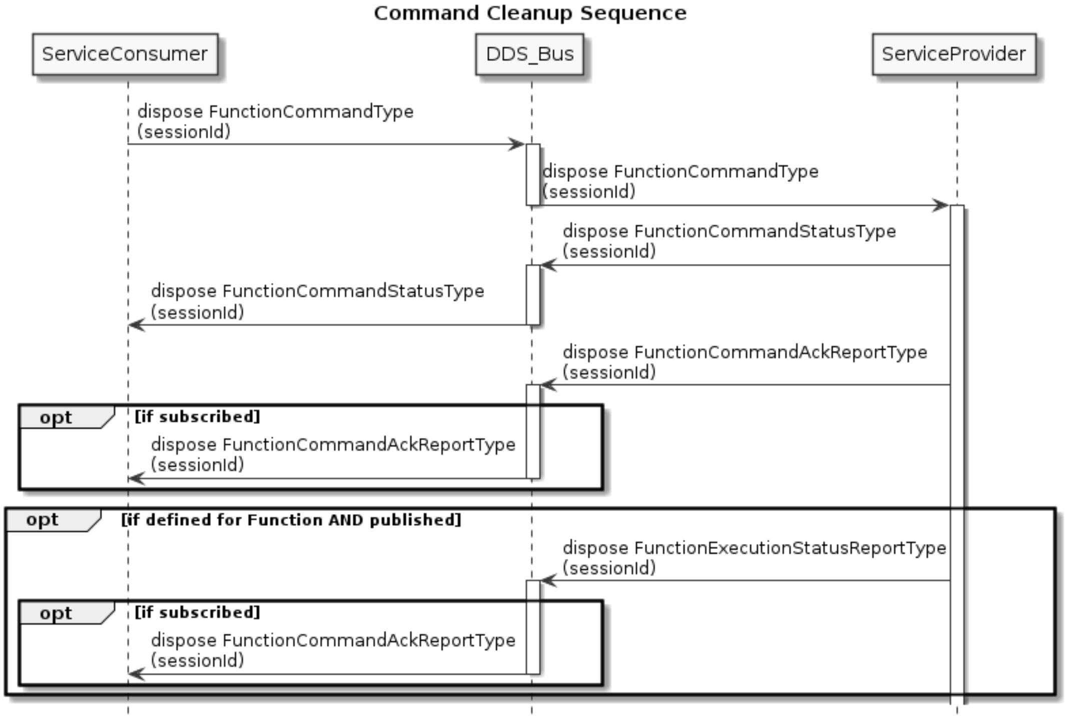

Command Cleanup Sequence
ServiceConsumer
DDS_Bus
ServiceProvider
dispose FunctionCommandType (sessionId)
dispose FunctionCommandType (sessionId)
dispose FunctionCommandStatusType (sessionId)
dispose FunctionCommandStatusType (sessionId)
dispose FunctionCommandAckReportType (sessionId)
opt [if subscribed]
dispose FunctionCommandAckReportType (sessionId)
opt [if defined for Function AND published]
dispose FunctionExecutionStatusReportType (sessionId)
opt [if subscribed]
dispose FunctionCommandAckReportType (sessionId)

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

5.1.6 Command Shutdown Sequence

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

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

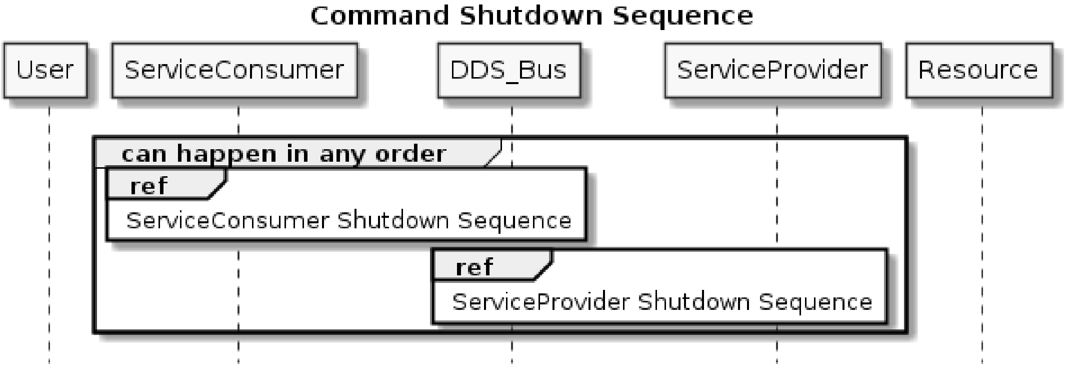

Command Shutdown Sequence
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
can happen in any order
ref
ServiceConsumer Shutdown Sequence
ref
ServiceProvider Shutdown Sequence

Figure 26: Sequence Diagram for Command Shutdown.

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

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

The Service Provider Shutdown sequence is shown in Figure 27.

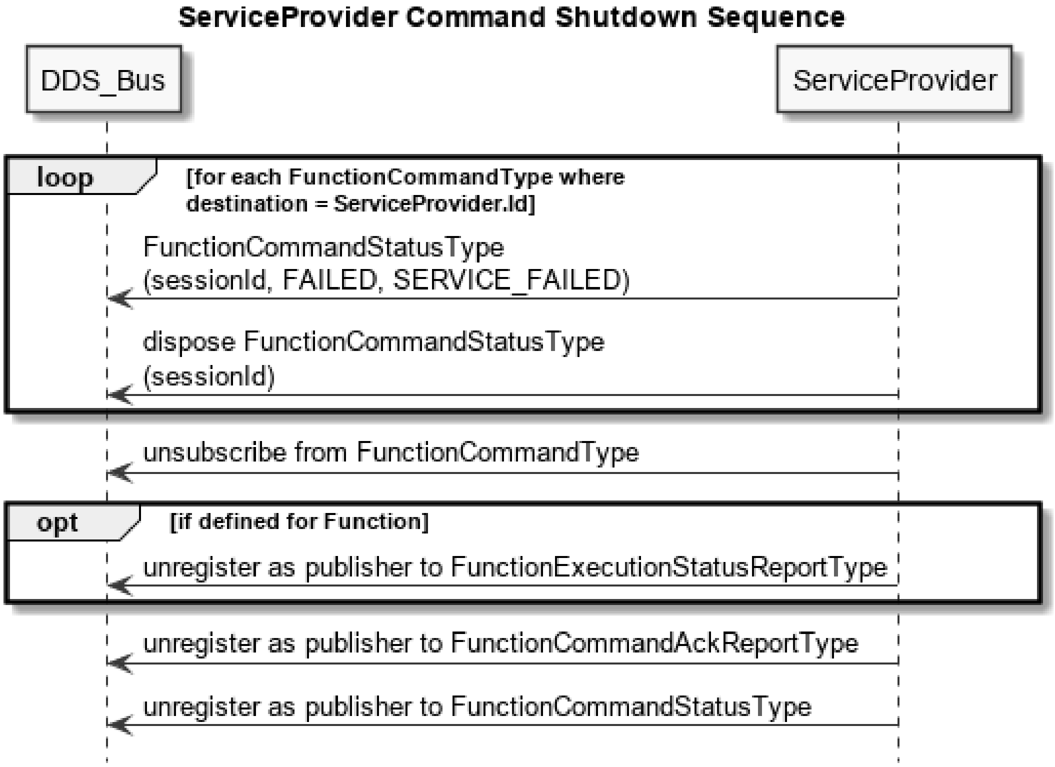

ServiceProvider Command Shutdown Sequence
DDS_Bus
ServiceProvider
loop
[for each FunctionCommandType where destination = ServiceProvider.Id]
FunctionCommandStatusType (sessionId, FAILED, SERVICE_FAILED)
dispose FunctionCommandStatusType (sessionId)
unsubscribe from FunctionCommandType
opt
[if defined for Function]
unregister as publisher to FunctionExecutionStatusReportType
unregister as publisher to FunctionCommandAckReportType
unregister as publisher to FunctionCommandStatusType

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

5.1.6.2 Service Consumer Shutdown Sequence

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

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

The Service Consumer Shutdown sequence is shown in Figure 28.

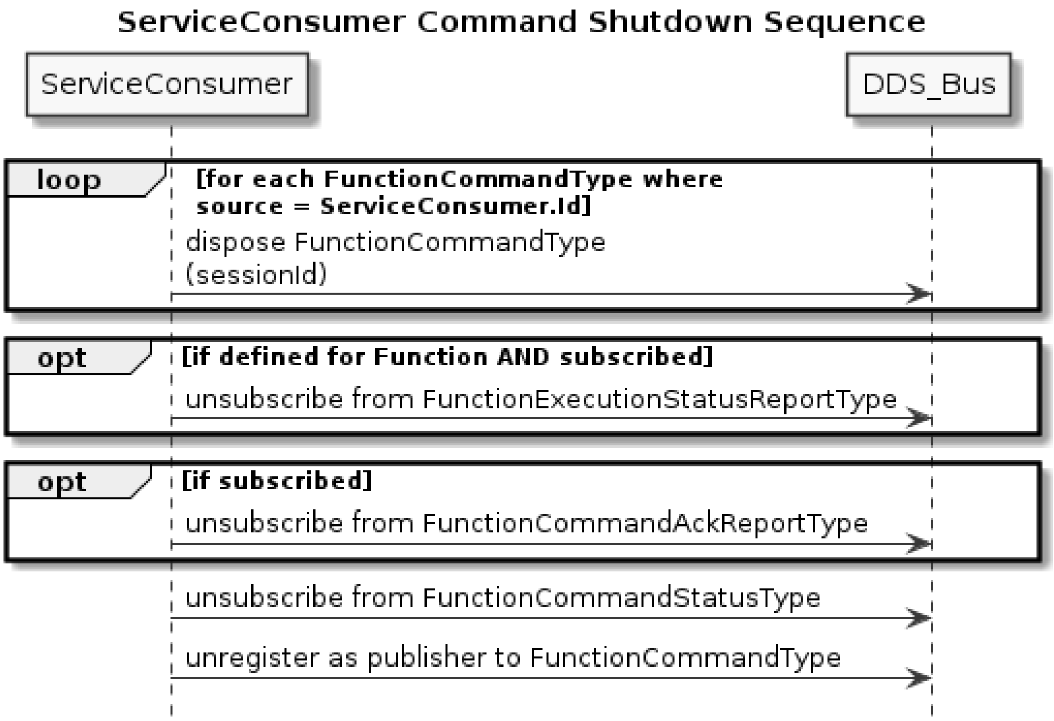

ServiceConsumer Command Shutdown Sequence
ServiceConsumer
DDS_Bus
loop
[for each FunctionCommandType where source = ServiceConsumer.Id]
dispose FunctionCommandType (sessionId)
opt
[if defined for Function AND subscribed]
unsubscribe from FunctionExecutionStatusReportType
opt
[if subscribed]
unsubscribe from FunctionCommandAckReportType
unsubscribe from FunctionCommandStatusType
unregister as publisher to FunctionCommandType

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

5.2 Request / Reply

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

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

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

5.2.1 Request/Reply without Query Data

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

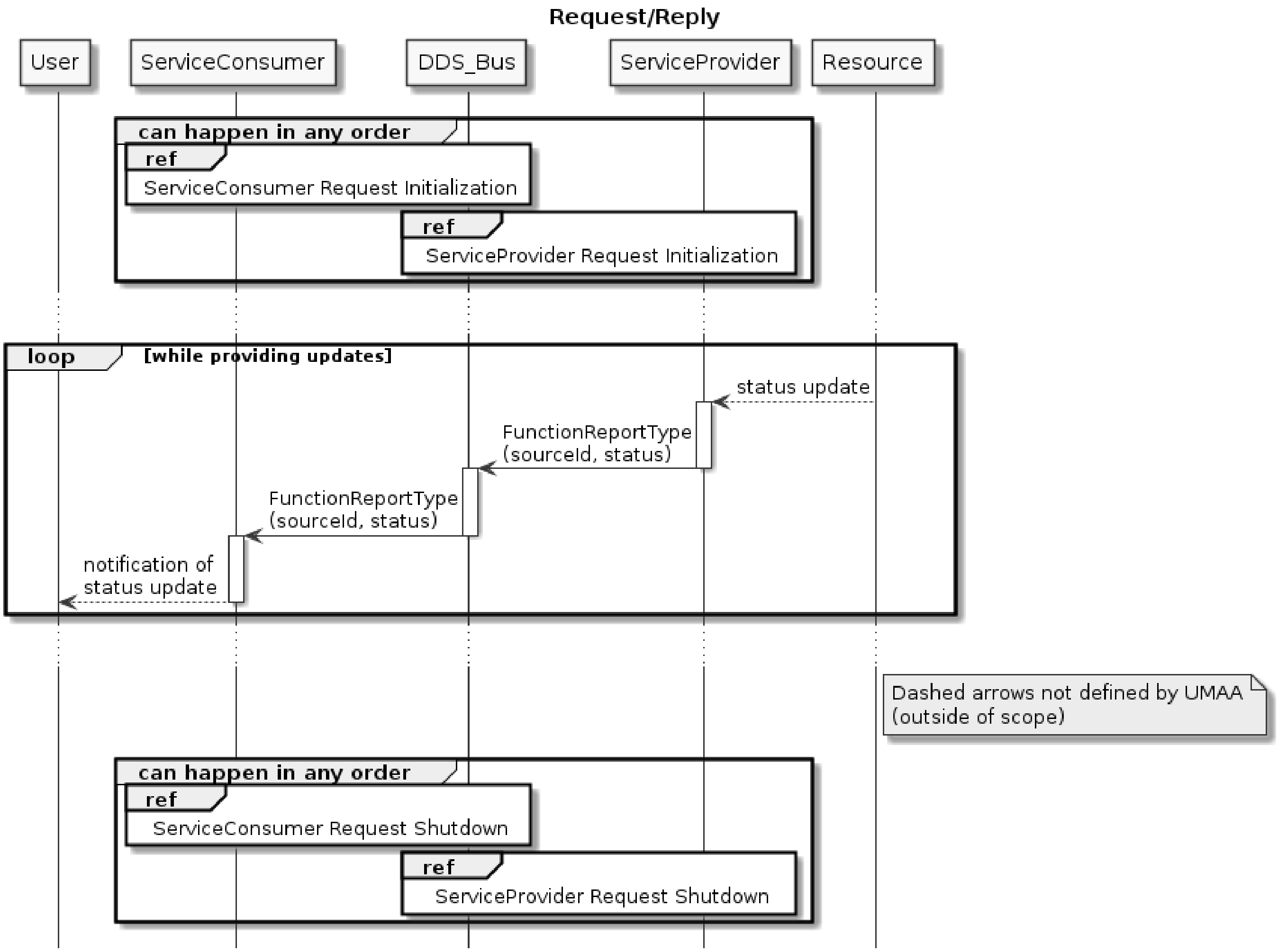

Request/Reply
User
ServiceConsumer
DDS_Bus
ServiceProvider
Resource
can happen in any order
ref
ServiceConsumer Request Initialization
ref
ServiceProvider Request Initialization
loop [while providing updates]
status update
FunctionReportType (sourceId, status)
FunctionReportType (sourceId, status)
notification of status update
Dashed arrows not defined by UMAA (outside of scope)
can happen in any order
ref
ServiceConsumer Request Shutdown
ref
ServiceProvider Request Shutdown

