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

MDE Version 4.2.1 Commit 7c2b513

UMAA Spec Commit f2fa426

Version 3.0.1
25 February 2021

UMAA Mission Management ICD

(UMAA-SPEC-MM-ICD)

Signature Page

Submitted by: Rothgeb.Mark.E.ORC3011008580.ID — Digitally signed by Rothgeb.Mark.E.ORC3011008580.ID Date: 2021.05.05 11:50:23 -04'00' **Date Signed:** _____

Mark Rothgeb
Unmanned Maritime Autonomy Architecture
Standards Board Chair

PMS 406 Approvals:

ANDERSON.JEREMIAH.P.1254266481 — Digitally signed by ANDERSON.JEREMIAH.P.1254266481 Date: 2021.05.10 14:50:37 -04'00' **Date Signed:** _____

CDR Jeremiah Anderson
PMS 406 Advanced Autonomous Capabilities,
Principal Assistant Program Manager

SMALL.PETER.D.1048778646 — Digitally signed by SMALL.PETER.D.1048778646 Date: 2021.06.03 17:16:45 -04'00' **Date Signed:** _____

CAPT Pete Small
Program Manager
PMS 406 Unmanned Maritime Systems,
PEO Unmanned and Small Combatants

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 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 UMVs.

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

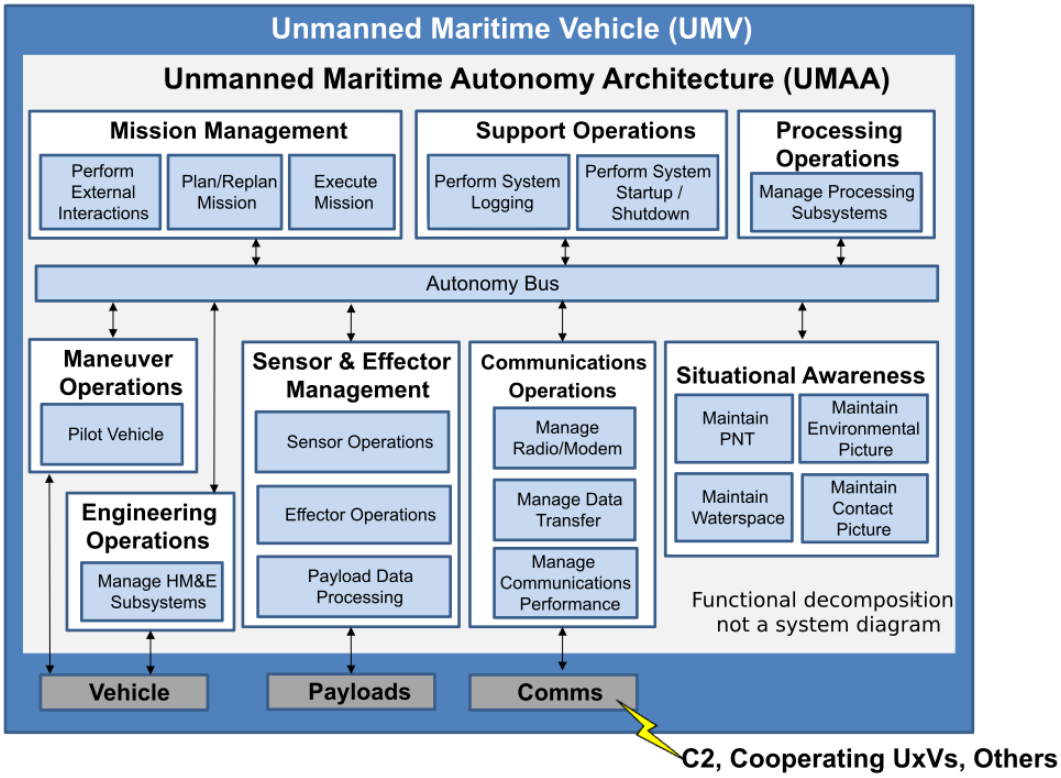

Figure 1: UMAA Functional Organization

1.2 Overview

The fundamental purpose of UMAA is to promote the development of common, modular, and scalable software for UMV's that is independent of a particular autonomy implementation. Unmanned Maritime Systems (UMSs) consist of Command and Control (C2), one or more UMVs, 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 UMV. This includes the autonomy for all classes of UMVs and must support varying levels of communication in mission (i.e., constant, intermittent, or none) with its C2 System. 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 platform interface.