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

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

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

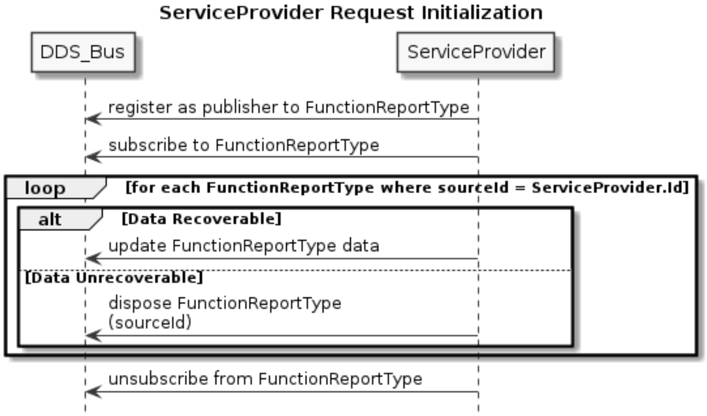

ServiceProvider Request Initialization
DDS_Bus
ServiceProvider
register as publisher to FunctionReportType
subscribe to FunctionReportType
loop
[for each FunctionReportType where sourceId = ServiceProvider.Id]
alt
[Data Recoverable]
update FunctionReportType data
[Data Unrecoverable]
dispose FunctionReportType
(sourceId)
unsubscribe from FunctionReportType

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

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

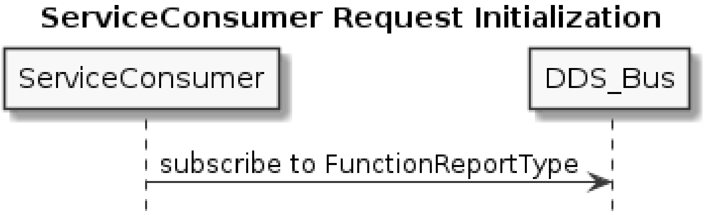

ServiceConsumer Request Initialization
ServiceConsumer
DDS_Bus
subscribe to FunctionReportType

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

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

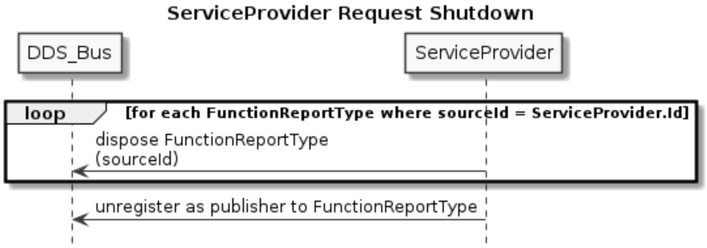

ServiceProvider Request Shutdown
DDS_Bus
ServiceProvider
loop
[for each FunctionReportType where sourceId = ServiceProvider.Id]
dispose FunctionReportType
(sourceId)
unregister as publisher to FunctionReportType

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

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

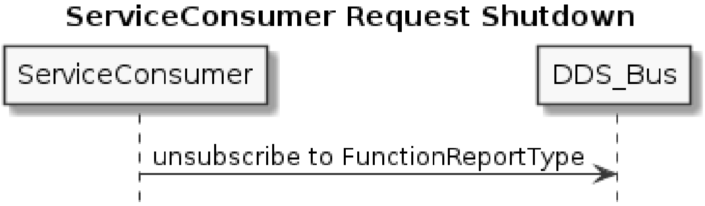

ServiceConsumer Request Shutdown
ServiceConsumer
DDS_Bus
unsubscribe to FunctionReportType

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

5.2.2 Request/Reply with Query Data

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

6 Mission Management - Experimental (MM-EXP) Services and Interfaces

Mission Management, as referenced in the context of UMAA, defines a collection of services and interfaces pertinent to the management of UMV mission operations. These services support fundamental operations such as ingesting a mission plan, decomposing the mission into more granular actions via objectives, assigning resources to satisfy objectives, and other related actions. The establishment of a multi-level mission plan structure is fundamental to the definition and implementation of the requisite services and interfaces. Equally important is the identification and definition of related terms. This section defines the mission plan structure and associated key term definitions useful in understanding and implementing the prescribed services and interfaces.

Key Terms

The terms listed in Table 8 provide context to the description of the mission plan structure, service specifications, and interface designs.

Table 8: Mission Management Key Terms and Definitions

Term	Definition
Mission Plan[3]	A plan for a series of related tasks aimed at achieving strategic or operational objectives within a given time and space. The **Mission Plan** is modeled as one or more related **Task Plans**, allowing for multiple planned tasks for either a single UMV or multiple UMVs.
Task Plan[3]	A complete and detailed plan containing a full description of the objectives, all execution constraints, and execution criteria. The **Task Plan** is modeled as one or more related **Objectives**, typically involving a single UMV.
Objective[3]	The clearly defined, decisive, and attainable goal toward which a **Task Plan** is directed for a **Resource** to achieve. An **Objective** provides a measurable outcome that can be assessed to ensure completion. The **Objective** is modeled as a generalization, containing attributes common to all **Objectives**, and a set of **Objective** specializations that define attributes specific to the particular goal of the **Objective**.
Resource[3]	The assets or capabilities apportioned or allocated to a **Mission Plan**, **Task Plan**, or **Objective**. A **Resource** can be decomposed into lower-level **Resources** and therefore, can be a component within a system, the system itself, or even a system of systems.
Constraint Plan[3]	In the context of planning, a requirement placed on the execution of a **Mission Plan**, **Task Plan**, or **Objective**, that restricts freedom of action.
Trigger	A mechanism that initiates executing a **Mission Plan**, **Task Plan**, or **Objective** and/or enabling a **Constraint Plan**, when its defined conditional expression is determined to be logically true.

Mission Definition

Missions are defined by the objectives that must be achieved along with any constraints that must be considered while attempting to achieve those objectives. Additionally, triggers are used to control when an objective is executed and/or when a constraint is enabled. The services used to provide the current mission's objectives, constraints, and triggers are `MissionPlanReport`, `ConstraintPlanReport`, and `TriggerReport`, respectively, and additional corresponding services support adding/deleting them to/from the mission definition. A provision for supporting the compartmentalization of objectives into operator-defined objective collections is afforded by the task plan. Figure 34 provides a UML diagram showing the relationship of the data structures used to model the mission definition (i.e., the mission's objectives, constraints, and triggers), as well as their associated services. The sections that follow provide additional details on these data structures.

[3] Adapted from the DOD Dictionary of Military and Associated Terms, August 2018.

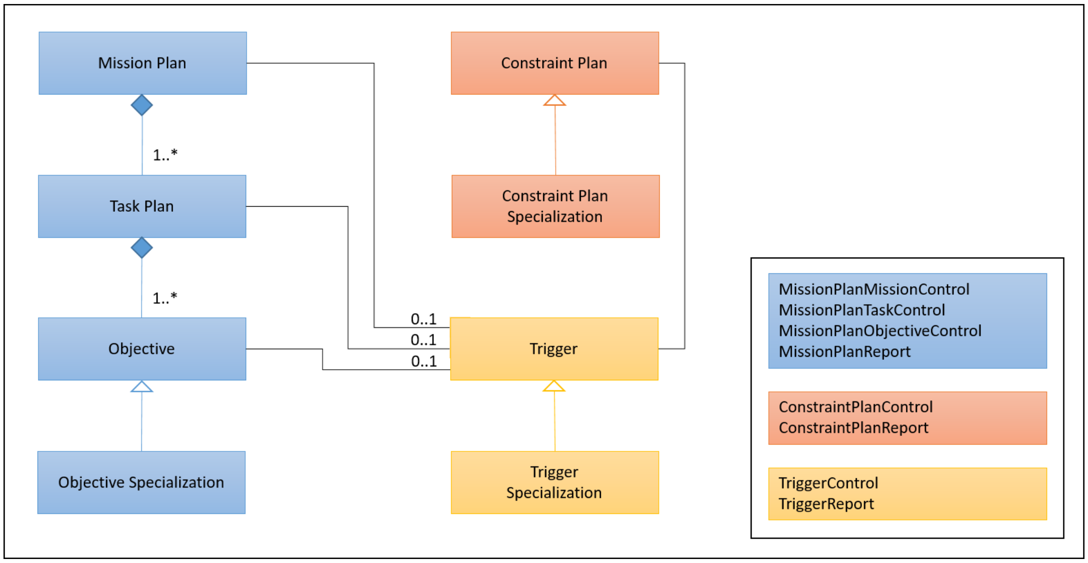


Figure 34: Mission definition UML diagram.

Mission Plan Structure

The mission plan structure, established to support MM service and interface definitions, represents a balance between required administrative, atomic deconstruction of higher level goals into manageable discrete actions and the increasing complexity imbued as additional hierarchical layers are introduced. The three level structure employed for a mission plan provides two layers of administrative segregation (mission plan and task plan) and, at a minimum, a single actionable layer addressing mission execution (objective). Two administrative layers ease the administrative planning burden anticipated with multiple platform and / or long duration, discrete operation missions. The most granular, required layer—Layer 3—is optionally decomposable into further nested layers at the discretion of the mission management implementation architect via child objectives. A diagrammatical representation of the mission plan structure is provided in Figure 35.

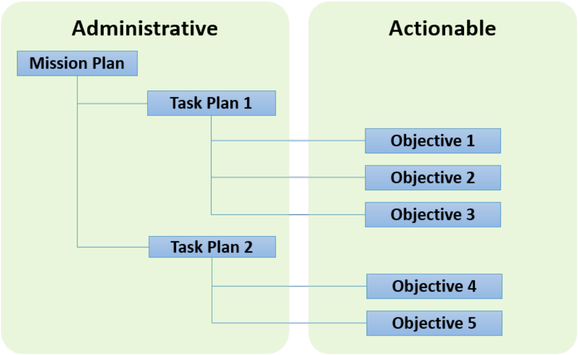


Figure 35: Mission plan structure depicting three-layer implementation.

Two example missions demonstrating the utility of the three-layer mission plan structure are provided in Figure 36. These examples illustrate the flexibility benefits of the two-tier administrative plan structure by accommodating both multi-vehicle, single-task and multi-task, single vehicle missions.

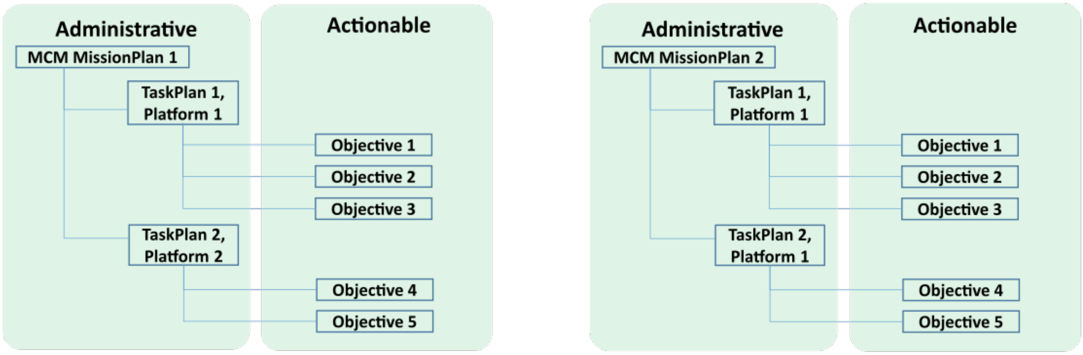
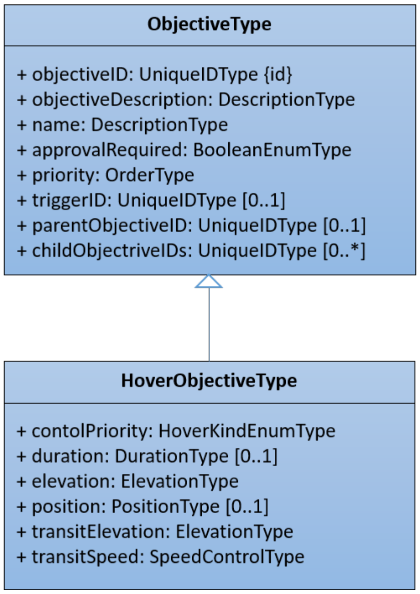


Figure 36: Depiction of flexible two-layer administrative mission plan structure.

The Objectives layer of the mission plan structure employs an inheritance model wherein attributes pertinent to all objectives are associated with a generalization, and attributes unique to specific objectives are captured within individual Objective specializations. As an example, the data model for a Hover Objective is provided in Figure 37.


ObjectiveType
+ objectiveID: UniqueIDType {id}
+ objectiveDescription: DescriptionType
+ name: DescriptionType
+ approvalRequired: BooleanEnumType
+ priority: OrderType
+ triggerID: UniqueIDType [0..1]
+ parentObjectiveID: UniqueIDType [0..1]
+ childObjectriveIDs: UniqueIDType [0..*]
HoverObjectiveType
+ contolPriority: HoverKindEnumType
+ duration: DurationType [0..1]
+ elevation: ElevationType
+ position: PositionType [0..1]
+ transitElevation: ElevationType
+ transitSpeed: SpeedControlType

Defines attributes general to all Objectives (i.e., meta-data)

Defines attributes specific to the Hover Objective

Figure 37: Objective and Objective Specialization data and message model representations.

Constraint Plan Data Model

The constraint plan structure is used to define restrictions on how a mission is executed. Its data structure is also modeled using a generalization/specialization relationship similar to objectives. An example of a constraint is to limit how a vehicle maneuvers while executing a route objective such as its maximum depth, maximum turn rate, and maximum speed. The data model for a maximum speed constraint is shown in Figure 38. Note that a constraint must define an associated trigger, which indicates when the constraint is enabled.

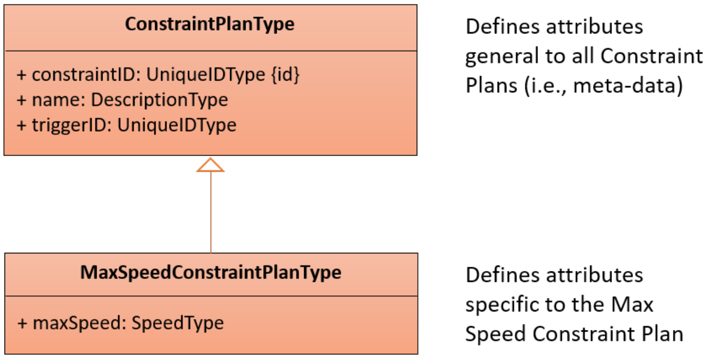

ConstraintPlanType
+ constraintID: UniqueIDType {id}
+ name: DescriptionType
+ triggerID: UniqueIDType
Defines attributes general to all Constraint Plans (i.e., meta-data)
MaxSpeedConstraintPlanType
+ maxSpeed: SpeedType
Defines attributes specific to the Max Speed Constraint Plan

Figure 38: Constraint Plan and Constraint Plan Specialization data and message model representations.

Trigger Data Model

The trigger data structure is used to define logical conditional expressions that control the execution of the mission and/or enable one or more constraints. Again, its data structure is modeled using a generalization/specialization relationship similar to objectives and constraints. Figure 39 shows an example of a trigger that depends on the current execution state of an objective.

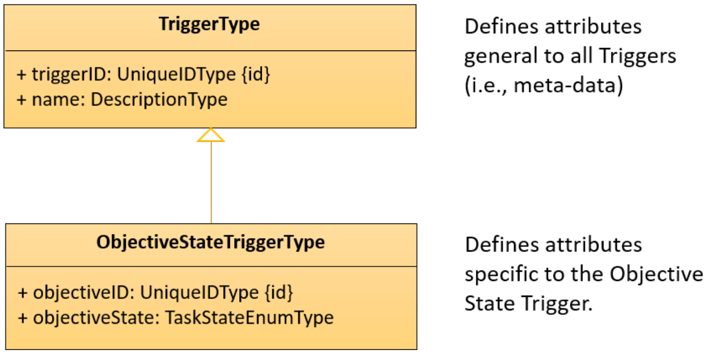

TriggerType
+ triggerID: UniqueIDType {id}
+ name: DescriptionType
Defines attributes general to all Triggers (i.e., meta-data)
ObjectiveStateTriggerType
+ objectiveID: UniqueIDType {id}
+ objectiveState: TaskStateEnumType
Defines attributes specific to the Objective State Trigger.

Figure 39: Trigger and Trigger Specialization data and message model representations.

Triggers can be combined through logical operation triggers. Triggers for the logical operations `AND`, `OR`, and `NOT` are currently defined, where Figure 40 shows the trigger for a logical `AND` operation. As this allows for a trigger to be defined by multiple layers of logical operations, in order to evaluate a logical trigger, all of its dependent triggers must be resolved first. As an example, consider the logical expressions `(A OR (B AND C))`, and `((A OR B) AND C)`. Figure 41 graphically shows these two cases in a tree structure, where for the `(A OR (B AND C))` case, the `AND` trigger must be resolved before evaluating the `OR` trigger. And similarly, for the `((A OR B) AND C)` case, the `OR` trigger must be resolved before evaluating the `AND` trigger. As this requirement on processing the triggers defines the order of evaluation, care should be taken when building the tree to ensure the structure defines the desired logic.

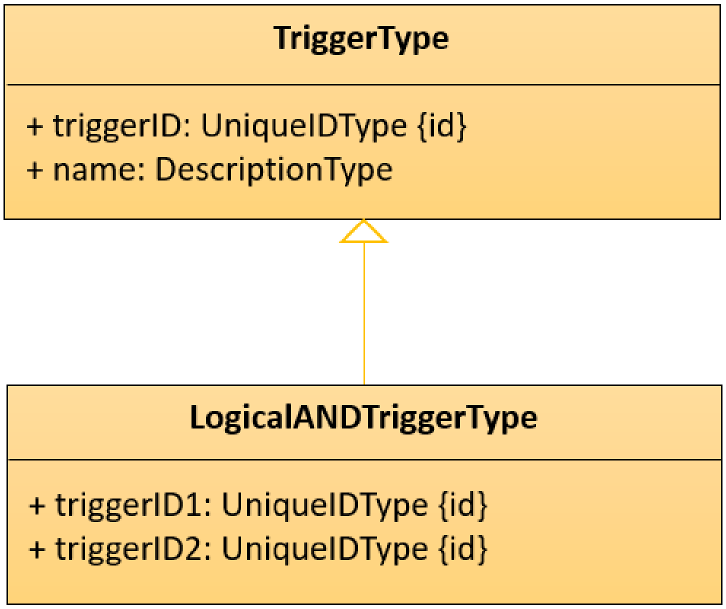

TriggerType
+ triggerID: UniqueIDType {id}
+ name: DescriptionType
LogicalANDTriggerType
+ triggerID1: UniqueIDType {id}
+ triggerID2: UniqueIDType {id}

Figure 40: Logical AND trigger data and message model representations.

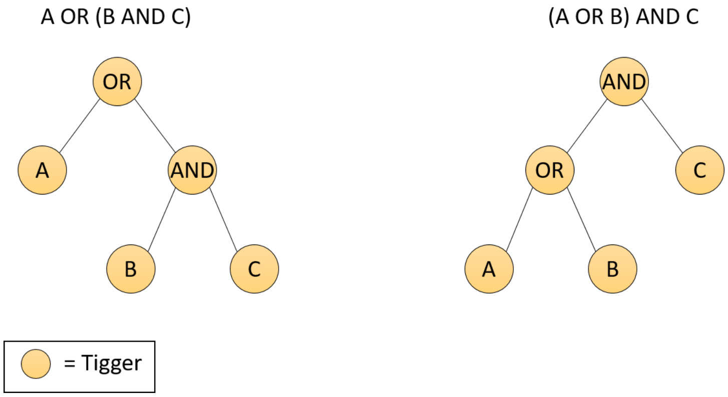


Figure 41: Logical operator trigger dependency tree.

6.1 Services and Interfaces

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

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

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

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

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

Operations without DDS Topics

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

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

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

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

6.1.1 ConstraintPlanControl

The purpose of this service is to manage planned constraints that must be handled during mission plan execution.

Table 9: ConstraintPlanControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setConstraintPlanAdd	reportConstraintPlanAddCommandStatus
queryConstraintPlanAddCommandAck⊕	reportConstraintPlanAddCommandAck
cancelConstraintPlanAddCommand⊕	reportConstraintPlanAddCancelCommandStatus⊕
setConstraintPlanDelete	reportConstraintPlanDeleteCommandStatus
queryConstraintPlanDeleteCommandAck⊕	reportConstraintPlanDeleteCommandAck
cancelConstraintPlanDeleteCommand⊕	reportConstraintPlanDeleteCancelCommandStatus⊕

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

6.1.1.1 reportConstraintPlanAddCommandAck

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

Namespace: UMAA::MM::ConstraintPlanControl

Topic: ConstraintPlanAddCommandAckReport

Data Type: ConstraintPlanAddCommandAckReportType

Table 10: ConstraintPlanAddCommandAckReportType Message Definition

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

6.1.1.2 reportConstraintPlanAddCommandStatus

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

Namespace: UMAA::MM::ConstraintPlanControl

Topic: ConstraintPlanAddCommandStatus

Data Type: ConstraintPlanAddCommandStatusType

Table 11: ConstraintPlanAddCommandStatusType Message Definition

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

6.1.1.3 reportConstraintPlanDeleteCommandAck

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

Namespace: UMAA::MM::ConstraintPlanControl

Topic: ConstraintPlanDeleteCommandAckReport

Data Type: ConstraintPlanDeleteCommandAckReportType

Table 12: ConstraintPlanDeleteCommandAckReportType Message Definition

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

6.1.1.4 reportConstraintPlanDeleteCommandStatus

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

Namespace: UMAA::MM::ConstraintPlanControl

Topic: ConstraintPlanDeleteCommandStatus

Data Type: ConstraintPlanDeleteCommandStatusType

Table 13: ConstraintPlanDeleteCommandStatusType Message Definition

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

6.1.1.5 setConstraintPlanAdd

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

Namespace: UMAA::MM::ConstraintPlanControl

Topic: ConstraintPlanAddCommand

Data Type: ConstraintPlanAddCommandType

Table 14: ConstraintPlanAddCommandType Message Definition

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

6.1.1.6 setConstraintPlanDelete

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

Namespace: UMAA::MM::ConstraintPlanControl

Topic: ConstraintPlanDeleteCommand

Data Type: ConstraintPlanDeleteCommandType

Table 15: ConstraintPlanDeleteCommandType 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.

6.1.2 ConstraintPlanReport

The purpose of this service is to report the set of planned constraints that have successfully been added for use during mission plan execution.

Table 16: ConstraintPlanReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryConstraintPlan⊕	reportConstraintPlan

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

6.1.2.1 reportConstraintPlan

Description: This operation provides the current set of constraints for mission plan execution.

Namespace: UMAA::MM::ConstraintPlanReport

Topic: ConstraintPlanReport

Data Type: ConstraintPlanReportType

Table 17: ConstraintPlanReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
constraints→setID	LargeSet<ConstraintPlanType>	Defines current set of constraints 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::ConstraintPlanReport::ConstraintPlanReportConstraintsSetElement.

6.1.3 MissionPlanAssignmentReport

The purpose of this service is to assign a mission plan to a particular resource or set of resources required to accomplish the mission plan.

Table 18: MissionPlanAssignmentReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMissionPlanAssignment⊕	reportMissionPlanAssignment

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

6.1.3.1 reportMissionPlanAssignment

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

Namespace: UMAA::MM::MissionPlanAssignmentReport

Topic: MissionPlanAssignmentReport

Data Type: MissionPlanAssignmentReportType

Table 19: MissionPlanAssignmentReportType Message Definition

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

Attribute Name	Attribute Type	Attribute Description
resourceIDs	sequence<NumericGUID> max size = 256	Identifies the resources that are assigned to accomplish the mission plan.
missionID*	NumericGUID	The identifier of the mission plan.

6.1.4 MissionPlanExecutionControl

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

Table 20: 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 ⊕.

6.1.4.1 reportMissionPlanExecutionCommandAck

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

Namespace: UMAA::MM::MissionPlanExecutionControl

Topic: MissionPlanExecutionCommandAckReport

Data Type: MissionPlanExecutionCommandAckReportType

Table 21: MissionPlanExecutionCommandAckReportType Message Definition

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

6.1.4.2 reportMissionPlanExecutionCommandStatus

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

Namespace: UMAA::MM::MissionPlanExecutionControl

Topic: MissionPlanExecutionCommandStatus

Data Type: MissionPlanExecutionCommandStatusType

Table 22: MissionPlanExecutionCommandStatusType Message Definition

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

6.1.4.3 setMissionPlanExecution

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

Namespace: UMAA::MM::MissionPlanExecutionControl

Topic: MissionPlanExecutionCommand

Data Type: MissionPlanExecutionCommandType

Table 23: MissionPlanExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
state	TaskControlEnumType	The desired state of the current mission plan specified by missionID.
missionID*	NumericGUID	The identifier of the mission plan.

6.1.5 MissionPlanExecutionStatus

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

Table 24: 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 ⊕.

6.1.5.1 reportMissionPlanExecution

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

Namespace: UMAA::MM::MissionPlanExecutionStatus

Topic: MissionPlanExecutionReport

Data Type: MissionPlanExecutionReportType

Table 25: MissionPlanExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
missionPlanStatuses	sequence<MissionPlanStatusType>	Provides the status of the mission plan(s).

6.1.6 MissionPlanMissionControl

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

Table 26: MissionPlanMissionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanMissionAdd	reportMissionPlanMissionAddCommandStatus
queryMissionPlanMissionAddCommandAck⊕	reportMissionPlanMissionAddCommandAck
cancelMissionPlanMissionAddCommand⊕	reportMissionPlanMissionAddCancelCommandStatus⊕
setMissionPlanMissionDelete	reportMissionPlanMissionDeleteCommandStatus
queryMissionPlanMissionDeleteCommandAck⊕	reportMissionPlanMissionDeleteCommandAck
cancelMissionPlanMissionDeleteCommand⊕	reportMissionPlanMissionDeleteCancelCommandStatus⊕

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

6.1.6.1 reportMissionPlanMissionAddCommandAck

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionAddCommandAckReport

Data Type: MissionPlanMissionAddCommandAckReportType

Table 27: MissionPlanMissionAddCommandAckReportType Message Definition

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

6.1.6.2 reportMissionPlanMissionAddCommandStatus

Description: This operation provides the status of the current add MissionPlanMission command.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionAddCommandStatus

Data Type: MissionPlanMissionAddCommandStatusType

Table 28: MissionPlanMissionAddCommandStatusType Message Definition

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

6.1.6.3 reportMissionPlanMissionDeleteCommandAck

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionDeleteCommandAckReport

Data Type: MissionPlanMissionDeleteCommandAckReportType

Table 29: MissionPlanMissionDeleteCommandAckReportType Message Definition

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

6.1.6.4 reportMissionPlanMissionDeleteCommandStatus

Description: This operation provides the status of the current delete MissionPlanMission command.

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionDeleteCommandStatus

Data Type: MissionPlanMissionDeleteCommandStatusType

Table 30: MissionPlanMissionDeleteCommandStatusType Message Definition

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

6.1.6.5 setMissionPlanMissionAdd

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionAddCommand

Data Type: MissionPlanMissionAddCommandType

Table 31: 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.

6.1.6.6 setMissionPlanMissionDelete

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

Namespace: UMAA::MM::MissionPlanMissionControl

Topic: MissionPlanMissionDeleteCommand

Data Type: MissionPlanMissionDeleteCommandType

Table 32: 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.

6.1.7 MissionPlanObjectiveControl

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

Table 33: MissionPlanObjectiveControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanObjectiveAdd	reportMissionPlanObjectiveAddCommandStatus
queryMissionPlanObjectiveAddCommandAck⊕	reportMissionPlanObjectiveAddCommandAck
cancelMissionPlanObjectiveAddCommand⊕	reportMissionPlanObjectiveAddCancelCommandStatus⊕

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanObjectiveDelete	reportMissionPlanObjectiveDeleteCommandStatus
queryMissionPlanObjectiveDeleteCommandAck⊕	reportMissionPlanObjectiveDeleteCommandAck
cancelMissionPlanObjectiveDeleteCommand⊕	reportMissionPlanObjectiveDeleteCancelCommandStatus ⊕

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

6.1.7.1 reportMissionPlanObjectiveAddCommandAck

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

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveAddCommandAckReport

Data Type: MissionPlanObjectiveAddCommandAckReportType

Table 34: MissionPlanObjectiveAddCommandAckReportType Message Definition

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

6.1.7.2 reportMissionPlanObjectiveAddCommandStatus

Description: This operation provides the status of the current add mission plan objective command.

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveAddCommandStatus

Data Type: MissionPlanObjectiveAddCommandStatusType

Table 35: MissionPlanObjectiveAddCommandStatusType Message Definition

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

6.1.7.3 reportMissionPlanObjectiveDeleteCommandAck

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

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveDeleteCommandAckReport

Data Type: MissionPlanObjectiveDeleteCommandAckReportType

Table 36: MissionPlanObjectiveDeleteCommandAckReportType Message Definition

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

6.1.7.4 reportMissionPlanObjectiveDeleteCommandStatus

Description: This operation provides the status of the current remove mission plan objective command.

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveDeleteCommandStatus

Data Type: MissionPlanObjectiveDeleteCommandStatusType

Table 37: MissionPlanObjectiveDeleteCommandStatusType Message Definition

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

6.1.7.5 setMissionPlanObjectiveAdd

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

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveAddCommand

Data Type: MissionPlanObjectiveAddCommandType

Table 38: MissionPlanObjectiveAddCommandType Message Definition

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

Attribute Name	Attribute Type	Attribute Description
missionID†	NumericGUID	Specifies the missionID that the objective should be added. If not specified, it is unconstrained and left to the service provider to determine the mission. Typically, it is added to the current active mission.
objective	ObjectiveType	An objective to be added to a task of a mission.
taskID†	NumericGUID	Specifies the taskID that the objective should be added. If not specified, it is unconstrained and left to the service provider to determine the task. Typically, it is added to the current active task.

6.1.7.6 setMissionPlanObjectiveDelete

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

Namespace: UMAA::MM::MissionPlanObjectiveControl

Topic: MissionPlanObjectiveDeleteCommand

Data Type: MissionPlanObjectiveDeleteCommandType

Table 39: MissionPlanObjectiveDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
objectiveID*	NumericGUID	Specifies the identifier of the objective that is to be deleted.

6.1.8 MissionPlanReport

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

Table 40: 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 ⊕.

6.1.8.1 reportMissionPlan

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

Namespace: UMAA::MM::MissionPlanReport

Topic: MissionPlanReport

Data Type: MissionPlanReportType

Table 41: MissionPlanReportType Message Definition

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

6.1.9 MissionPlanTaskControl

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

Table 42: 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 ⊕.

6.1.9.1 reportMissionPlanTaskAddCommandAck

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

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskAddCommandAckReport

Data Type: MissionPlanTaskAddCommandAckReportType

Table 43: MissionPlanTaskAddCommandAckReportType Message Definition

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

6.1.9.2 reportMissionPlanTaskAddCommandStatus

Description: This operation provides the status of the current add MissionPlanTask command.

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskAddCommandStatus

Data Type: MissionPlanTaskAddCommandStatusType

Table 44: MissionPlanTaskAddCommandStatusType Message Definition

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

6.1.9.3 reportMissionPlanTaskDeleteCommandAck

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

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskDeleteCommandAckReport

Data Type: MissionPlanTaskDeleteCommandAckReportType

Table 45: MissionPlanTaskDeleteCommandAckReportType Message Definition

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

6.1.9.4 reportMissionPlanTaskDeleteCommandStatus

Description: This operation provides the status of the current delete MissionPlanTask command.

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskDeleteCommandStatus

Data Type: MissionPlanTaskDeleteCommandStatusType

Table 46: MissionPlanTaskDeleteCommandStatusType Message Definition

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

6.1.9.5 setMissionPlanTaskAdd

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

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskAddCommand

Data Type: MissionPlanTaskAddCommandType

Table 47: MissionPlanTaskAddCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
taskPlan	TaskPlanType	Specifies the task, which consists of objective(s), that is to be added to the mission plan.
missionID*	NumericGUID	Specifies the missionID that the task should be added. If not specified, it is unconstrained and left to the service provider to determine the mission. Typically, it is added to the current active mission.

6.1.9.6 setMissionPlanTaskDelete

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

Namespace: UMAA::MM::MissionPlanTaskControl

Topic: MissionPlanTaskDeleteCommand

Data Type: MissionPlanTaskDeleteCommandType

Table 48: MissionPlanTaskDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
taskID*	NumericGUID	Specifies the identifier of the task that is to be deleted.

6.1.10 ObjectiveAssignmentReport

The purpose of this service is to associate an objective to a particular resource or set of resources required to accomplish the objective.

Table 49: ObjectiveAssignmentReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryObjectiveAssignment⊕	reportObjectiveAssignment

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

6.1.10.1 reportObjectiveAssignment

Description: This operation is used to report the current assignment of an objective to a resource.

Namespace: UMAA::MM::ObjectiveAssignmentReport

Topic: ObjectiveAssignmentReport

Data Type: ObjectiveAssignmentReportType

Table 50: ObjectiveAssignmentReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
resourceIDs	sequence<NumericGUID> max size = 256	Identifies the resources that are assigned to a specific objective.
missionID*	NumericGUID	Identifies the associated mission plan.
objectiveID*	NumericGUID	Identifies the associated objective within a task plan of a mission plan.
taskID*	NumericGUID	Identifies the associated task plan within the mission plan.

6.1.11 ObjectiveExecutionControl

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

Table 51: 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: ObjectiveExecutionCommandAckReport

Data Type: ObjectiveExecutionCommandAckReportType

Table 52: 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: ObjectiveExecutionCommandStatus

Data Type: ObjectiveExecutionCommandStatusType

Table 53: 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: ObjectiveExecutionCommand

Data Type: ObjectiveExecutionCommandType

Table 54: ObjectiveExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
state	TaskControlEnumType	The desired state of the current objective.
missionID*	NumericGUID	The current mission plan identification.
objectiveID*	NumericGUID	The current objective identification within the current task plan.
taskID*	NumericGUID	The current task plan identification within the current mission plan.

6.1.12 ObjectiveExecutionStatus

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

Table 55: 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: ObjectiveExecutionReport

Data Type: ObjectiveExecutionReportType

Table 56: ObjectiveExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
objectiveStatuses→setID	LargeSet<ObjectiveStatusType>	Provides the current status of an objective. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::ObjectiveExecutionStatus::ObjectiveExecutionReportObjectiveStatusesSetElement.

6.1.13 TaskPlanAssignmentReport

The purpose of this service is to assign a task plan to a particular resource or set of resources required to accomplish the task plan.

Table 57: TaskPlanAssignmentReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryTaskPlanAssignment⊕	reportTaskPlanAssignment

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

6.1.13.1 reportTaskPlanAssignment

Description: This operation is used to provide the current assignment of a task plan to a resource.

Namespace: UMAA::MM::TaskPlanAssignmentReport

Topic: TaskPlanAssignmentReport

Data Type: TaskPlanAssignmentReportType

Table 58: TaskPlanAssignmentReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
resourceIDs	sequence<NumericGUID> max size = 256	Identifies the resources that are assigned to a specific task.
missionID*	NumericGUID	Identifies the associated mission plan.
taskID*	NumericGUID	Identifies the associated task within the mission plan.

6.1.14 TaskPlanExecutionControl

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

Table 59: 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 ⊕.

6.1.14.1 reportTaskPlanExecutionCommandAck

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

Namespace: UMAA::MM::TaskPlanExecutionControl

Topic: TaskPlanExecutionCommandAckReport

Data Type: TaskPlanExecutionCommandAckReportType

Table 60: TaskPlanExecutionCommandAckReportType Message Definition

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

6.1.14.2 reportTaskPlanExecutionCommandStatus

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

Namespace: UMAA::MM::TaskPlanExecutionControl

Topic: TaskPlanExecutionCommandStatus

Data Type: TaskPlanExecutionCommandStatusType

Table 61: TaskPlanExecutionCommandStatusType Message Definition

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

6.1.14.3 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: TaskPlanExecutionCommand

Data Type: TaskPlanExecutionCommandType

Table 62: TaskPlanExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
state	TaskControlEnumType	A desired state of the task plan.

Attribute Name	Attribute Type	Attribute Description
missionID*	NumericGUID	The mission plan identification.
taskID*	NumericGUID	The task plan identification within the current mission.

6.1.15 TaskPlanExecutionStatus

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

Table 63: 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 ⊕.

6.1.15.1 reportTaskPlanExecution

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

Namespace: UMAA::MM::TaskPlanExecutionStatus

Topic: TaskPlanExecutionReport

Data Type: TaskPlanExecutionReportType

Table 64: TaskPlanExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
taskPlanStatuses→setID	LargeSet<TaskPlanStatusType>	Describes the current status of each task plan. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::TaskPlanExecutionStatus::TaskPlanExecutionReportTaskPlanStatusesSetElement.

6.1.16 TriggerControl

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

Table 65: TriggerControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setTriggerAdd	reportTriggerAddCommandStatus
queryTriggerAddCommandAck⊕	reportTriggerAddCommandAck
cancelTriggerAddCommand⊕	reportTriggerAddCancelCommandStatus⊕

Service Requests (Inputs)	Service Responses (Outputs)
setTriggerDelete	reportTriggerDeleteCommandStatus
queryTriggerDeleteCommandAck⊕	reportTriggerDeleteCommandAck
cancelTriggerDeleteCommand⊕	reportTriggerDeleteCancelCommandStatus⊕

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

6.1.16.1 reportTriggerAddCommandAck

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

Namespace: UMAA::MM::TriggerControl

Topic: TriggerAddCommandAckReport

Data Type: TriggerAddCommandAckReportType

Table 66: TriggerAddCommandAckReportType Message Definition

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

6.1.16.2 reportTriggerAddCommandStatus

Description: This operation is used to provide the status of the current add trigger command.

Namespace: UMAA::MM::TriggerControl

Topic: TriggerAddCommandStatus

Data Type: TriggerAddCommandStatusType

Table 67: TriggerAddCommandStatusType Message Definition

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

6.1.16.3 reportTriggerDeleteCommandAck

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

Namespace: UMAA::MM::TriggerControl

Topic: TriggerDeleteCommandAckReport

Data Type: TriggerDeleteCommandAckReportType

Table 68: TriggerDeleteCommandAckReportType Message Definition

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

6.1.16.4 reportTriggerDeleteCommandStatus

Description: This operation is used to provide the status of the current delete trigger command.

Namespace: UMAA::MM::TriggerControl

Topic: TriggerDeleteCommandStatus

Data Type: TriggerDeleteCommandStatusType

Table 69: TriggerDeleteCommandStatusType Message Definition

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

6.1.16.5 setTriggerAdd

Description: This operation is used to add a new trigger.

Namespace: UMAA::MM::TriggerControl

Topic: TriggerAddCommand

Data Type: TriggerAddCommandType

Table 70: TriggerAddCommandType Message Definition

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

6.1.16.6 setTriggerDelete

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

Namespace: UMAA::MM::TriggerControl

Topic: TriggerDeleteCommand

Data Type: TriggerDeleteCommandType

Table 71: TriggerDeleteCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
triggerID*	NumericGUID	Specifies the identifier of the trigger to be deleted.

6.1.17 TriggerReport

The purpose of this service is to report the set of triggers that have successfully been added for use during mission plan execution.

Table 72: TriggerReport Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryTrigger⊕	reportTrigger

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

6.1.17.1 reportTrigger

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

Namespace: UMAA::MM::TriggerReport

Topic: TriggerReport

Data Type: TriggerReportType

Table 73: TriggerReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
triggers→setID	LargeSet<TriggerType>	Defines the current set of triggers 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::TriggerReport::TriggerReportTriggersSetElement.

6.1.18 VehicleMode

The purpose of this service is to define a state machine for various modes of vehicle operation.

Table 74: VehicleMode Operations

Service Requests (Inputs)	Service Responses (Outputs)
setVehicleMode	reportVehicleModeCommandStatus
queryVehicleModeCommandAck⊕	reportVehicleModeCommandAck
queryVehicleMode⊕	reportVehicleMode
queryVehicleModeSpecs⊕	reportVehicleModeSpecs

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

6.1.18.1 reportVehicleMode

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

Namespace: UMAA::MM::VehicleMode

Topic: VehicleModeReport

Data Type: VehicleModeReportType

Table 75: VehicleModeReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
mode	VehicleModeType	The mode that the vehicle is currently in.

6.1.18.2 reportVehicleModeCommandAck

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

Namespace: UMAA::MM::VehicleMode

Topic: VehicleModeCommandAckReport

Data Type: VehicleModeCommandAckReportType

Table 76: VehicleModeCommandAckReportType Message Definition

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

6.1.18.3 reportVehicleModeCommandStatus

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

Namespace: UMAA::MM::VehicleMode

Topic: VehicleModeCommandStatus

Data Type: VehicleModeCommandStatusType

Table 77: VehicleModeCommandStatusType Message Definition

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

6.1.18.4 reportVehicleModeSpecs

Description: This operation is used to report a list of available modes and which modes can be transitioned to from each mode, effectively defining a state machine.

Namespace: UMAA::MM::VehicleMode

Topic: VehicleModeSpecsReport

Data Type: VehicleModeSpecsReportType

Table 78: VehicleModeSpecsReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
modes	sequence<VehicleModeType> max size = 32	List of modes that the vehicle supports.

6.1.18.5 setVehicleMode

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

Namespace: UMAA::MM::VehicleMode

Topic: VehicleModeCommand

Data Type: VehicleModeCommandType

Table 79: VehicleModeCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
	Additional fields included from UMAA::UMAACommand	
mode	VehicleModeType	The mode that the vehicle should enter if it is a valid transition.

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 80: UCSMDEInterfaceSet Structure Definition

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

6.2.2 UMAACommand

Namespace: UMAA::UMAACommand

Description: Defines the common UMAA Command Message Fields.

Table 81: UMAACommand Structure Definition

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

6.2.3 UMAAStatus

Namespace: UMAA::UMAAStatus

Description: Defines the common UMAA Status Message Fields.

Table 82: UMAAStatus Structure Definition

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

6.2.4 UMAACommandStatusBase

Namespace: UMAA::UMAACommandStatusBase

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

Table 83: UMAACommandStatusBase Structure Definition

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

6.2.5 UMAACommandStatus

Namespace: UMAA::UMAACommandStatus

Description: Defines the common UMAA Command Status Message Fields.

Table 84: UMAACommandStatus Structure Definition

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

6.2.6 DateTime

Namespace: UMAA::Measurement::DateTime

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

Table 85: 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 86: 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 AirSpeedTolerance

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

Description: Defines the speed through air tolerance.

Table 87: AirSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
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.

6.2.9 AltitudeAGLType

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

Description: The height above ground level.

Table 88: AltitudeAGLType Structure Definition

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

6.2.10 AltitudeASFType

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

Description: The height above sea floor.

Table 89: AltitudeASFType Structure Definition

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

6.2.11 AltitudeGeodeticType

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

Description: The geodetic height above the ellipsoid.

Table 90: AltitudeGeodeticType Structure Definition

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

6.2.12 AltitudeMSLType

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

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

Table 91: AltitudeMSLType Structure Definition

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

6.2.13 ConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::ConstraintPlanType

Description: This structure defines common attributes across all Constraint Plans.

Table 92: ConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
name	StringShortDescription	Defines a short name for the constraint.
triggerID	NumericGUID	Defines a unique identifier of the trigger that enables the constraint.
constraintID*	NumericGUID	Defines a unique identifier for the constraint.

Table 93: ConstraintPlanType Subtypes

Type Name	Type Description
DepthRateConstraintPlanType	This structure defines a depth rate constraint of the vehicle while maneuvering.
EmitterPresetConstraintPlanType	This structure defines an EmitterPreset level constraint.
ExpConstraintType	This structure is used to define the an experimental constraint by specifying key/value pairs.
HeadingSectorConstraintPlanType	This structure defines a heading sector constraint that the vehicle must either avoid or maintain.

Type Name	Type Description
MaxDepthConstraintPlanType	This structure defines a maximum depth constraint.
MaxSpeedConstraintPlanType	This structure defines a maximum speed constraint of the vehicle while maneuvering.
MinDepthConstraintPlanType	This structure defines a minimum depth constraint.
MinSpeedConstraintPlanType	This structure defines a minimum speed constraint of the vehicle while maneuvering.
PitchRateConstraintPlanType	This structure defines a maximum pitch rate constraint of the vehicle while maneuvering.
ResourceConstraintPlanType	This structure defines a resource constraint, i.e., the resources that are allocated/assigned to achieve a particular goal (e.g., the resource(s) allocated to achieve a mission).
TurnRateConstraintPlanType	This structure defines a maximum turning rate constraint of the vehicle while maneuvering.
ZoneConstraintPlanType	This structure defines a zone constraint that the vehicle must keep in and/or keep out of while maneuvering.

6.2.14 ContingencyObjectiveType

Namespace: UMAA::MM::BaseType::ContingencyObjectiveType

Description: This structure is used to describe a contingency objective in case of emergency or lost communications with the control station.

Table 94: ContingencyObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	GeodeticAltitude	Specifies the distance along the vehicle path to the waypoint.
altitudeAGL	DistanceAGL	Specifies the distance above ground level.
altitudeASF	DistanceASF	Specifies the distance above sea level.
behavior	ContingencyBehaviorEnumType	Defines the system contingency level for the mission.
depth	DistanceBSL	Specifies the distance of the waypoint below sea level.
DTEDAltitude	Distance	Specifies DTED altitude at the origin.
mode	StringShortDescription	Specifies the navigation mode in which the vehicle is operating.
position	GeoPosition2D	Specifies the location for the vehicle to be at in case of emergency or lost communication.
radius	Distance	Specifies the radius the vehicle should be in.
safeAltitude	Distance	Specifies the altitude that is safe or within limit.
safeAltitudeOffset	Distance	Specifies the offset of safe.
speed	Speed	Specifies the speed of the vehicle.
vehicleRunTime	DurationHours	Specifies the duration that the vehicle can go.

6.2.15 DateTimeRequirement

Namespace: UMAA::Common::Requirements::DateTimeRequirement

Description: Realizes TimeRequirementType: a Requirement that specifies the type of time.

Table 95: DateTimeRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
time	DateTime	Describes the required time value.
timeTolerance	DateTimeTolerance	Describes the time tolerance.

6.2.16 DateTimeTolerance

Namespace: UMAA::Common::Measurement::DateTimeTolerance

Description: Realizes TimeToleranceType: an ObservableTolerance that specifies the range of allowable values for a time attribute.

Table 96: DateTimeTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
lowerLimit	DateTime	specifies the minimum value of the time point.
upperLimit	DateTime	specifies the maximum value of time point.

6.2.17 DeploymentObjectiveType

Namespace: UMAA::MM::BaseType::DeploymentObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for vehicle deployment.

Table 97: DeploymentObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	GeodeticAltitude	Specifies the distance along the vehicle path to the waypoint.
altitudeAGL	DistanceAGL	Specifies the distance above ground level.
altitudeASF	DistanceASF	Specifies the distance above sea level.
heading	HeadingTrueNorthAngle	Specifies the heading to be maintained during deployment.
position	GeoPosition2D	Specifies the location for deploying the vehicle.
releaseDepth	DistanceBSL	Specifies the distance below sea level for releasing the vehicle.
speed	GroundSpeed	Specifies the speed to be maintained during deployment.

6.2.18 DepthRateConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::DepthRateConstraintPlanType

Description: This structure defines a depth rate constraint of the vehicle while maneuvering.

Table 98: DepthRateConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
maxDepthRate	SpeedBSL	Defines the maximum depth rate that the vehicle is allowed to achieve while maneuvering.

6.2.19 DepthType

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

Description: Defines the depth below sea level.

Table 99: DepthType Structure Definition

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

6.2.20 DirectionCurrentRequirement

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

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

Table 100: DirectionCurrentRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	HeadingCurrentDirection	Specifies the heading offset angle relative to the current.
directionTolerance	DirectionToleranceType	Specifies the heading reference angle tolerance relative to the current.

6.2.21 DirectionMagneticNorthRequirement

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

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

Table 101: DirectionMagneticNorthRequirement Structure Definition

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

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

6.2.22 DirectionRequirementType

Namespace: UMAA::Common::Orientation::DirectionRequirementType

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

Table 102: DirectionRequirementType Union(s)

Type Name	Type Description
DirectionCurrentRequirement	A requirement that specifies the direction with respect to the current.
DirectionMagneticNorthRequirement	A requirement that specifies the direction with respect to magnetic north.
DirectionTrueNorthRequirement	A requirement that specifies the direction with respect to true north.
DirectionWindRequirement	A requirement that specifies the direction with respect to the direction of the wind.

6.2.23 DirectionToleranceType

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

Description: An angle tolerance associated with a direction.

Table 103: DirectionToleranceType Structure Definition

Attribute Name	Attribute Type	Attribute Description
lowerlimit	Angle	Describes the direction bound counterclockwise from the specified direction.
upperlimit	Angle	Describes the direction bound clockwise from the specified direction.

6.2.24 DirectionTrueNorthRequirement

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

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

Table 104: 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.

6.2.25 DirectionWindRequirement

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

Description: A requirement that specifies the direction with respect to the direction of the wind.

Table 105: DirectionWindRequirement Structure Definition

Attribute Name	Attribute Type	Attribute Description
direction	HeadingWindDirection	Specifies the heading reference angle relative to the wind direction.
directionTolerance	DirectionToleranceType	Specifies the heading reference angle tolerance relative to the wind direction.

6.2.26 DriftObjectiveType

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

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for drifting.

Table 106: DriftObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
driftTolerance	Distance	Defines the drift radius that specifies the maximum distance from the reference position the vehicle is allowed to drift.
duration†	DurationSeconds	Defines the desired duration to execute the pattern; if not specified, runs indefinitely until it is interrupted (e.g., another objective takes precedence, it is canceled, etc.).
elevation	ElevationType	The desired elevation of the vehicle.
position†	GeoPosition2DRequirement	Defines the reference position for drifting. When not specified, means at current location.
speed	SpeedControlType	The desired speed to return to the drift position when tolerance exceeded.
transitElevation	ElevationType	The elevation used while driving to the loiter track (vehicles must specify 0 as it is a required field).
transitSpeed	SpeedControlType	The speed at which one drives to the loiter track.

6.2.27 ElevationType

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

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

Table 107: ElevationType Union(s)

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

6.2.28 EmitterPresetConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::EmitterPresetConstraintPlanType

Description: This structure defines an EmitterPreset level constraint.

Table 108: EmitterPresetConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
levelID	NumericGUID	Defines a unique identifier of the EmitterPreset level.

6.2.29 EndTimeTriggerType

Namespace: UMAA::MM::Trigger::EndTimeTriggerType

Description: This structure defines an end time trigger. The conditional is true when the associated action(s) can be completed within the tolerance of the specified end time.

Table 109: EndTimeTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime	DateTimeRequirement	Defines the time that the associated action must be completed.

6.2.30 EngineRPM

Namespace: UMAA::Common::Speed::EngineRPM

Description: Defines the engine RPM.

Table 110: EngineRPM Structure Definition

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

6.2.31 EngineRPMSpeedRequirement

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

Description: Defines the engine rpm.

Table 111: 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.

6.2.32 EngineRPMSpeedTolerance

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

Description: Defines the speed through engine rpm.

Table 112: EngineRPMSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
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.

6.2.33 ExpConstraintType

Namespace: UMAA::MM::ConstraintPlan::ExpConstraintType

Description: This structure is used to define the an experimental constraint by specifying key/value pairs.

Table 113: ExpConstraintType Structure Definition

Attribute Name	Attribute Type	Attribute Description
expConstraintName	StringShortDescription	Defines a short name for the experimental constraint.
keyValues	sequence<KeyValueType> max size = 32	Defines a set of key/value pairs for the experimental constraint.

6.2.34 ExpObjectiveType

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

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

Table 114: ExpObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
expObjectiveDescription	StringShortDescription	Defines a short name for the experimental objective.
keyValues	sequence<KeyValueType> max size = 32	Defines a set of key/value pairs for the experimental objective.

6.2.35 ExpTriggerType

Namespace: UMAA::MM::Trigger::ExpTriggerType

Description: This structure is used to define the an experimental trigger by specifying key/value pairs.

Table 115: ExpTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
expTriggerName	StringShortDescription	Defines a short name for the experimental trigger.
keyValues	sequence<KeyValueType> max size = 32	Defines a set of key/value pairs for the experimental trigger.

6.2.36 Figure8ObjectiveType

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

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for loitering.

Table 116: Figure8ObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
crossTrackTolerance	Distance	The amount of error in position allowed from the pattern being executed.
duration†	DurationSeconds	Defines the desired duration to execute the pattern; if not specified, runs indefinitely until it is interrupted (e.g., another objective takes precedence, it is canceled, etc.).
elevation	ElevationType	The desired elevation of the vehicle.
length	Distance	Defines the length between the semicircles at either end of the figure 8 in which the vehicle should stay.
orientation	DirectionRequirementType	Defines the orientation of the figure 8, measured perpendicular to the length axis.
position†	GeoPosition2D	The desired loiter position of the vehicle. When not specified, means at current location.
radius	Distance	Defines the radius of the semicircles at either end of the figure 8 in which the vehicle should stay.
speed	SpeedControlType	The desired pattern execution speed of the vehicle.
transitElevation	ElevationType	The elevation used while driving to the loiter track (vehicles must specify 0 as it is a required field).
transitSpeed	SpeedControlType	The speed at which one drives to the loiter track.

Attribute Name	Attribute Type	Attribute Description
turnDirection	WaterTurnDirectionEnumType	The desired turn direction for the figure 8 pattern of the vehicle.

6.2.37 GeoPosition2D

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

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

Table 117: 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.

6.2.38 GeoPosition2DRequirement

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

Description: Defines a position requirement.

Table 118: GeoPosition2DRequirement Structure Definition

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

6.2.39 GeoPosition2DTolerance

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

Description: Defines a position tolerance.

Table 119: GeoPosition2DTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
limit	Distance	Specifies the limit of the tolerance.

6.2.40 GroundSpeedRequirement

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

Description: Defines the speed over ground.

Table 120: 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.

6.2.41 GroundSpeedTolerance

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

Description: Defines the speed over ground tolerance.

Table 121: GroundSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
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.

6.2.42 HeadingSectorConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::HeadingSectorConstraintPlanType

Description: This structure defines a heading sector constraint that the vehicle must either avoid or maintain.

Table 122: HeadingSectorConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
sector	sequence<HeadingSectorType> max size = 32	Defines the heading sector that the vehicle must either avoid or maintain.

6.2.43 HeadingSectorType

Namespace: UMAA::MM::ConstraintPlan::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 123: HeadingSectorType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endHeading	HeadingTrueNorthAngle	Defines the end heading of the defined sector.
headingSectorKind	HeadingSectorKindEnumType	Defines the type of heading sector, i.e., keep in, keep out.
startHeading	HeadingTrueNorthAngle	Defines the start heading of the defined sector.

6.2.44 HoverObjectiveType

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

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for hovering.

Table 124: HoverObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
controlPriority	HoverKindEnumType	The desired priority to hover at the specified point.
duration†	DurationSeconds	Defines the desired duration to execute the pattern; if not specified, runs indefinitely until it is interrupted (e.g., another objective takes precedence, it is canceled, etc.).
elevation	ElevationType	The desired elevation of the vehicle.
heading†	DirectionRequirementType	Defines the heading that the vehicle must maintain for hovering. When not specified, the system will determine the best heading (e.g. current heading, into the wind/current, etc.)
position†	GeoPosition2DRequirement	The desired hover position of the vehicle in the global coordinate system. When not specified, means at current location.
transitElevation	ElevationType	The elevation used while driving to the loiter track (vehicles must specify 0 as it is a required field).
transitSpeed	SpeedControlType	The speed at which one drives to the loiter track.

6.2.45 KeyValueType

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

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

Table 125: KeyValueType Structure Definition

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

6.2.46 LogicalANDTriggerType

Namespace: UMAA::MM::Trigger::LogicalANDTriggerType

Description: This structure defines a logical AND operator for a set of triggers.

Table 126: LogicalANDTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
triggerID1*	NumericGUID	Defines the first trigger that the logical AND operation is applied.
triggerID2*	NumericGUID	Defines the second trigger that the logical AND operation is applied.

6.2.47 LogicalNOTTriggerType

Namespace: UMAA::MM::Trigger::LogicalNOTTriggerType

Description: This structure defines a logical NOT operator for a trigger.

Table 127: LogicalNOTTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
notTriggerID*	NumericGUID	Defines the trigger that the logical NOT operation is applied.

6.2.48 LogicalORTriggerType

Namespace: UMAA::MM::Trigger::LogicalORTriggerType

Description: This structure defines a logical OR operator for a set of triggers.

Table 128: LogicalORTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
triggerID1*	NumericGUID	Defines the first trigger that the logical OR operation is applied.
triggerID2*	NumericGUID	Defines the second trigger that the logical OR operation is applied.

6.2.49 MaxDepthConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::MaxDepthConstraintPlanType

Description: This structure defines a maximum depth constraint.

Table 129: MaxDepthConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
maxDepth	DistanceBSL	Defines the maximum depth the vehicle is allowed to achieve.

6.2.50 MaxSpeedConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::MaxSpeedConstraintPlanType

Description: This structure defines a maximum speed constraint of the vehicle while maneuvering.

Table 130: MaxSpeedConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
maxSpeed	GroundSpeed	Defines the maximum speed that the vehicle is allowed to achieve while maneuvering.

6.2.51 MinDepthConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::MinDepthConstraintPlanType

Description: This structure defines a minimum depth constraint.

Table 131: MinDepthConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
minDepth	DistanceBSL	Defines the minimum depth the vehicle should maintain to begin operations.

6.2.52 MinSpeedConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::MinSpeedConstraintPlanType

Description: This structure defines a minimum speed constraint of the vehicle while maneuvering.

Table 132: MinSpeedConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
minSpeed	SpeedLocalWaterMass	Defines the minimum speed that the vehicle must maintain while maneuvering e.g., to enable control surfaces to operate.

6.2.53 MissionPlanStatusType

Namespace: UMAA::MM::MissionPlanExecutionStatus::MissionPlanStatusType

Description: This structure is used to report the status of the current mission plan.

Table 133: MissionPlanStatusType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime†	DateTimeRequirement	Provides the estimated (future time) or actual (past time) end time for the mission associated with missionID.

Attribute Name	Attribute Type	Attribute Description
feedback	StringShortDescription	Provides a reason for the current state of the Mission/Task plan (e.g. why the mission plan was rejected).
missionPlanDescription	StringShortDescription	Provides the description of the mission plan.
name	StringShortDescription	Provides the name of the mission plan.
startTime†	DateTimeRequirement	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.

6.2.54 MissionPlanType

Namespace: UMAA::MM::MissionPlanReport::MissionPlanType

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

Table 134: MissionPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
approvalRequired	boolean	An indication whether approval is required for the specified objective within a mission.
missionDescription	StringShortDescription	A description of the mission.
name	StringShortDescription	A short name for the mission.
priority	Priority	Specifies the execution priority for the objective.
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::MissionPlanReport::MissionPlanTaskPlansSetElement.
triggerID†	NumericGUID	Defines a unique identifier for the optional trigger used to initiate the execution of the specified mission plan.
missionID*	NumericGUID	Unique identifier for the mission.

6.2.55 MissionStateTriggerType

Namespace: UMAA::MM::Trigger::MissionStateTriggerType

Description: This structure defines a mission state trigger. The conditional is true when the current state of the specified mission is equal to the defined mission state.

Table 135: MissionStateTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
missionState	TaskStateEnumType	Specifies the state of the mission to be used in the conditional statement.
missionID*	NumericGUID	Identifies the mission to be used in the conditional statement.

6.2.56 ObjectiveStateTriggerType

Namespace: UMAA::MM::Trigger::ObjectiveStateTriggerType

Description: This structure defines an objective state trigger. The conditional is true when the current state of the specified objective is equal to the defined objective state.

Table 136: ObjectiveStateTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
objectiveState	TaskStateEnumType	Specifies the state of the objective to be used in the conditional statement.
objectiveID*	NumericGUID	Identifies the objective to be used in the conditional statement.

6.2.57 ObjectiveStatusType

Namespace: UMAA::MM::ObjectiveExecutionStatus::ObjectiveStatusType

Description: This structure is used to report the status of an objective within a task plan of a mission plan.

Table 137: ObjectiveStatusType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime†	DateTimeRequirement	Provides the estimated (future time) or actual (past time) end time for the objective associated with missionID, taskID, objectiveID.
startTime†	DateTimeRequirement	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.

6.2.58 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 138: ObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
approvalRequired	boolean	An indication whether approval is required for the specified objective within a mission.
childObjectiveIDs	sequence<NumericGUID> max size = 256	If the objective is decomposed into lower level objectives, specifies one or more unique identifiers of the children objectives that decompose the objective.
name	StringShortDescription	A short name for the objective.
objectiveDescription	StringShortDescription	A description of the objective.
parentObjectiveID†	NumericGUID	If the objective was decomposed from a high level objective, specifies the unique identifier of the parent objective from which it was decomposed.
priority	Priority	Specifies the execution priority for the objective.
triggerID†	NumericGUID	Defines a unique identifier for the optional trigger used to initiate the execution of the specified objective.
objectiveID*	NumericGUID	Unique identifier for the objective within a mission.

Table 139: ObjectiveType Subtypes

Type Name	Type Description
ContingencyObjectiveType	This structure is used to describe a contingency objective in case of emergency or lost communications with the control station.
DeploymentObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for vehicle deployment.
DriftObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for drifting.
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 loitering.
HoverObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for hovering.
RacetrackObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for following the racetrack pattern.
RecoveryObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for vehicle recovery.
RegularPolygonObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for following the polygon pattern.
RouteObjectiveType	This structure is used to report an element that describes a clearly defined goal specifying the action(s) required for route plans.
StationkeepObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for stationkeeping.

6.2.59 Orientation3DNEDRequirement

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

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

Table 140: 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.

6.2.60 PitchRateConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::PitchRateConstraintPlanType

Description: This structure defines a maximum pitch rate constraint of the vehicle while maneuvering.

Table 141: PitchRateConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
maxPitchRate	PitchRate	Defines the maximum pitch rate that the vehicle is allowed to achieve while maneuvering.

6.2.61 PitchYNEDRequirement

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

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

Table 142: 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.

6.2.62 PitchYNEDTolerance

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

Description: A down or up angle tolerance.

Table 143: PitchYNEDTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
lowerlimit	PitchYNEDType	Defines the steepest downangle allowed.

Attribute Name	Attribute Type	Attribute Description
upperlimit	PitchYNEDType	Defines the steepest upangle allowed.

6.2.63 PitchYNEDType

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

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

Table 144: PitchYNEDType Structure Definition

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

6.2.64 Polygon

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

Description: Specifies an area defined by a polygon.

Table 145: 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.

6.2.65 Quaternion

Namespace: BasicTypes::Quaternion

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

Table 146: Quaternion Structure Definition

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

6.2.66 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.

Table 147: RacetrackObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
crossTrackTolerance	Distance	The amount of error in position allowed from the pattern being executed.
duration†	DurationSeconds	Defines the desired duration to execute the pattern; if not specified, runs indefinitely until it is interrupted (e.g., another objective takes precedence, it is canceled, etc.).
elevation	ElevationType	The desired elevation of the vehicle.
length	Distance	Defines the length between the semicircles at either end of the racetrack in which the vehicle should stay.
orientation	DirectionRequirementType	Defines the orientation of the racetrack, measured perpendicular to the length axis.
position†	GeoPosition2D	The desired loiter position of the vehicle. When not specified, means at current location.
radius	Distance	Defines the radius of the semicircles at either end of the racetrack in which the vehicle should stay.
speed	SpeedControlType	The desired pattern execution speed of the vehicle.
transitElevation	ElevationType	The elevation used while driving to the loiter track (vehicles must specify 0 as it is a required field).
transitSpeed	SpeedControlType	The speed at which one drives to the loiter track.
turnDirection	WaterTurnDirectionEnumType	The desired turn direction for the racetrack pattern of the vehicle.

6.2.67 RecommendedSpeedControl

Namespace: UMAA::Common::Speed::RecommendedSpeedControl

Description: Defines the recommended speed mode.

Table 148: RecommendedSpeedControl Structure Definition

Attribute Name	Attribute Type	Attribute Description
recommendedSpeedControl	SpeedControlType	Specifies the recommended speed mode.

6.2.68 RecoveryObjectiveType

Namespace: UMAA::MM::BaseType::RecoveryObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for vehicle recovery.

Table 149: RecoveryObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	GeodeticAltitude	Specifies the distance along the vehicle path to the waypoint.
altitudeAGL	DistanceAGL	Specifies the distance above ground level.
altitudeASF	DistanceASF	Specifies the distance above sea level.
position	GeoPosition2D	Specifies the location for recovering the vehicle.

6.2.69 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.

Table 150: RegularPolygonObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
crossTrackTolerance	Distance	The amount of error in position allowed from the pattern being executed.
diameter	Distance	Defines the diameter of a circumscribed circle around the polygon.
duration†	DurationSeconds	Defines the desired duration to execute the pattern; if not specified, runs indefinitely until it is interrupted (e.g., another objective takes precedence, it is canceled, etc.).
elevation	ElevationType	The desired elevation of the vehicle.
numberSides	SidesCount	Defines the number of sides on the polygon.
orientation	DirectionRequirementType	Defines the orientation of the racetrack, measured perpendicular to the length axis.
position†	GeoPosition2D	The desired loiter position of the vehicle. When not specified, means at current location.
speed	SpeedControlType	The desired pattern execution speed of the vehicle.
transitElevation	ElevationType	The elevation used while driving to the loiter track (vehicles must specify 0 as it is a required field).
transitSpeed	SpeedControlType	The speed at which one drives to the loiter track.
turnDirection	WaterTurnDirectionEnumType	The desired turn direction for the polygon pattern of the vehicle.

6.2.70 RequiredSpeedControl

Namespace: UMAA::Common::Speed::RequiredSpeedControl

Description: Defines the required speed mode.

Table 151: RequiredSpeedControl Structure Definition

Attribute Name	Attribute Type	Attribute Description
requiredSpeedControl	SpeedControlType	Specifies the required speed mode.

6.2.71 ResourceConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::ResourceConstraintPlanType

Description: This structure defines a resource constraint, i.e., the resources that are allocated/assigned to achieve a particular goal (e.g., the resource(s) allocated to achieve a mission).

Table 152: ResourceConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
resourceIDs	sequence<NumericGUID> max size = 256	Defines one or more unique identifiers of the resources that are allocated for use.

6.2.72 RollXNEDRequirement

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

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

Table 153: 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.

6.2.73 RollXNEDTolerance

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

Description: A rotational tolerance.

Table 154: RollXNEDTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
lowerlimit	RollXNEDType	Defines the lower bound.
upperlimit	RollXNEDType	Defines the lower bound.

6.2.74 RollXNEDType

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

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

Table 155: RollXNEDType Structure Definition

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

6.2.75 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 route plans.

Table 156: 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::RouteObjectiveWaypointsListElement.

6.2.76 SpeedControlType

Namespace: UMAA::Common::Speed::SpeedControlType

Description: **Union Type**. Speed of the vehicle.

Table 157: SpeedControlType Union(s)

Type Name	Type Description
EngineRPM	Defines the engine RPM.
SpeedOverGround	Defines the speed over ground.
SpeedThroughAir	Defines the speed through air.
SpeedThroughWater	Defines the speed through water.
VehicleSpeedMode	Defines the speed mode.

6.2.77 SpeedOverGround

Namespace: UMAA::Common::Speed::SpeedOverGround

Description: Defines the speed over ground.

Table 158: SpeedOverGround Structure Definition

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

6.2.78 SpeedThroughAir

Namespace: UMAA::Common::Speed::SpeedThroughAir

Description: Defines the speed through air.

Table 159: SpeedThroughAir Structure Definition

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

6.2.79 SpeedThroughWater

Namespace: UMAA::Common::Speed::SpeedThroughWater

Description: Defines the speed through water.

Table 160: SpeedThroughWater Structure Definition

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

6.2.80 StartTimeTriggerType

Namespace: UMAA::MM::Trigger::StartTimeTriggerType

Description: This structure defines a start time trigger. The conditional is true when the current time is within the tolerance of the specified start time.

Table 161: StartTimeTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
startTime	DateTimeRequirement	Defines the time that the associated action must start.

6.2.81 StationkeepObjectiveType

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

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for stationkeeping.

Table 162: StationkeepObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
angleType	BearingAngleEnumType	Defines angle reference frame.
bearing	Angle	Defines bearing to contact for station keeping.
closingSpeed	GroundSpeed	Defines closingSpeed to contact for station keeping.
contactTrackID	NumericGUID	Defines contactTrackID for station keeping.
distance	Distance	Defines distance to contact for station keeping.
duration†	DurationSeconds	Defines duration for station keeping.

6.2.82 TaskPlanStatusType

Namespace: UMAA::MM::TaskPlanExecutionStatus::TaskPlanStatusType

Description: This structure is used to define the attributes for reporting status of a task plan within a mission plan.

Table 163: TaskPlanStatusType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime†	DateTimeRequirement	Specifies the estimated (future time) or actual (past time) end time for the task plan associated with missionID, taskID.
startTime†	DateTimeRequirement	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.

6.2.83 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 164: 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::TaskPlanObjectivesSetElement.
priority	Priority	Specifies the execution priority for the task.

Attribute Name	Attribute Type	Attribute Description
taskDescription	StringShortDescription	A description of the task.
triggerID†	NumericGUID	Defines a unique identifier for the optional trigger used to initiate the execution of the specified task plan.
taskID*	NumericGUID	Unique identifier for the task within a mission.

6.2.84 TaskStateTriggerType

Namespace: UMAA::MM::Trigger::TaskStateTriggerType

Description: This structure defines a task state trigger. The conditional is true when the current state of the specified task is equal to the defined task state.

Table 165: TaskStateTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
taskState	TaskStateEnumType	Specifies the state of the task to be used in the conditional statement.
taskID*	NumericGUID	Identifies the task to be used in the conditional statement.

6.2.85 TimePeriodTriggerType

Namespace: UMAA::MM::Trigger::TimePeriodTriggerType

Description: This structure defines a time period trigger. The conditional is true when the current time is within the tolerance of the specified start time and when the associated action(s) can be completed within the tolerance of the specified end time.

Table 166: TimePeriodTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime	DateTimeRequirement	Defines the time that the associated action must be completed.
startTime	DateTimeRequirement	Defines the time that the associated action must start.

6.2.86 TimeWithSpeed

Namespace: UMAA::Common::Speed::TimeWithSpeed

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

Table 167: TimeWithSpeed Structure Definition

Attribute Name	Attribute Type	Attribute Description
arrivalTime	DateTimeRequirement	Specifies the arrival time of the waypoint.
recommendedSpeed†	SpeedControlType	Specifies the recommended speed of the waypoint.

6.2.87 TriggerType

Namespace: UMAA::MM::Trigger::TriggerType

Description: This structure defines common attributes across all triggers.

Table 168: TriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
name	StringShortDescription	Defines a short name for the trigger.
triggerID*	NumericGUID	Defines a unique identifier for the trigger.

Table 169: TriggerType Subtypes

Type Name	Type Description
EndTimeTriggerType	This structure defines an end time trigger. The conditional is true when the associated action(s) can be completed within the tolerance of the specified end time.
ExpTriggerType	This structure is used to define the an experimental trigger by specifying key/value pairs.
LogicalANDTriggerType	This structure defines a logical AND operator for a set of triggers.
LogicalNOTTriggerType	This structure defines a logical NOT operator for a trigger.
LogicalORTriggerType	This structure defines a logical OR operator for a set of triggers.
MissionStateTriggerType	This structure defines a mission state trigger. The conditional is true when the current state of the specified mission is equal to the defined mission state.
ObjectiveStateTriggerType	This structure defines an objective state trigger. The conditional is true when the current state of the specified objective is equal to the defined objective state.
StartTimeTriggerType	This structure defines a start time trigger. The conditional is true when the current time is within the tolerance of the specified start time.
TaskStateTriggerType	This structure defines a task state trigger. The conditional is true when the current state of the specified task is equal to the defined task state.
TimePeriodTriggerType	This structure defines a time period trigger. The conditional is true when the current time is within the tolerance of the specified start time and when the associated action(s) can be completed within the tolerance of the specified end time.
ZoneTriggerType	This structure defines a zone trigger. The conditional is true when the vehicle location is contained within the defined zone.

6.2.88 TurnRateConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::TurnRateConstraintPlanType

Description: This structure defines a maximum turning rate constraint of the vehicle while maneuvering.

Table 170: TurnRateConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
maxTurnRate	TurnRate	Defines the maximum turning rate that the vehicle is allowed to achieve while maneuvering.

6.2.89 VariableSpeedControlType

Namespace: UMAA::Common::Speed::VariableSpeedControlType

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

Table 171: VariableSpeedControlType Union(s)

Type Name	Type Description
RecommendedSpeedControl	Defines the recommended speed mode.
RequiredSpeedControl	Defines the required speed mode.
TimeWithSpeed	Defines the time window and the recommended speed of a vehicle.

6.2.90 VehicleModeType

Namespace: UMAA::MM::VehicleMode::VehicleModeType

Description: Data structure that defines a mode and which modes can be transitioned to.

Table 172: VehicleModeType Structure Definition

Attribute Name	Attribute Type	Attribute Description
mode	StringShortDescription	The mode that can be commanded or reported.
transitions	sequence<StringShortDescription> max size = 32	A list of other modes that can be transitioned to from this mode.

6.2.91 VehicleSpeedMode

Namespace: UMAA::Common::Speed::VehicleSpeedMode

Description: Defines the speed mode.

Table 173: VehicleSpeedMode Structure Definition

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

6.2.92 WaterSpeedRequirement

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

Description: Defines the speed through water.

Table 174: 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.

6.2.93 WaterSpeedTolerance

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

Description: Defines the speed through water tolerance.

Table 175: WaterSpeedTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
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.

6.2.94 WaterspaceVolumeType

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

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

Table 176: WaterspaceVolumeType Structure Definition

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

6.2.95 WaypointType

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

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

Table 177: WaypointType Structure Definition

Attribute Name	Attribute Type	Attribute Description
attitude†	Orientation3DNEDRequirement	Describes the yaw, pitch, roll that the vehicle should assume as arriving at the given waypoint.
elevation†	ElevationType	The optional elevation used for the vehicle.
name†	StringShortDescription	A short name for the waypoint.
position	GeoPosition2DRequirement	Specifies the location of the waypoint.
speed†	VariableSpeedControlType	Specifies the speed to be maintained traveling to the waypoint.
trackTolerance†	Distance	The current tolerance of the path measured by distance.
waypointID*	NumericGUID	An unique identification of the waypoint.

6.2.96 YawZNEDRequirement

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

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

Table 178: 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.

6.2.97 YawZNEDTolerance

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

Description: A directional tolerance.

Table 179: YawZNEDTolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
lowerlimit	YawZNEDType	Defines the lower bound.
upperlimit	YawZNEDType	Defines the lower bound.

6.2.98 YawZNEDType

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

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

Table 180: YawZNEDType Structure Definition

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

6.2.99 ZoneConstraintPlanType

Namespace: UMAA::MM::ConstraintPlan::ZoneConstraintPlanType

Description: This structure defines a zone constraint that the vehicle must keep in and/or keep out of while maneuvering.

Table 181: ZoneConstraintPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
zone→setID	LargeSet<ZoneType>	Defines the vehicle keep-in and keep-out operating zones. This attribute is implemented as a large set, see subsection 3.8 for an explanation. The associated topic is UMAA::MM::ConstraintPlan::ZoneConstraintPlanZoneSetElement.

6.2.100 ZoneTriggerType

Namespace: UMAA::MM::Trigger::ZoneTriggerType

Description: This structure defines a zone trigger. The conditional is true when the vehicle location is contained within the defined zone.

Table 182: ZoneTriggerType Structure Definition

Attribute Name	Attribute Type	Attribute Description
zone	WaterspaceVolumeType	Defines the zone.

6.2.101 ZoneType

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

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

Table 183: ZoneType Structure Definition

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

6.3 Enumerations

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

6.3.1 BearingAngleEnumType

Namespace: UMAA::Common::MaritimeEnumeration::BearingAngleEnumType

Description: Defines a mutually exclusive set of values for the type of bearing angle.

Table 184: BearingAngleEnumType Enumeration

Enumeration Value	Description
NORTH	Angle is relative to true north
OWNSHIP	Angle is relative to ownship

6.3.2 CommandStatusReasonEnumType

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

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

Table 185: 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.

6.3.3 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 186: 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 HeadingSectorKindEnumType

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

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

Table 187: HeadingSectorKindEnumType Enumeration

Enumeration Value	Description
KEEP_IN	The heading sector kind is keep in.
KEEP_OUT	The heading sector kind is keep out.

6.3.5 HoverKindEnumType

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

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

Table 188: HoverKindEnumType Enumeration

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

6.3.6 LineSegmentEnumType

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

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

Table 189: 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).

6.3.7 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 190: 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.

6.3.8 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 191: TaskControlEnumType Enumeration

Enumeration Value	Description
ALLOCATE	Allocate the mission plan, mission task, or mission objective.
ALLOCATION_APPROVED	Approve the allocation of the mission plan, mission task, or mission objective.
ALLOCATION_NOT_APPROVED	Reject the allocation of the mission plan, mission task, or mission objective.
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.

Enumeration Value	Description
PAUSE	Pause the execution of the approved mission plan, mission task, or mission objective.
PLAN	Plan the allocated mission plan, mission task, or mission objective.
PLAN_APPROVED	Approve the planning of the mission plan, mission task, or mission objective.
PLAN_NOT_APPROVED	Reject the planning of the mission plan, mission task, or mission objective.
QUEUE	Queue the mission plan, mission task, or mission objective for execution.
RESUME	Resume the execution of the mission plan, mission task, or mission objective.
VALIDATE	Validate the mission plan, mission task, or mission objective.

6.3.9 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 192: TaskStateEnumType Enumeration

Enumeration Value	Description
ALLOCATED	The mission plan, a mission task, or mission objective has been allocated and approval has been granted.
ALLOCATED_PENDING_APPROVAL	The mission plan, mission task, or mission objective allocation has been completed and approval is pending.
ALLOCATING	The mission plan, mission task, or mission objective is currently being allocated. The allocation has not completed.
AWAITING_EXECUTION_APPROVAL	The mission plan, mission task, or mission objective is awaiting execution approval.
AWAITING_MISSION_PLAN	The initial state when there is no mission plan reported.
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.).
DROPPED	The mission plan, mission task, or mission objective has been dropped and is subject for reallocation within the UxS node.
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.
NOT_PLANNED	The mission plan, mission task, or mission objective has not been planned.

Enumeration Value	Description
NOT_QUEUED	The mission plan, mission task, or mission objective has not been queued for execution.
NOT_VALIDATED	The mission plan, mission task, or mission objective has not been validated.
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.
PLANNED_PENDING_APPROVAL	The mission plan, mission task, or mission objective is pending approval.
PLANNING	The mission plan, mission task, or mission objective is still in the planning state.
PROPOSED	The mission plan, mission task, or mission objective has been proposed to an allocation service.
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
RESUMING	The mission plan, mission task, or mission objective is in the process of being resumed.
UNALLOCATED	The mission plan, mission task, or mission objective has been unallocated. The mission plan, mission task, or mission objective could not be allocated to a system, or it has just been created.
VALIDATED	The mission plan, mission task, or mission objective has been validated.
VALIDATING	The mission plan, mission task, or mission objective is in the process of being validated.

6.3.10 VehicleSpeedModeEnumType

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

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

Table 193: 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

6.3.11 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 194: WaterTurnDirectionEnumType Enumeration

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

6.3.12 ZoneKindEnumType

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

Description: Defines a mutually exclusive set of zone kinds.

Table 195: ZoneKindEnumType Enumeration

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

6.4 Type Definitions

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

Table 196: Type Definitions

Type Name	Primitive Type	Range of Values	Description

Angle	double	fractionDigits=3
		maxInclusive=3.1415926535897932384626433832795
		minInclusive=-3.1415926535897931264626433832795
		units=Radian
		referenceFrame=Counting
			Specifies the amount of turning necessary to bring one ray, line or plane into coincidence with or parallel to another. The measurement is stated in radians between -pi and pi.

BooleanEnumType	boolean		A mutually exclusive set of values that defines the truth values of logical algebra.

DateTimeNanoseconds	long	units=Nanoseconds
		minInclusive=0
		maxInclusive=999999999
		fractionDigits=0
			number of nanoseconds elapsed within the current second.

DateTimeSeconds	longlong	units=Seconds
		minInclusive=-9223372036854775807
		maxInclusive=9223372036854775807
		fractionDigits=0
			seconds offset from the standard POSIX (IEEE Std 1003.1-2017) epoch reference point of January 1st, 1970 00:00:00 UTC.

Distance	double	fractionDigits=3
		maxInclusive=401056000
		minInclusive=0
		units=Meter
		referenceFrame=Counting
			This type stores a distance in meters.

DistanceAGL	double	fractionDigits=3
		minInclusive=0.0
		units=Meter
		referenceFrame=AGL
			Describes the height above ground level of the vehicle.

DistanceASF	double	fractionDigits=3
		maxInclusive=401056000
		minInclusive=0
		units=Meter
		referenceFrame=ASF
			The altitude or distance above the sea floor in meters.

DistanceBSL	double	fractionDigits=3
		maxInclusive=10000
		minInclusive=0
		units=Meter
		referenceFrame=BSL
			The distance below sea level in meters.

DurationHours	double	fractionDigits=3
		maxInclusive=10505
		minInclusive=0
		units=Hour
		referenceFrame=Counting
			Represents a time duration in hours.

Type Name	Primitive Type	Range of Values	Description

DurationSeconds	double	fractionDigits=6
		maxInclusive=37817280
		minInclusive=0
		units=Seconds
		referenceFrame=Counting
			Represents a time duration in seconds.

FrequencyRPM	long	fractionDigits=0
		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	fractionDigits=6
		maxInclusive=700000
		minInclusive=-10000
		units=Meter
		axisAbbrev=Altitude
		axisDirection=up
		axisUnit=Meter
		rangeMeaning=exact
		resolution=0.0000000001
			Used for measuring position and increases in magnitude as position extends upward. Altitude measurements are expressed in meters.

GeodeticLatitude	double	axisAbbrev=Latitude
		axisDirection=north/south
		axisUnit=Degrees
		maximumValue=90.0
		minimumValue=-90.0
		rangeMeaning=exact
		resolution=0.0000000001
			Used for measuring position and increases in magnitude as position extends from the south pole to the north pole. Latitude measurements are expressed in degrees.

GeodeticLongitude	double	axisAbbrev=Longitude
		axisDirection=east
		axisUnit=Degrees
		maximumValue=180.0
		minimumValue=-180.0
		rangeMeaning=wraparound
		resolution=0.0000000001
			Used for measuring position and increases in magnitude as position extends eastward. Longitude measurements are expressed in degrees. Longitude measurements are periodic and whose limits (min and max), while mathematically discontinuous, represent a continuous range.

GroundSpeed	double	fractionDigits=3
		maxInclusive=299,792,458
		minInclusive=-299,792,458
		units=MeterPerSecond
		referenceFrame=TrueNorth
			The magnitude of the horizontal velocity vector of an aircraft relative to the ground.

HeadingCurrentDirection	double	fractionDigits=3
		maxInclusive=3.142
		minInclusive=-3.142
		units=Radian
		referenceFrame=CurrentDirection
			Describes heading as a value between -pi and pi with respect to the current direction.

HeadingMagneticNorth	double	fractionDigits=3
		maxInclusive=3.142
		minInclusive=-3.142
		units=Radian
		referenceFrame=MagneticNorth
			Describes heading as a value between -pi and pi with respect to Magnetic North.

Type Name	Primitive Type	Range of Values	Description

HeadingTrueNorthAngle	double	fractionDigits=3
		maxInclusive=3.142
		minInclusive=-3.142
		units=Radian
		referenceFrame=TrueNorth
			Describes heading as a value between -pi and pi with respect to True North.

HeadingWindDirection	double	fractionDigits=3
		maxInclusive=3.142
		minInclusive=-3.142
		units=Radian
		referenceFrame=WindDirection
			Describes heading as a value between -pi and pi with respect to the wind direction.

IndicatedAirspeed	double	fractionDigits=6
		maxInclusive=299,792,458
		minInclusive=0
		units=MeterPerSecond
		referenceFrame=LocalAirMass
			This type specifies the magnitude of an aircraft's velocity (the rate of change of its position). Indicated airspeed (IAS) is the airspeed read directly from the airspeed indicator on an aircraft, driven by the pitot-static system.

LargeCollectionSize	long	fractionDigits=0
		maxInclusive=2147483647
		minInclusive=0
		units=N/A
			Specifies the size of a Large Collection.

MSLAltitude	double	fractionDigits=3
		minInclusive=0.0
		units=Meter
		referenceFrame=Altitude
			Describes the current orthometric height above the Geoid (Mean Sea Level).

NumericGUID	octet[16]	units=N/A
		minInclusive=0
		maxInclusive=(2^128)-1
		fractionDigits=0
			Represents a 128-bit number according to RFC 4122 variant 2.

PitchHalfAngle	double	fractionDigits=3
		maxInclusive=1.571
		minInclusive=-1.571
		units=Radian
		referenceFrame=PlatformNED
			Specifies the platform's rotation about the lateral axis (e.g. the axis parallel to the wings) in a locally level, North-East-Down coordinate system centered on the platform. Pitch is zero when the platform is "nose to tail level" in the North-East plane. The measurement is stated in radians between -0.5 pi and 0.5 pi.

PitchRate	double	fractionDigits=3
		maxInclusive=32.767
		minInclusive=0
		units=RadianPerSecond
		referenceFrame=Counting
			Specifies the rate of change of the platform's pitch angle relative to a NED frame centered at the platform location.

Priority	long	fractionDigits=0
		maxInclusive=255
		minInclusive=0
			Represents the priority as a positive integer. Low numbers represent low priority while higher numbers represent high priority.

Type Name	Primitive Type	Range of Values	Description

RollAngle	double	fractionDigits=3
		maxInclusive=3.142
		minInclusive=-3.142
		units=Radian
		referenceFrame=PlatformNED
			Specifies a platform's rotation about the longitudinal axis (e.g. the axis through the body of the vehicle from tail to nose) in a locally level, North-East-Down coordinate system centered on the vehicle. Roll is zero when the platform is "wing-tip to wing-tip" level in the North-East plane. The measurement is stated in radians between -pi and pi.

SidesCount	long	fractionDigits=0
		maxInclusive=255
		minInclusive=3
		units=N/A
			Represents the number of sides a polygon has using a positive integer.

Speed	double	fractionDigits=6
		maxInclusive=299,792,458
		minInclusive=0
		units=MeterPerSecond
		referenceFrame=Counting
			This type stores speed in meters/s.

SpeedBSL	double	fractionDigits=3
		maxInclusive=299,792,458
		minInclusive=-299,792,458
		units=MeterPerSecond
		referenceFrame=BSL
			This type stores speed in meters/s in a below sea level reference frame.

SpeedLocalWater Mass	double	fractionDigits=6
		maxInclusive=299,792,458
		minInclusive=0
		units=MeterPerSecond
		referenceFrame=LocalWaterM ass
			This type stores speed in meters/s.

StringLongDescrip tion	string	length=4095
		units=N/A
		minInclusive=N/A
		maxInclusive=N/A
			Represents a long format description.

StringShortDescri ption	string	length=1023
		units=N/A
		minInclusive=N/A
		maxInclusive=N/A
			Represents a short format description.

TurnRate	double	fractionDigits=3
		maxInclusive=32.767
		minInclusive=0
		units=RadianPerSecondreferen ce
		referenceFrame=Counting
			Specifies the rate of change of the heading angle of a platform.

YawPosAngle	double	fractionDigits=3
		maxInclusive=6.283185307179 586364925286766559
		minInclusive=0
		units=Radian
		referenceFrame=PlatformNED
			The yaw angle relative to the NED coordinate system centered at the platform location.

A Appendices

A.1 Glossary

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

Almanac Data (GPS)
A navigation message that contains information about the time and status of the entire satellite constellation.

Coulomb
The SI unit of electric charge, equal to the quantity of electricity conveyed in one second by a current of one ampere.

Ephemeris Data (GPS)
A navigation message used to calculate the position of each satellite in orbit.

Glowplug or Glow Plug
A heating device used to aid in starting diesel engines.

Interoperability
1) The ability to act together coherently, effectively, and efficiently to achieve tactical, operational, and strategic objectives. 2) The condition achieved among communications-electronics systems or items of communications-electronics equipment when information or services can be exchanged directly and satisfactorily between them and/or their users.

Mean Sea Level
The average height of the surface of the sea for all stages of the tide; used as a reference for elevations.

Middleware
A type of computer software that provides services to software applications beyond those available from the operating system. Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.

SoaML
The Service oriented architecture Modeling Language (SoaML) specification that provides a metamodel and a UML profile for the specification and design of services within a service-oriented architecture. The specification is managed by the Object Management Group (OMG).

A.2 Acronyms

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

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

ID	Identifier
IDL	Interface Definition Language Specification
IMO	International Maritime Organization
INU	Inertial Navigation Unit
LDM	Logical Data Model
LOA	Length Over All
LRC	Long Range Cruise
LWL	Length at Waterline
MDE	Maritime Domain Extensions
MEC	Maximum Endurance Cruise
MM	Mission Management
MMSI	Maritime Mobile Service Identity
MO	Maneuver Operations
MRC	Maximum Range Cruise
MSL	Mean Sea Level
OMG	Object Management Group
PIM	Platform Independent Model
PMC	Primary Mission Control
PNT	Precision Navigation and Timing
PO	Processing Operations
PSM	Platform Specific Model
RMS	Root-Mean-Square
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