Mission Management defines the services required to specify an UMV mission. Figure 2 depicts an example of a possible component service grouping is shown with the dashed lines.

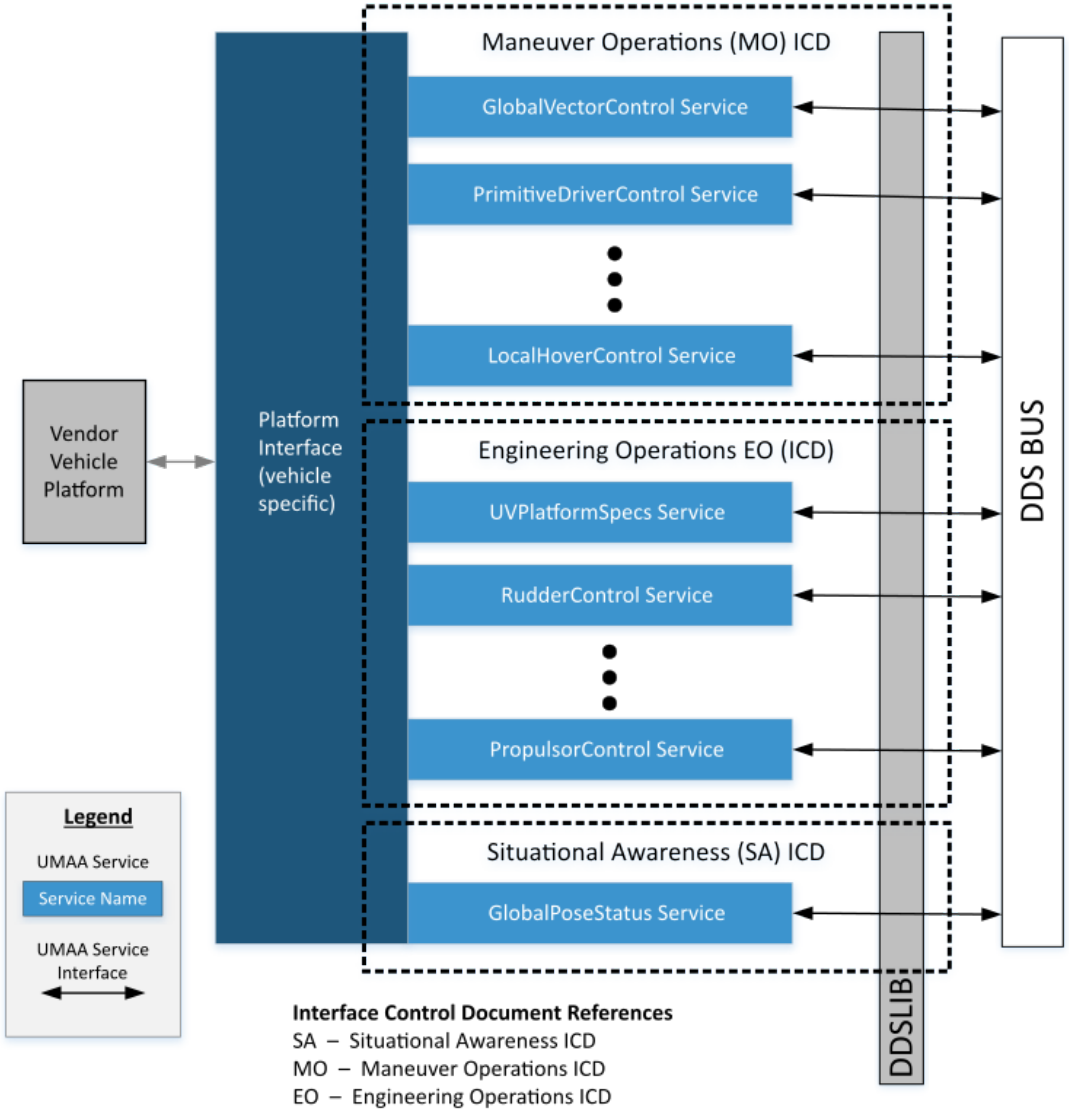

Figure 2: UMAA Services and Interfaces Example

1.3 Document Organization

This interface control document is organized as follows:

Section 1 – Scope: A brief purview of this document

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

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

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

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

Section 6 – Mission Management (MM) Services and Interfaces: A description of specific services and interfaces for this ICD

2 Referenced Documents

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

Table 3: 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

Table 4: 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, a "data in motion" view of system level information.

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

The data model is maintained as a maritime extension to the UCS Architecture and will be maintained under configuration control by the UMAA Board. 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 compliance with a UMAA ICD is to provide service and interface specification compliance. Components may group services and their 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 represent 1 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. Compliance is satisfied at the individual service level. Given this context, section 6 correspondingly defines services with their interfaces and not components.

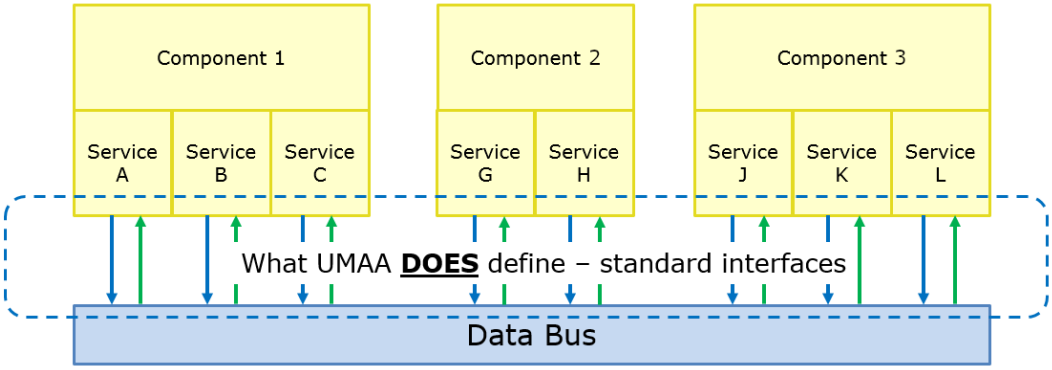

Figure 3: Services and Interfaces Exposed on the UMAA Data Bus

Services may use other services within this ICD or in other UMAA defined ICDs in order to provide their capability. Additionally, components for acquisition and development may span ICDs. An example of this would be a vehicle control system on a UMV. The control of the vehicle would be found in the Maneuver Operations ICD. However, an Inertial Navigation Unit (INU) that gives dynamic vehicle status is found in the Situational Awareness ICD. These are often organic to a vehicle and in that case are provided together with the vehicle as a component.

3.3 Data Distribution Service (DDS™)

The data bus supporting autonomy messaging as depicted in figure 3 is implemented via DDS™. DDS is a middleware protocol and API standard for data-centric connectivity from the Object Management Group (OMG). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture. In a distributed system, middleware is the software layer that lies between the operating system and applications. It enables the various components of a system 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 a 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. (UCSArchitecture,ArchitectureTechnicalGovernance)

In order to promote commonality across service definitions, a common way of naming services and their set 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 will not be broken up as indicated below 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 in implementation. The conventions used to define UCS-MDE services are as follows:

Service Name
- <SBN>Config
- <SBN>Control
- <SBN>Specs
- <SBN>Status

where the SBN should be descriptive of the task or information provided by the service.

Table 5: Service Requests and Associated Responses

	Service Requests (Inputs)	**Service Responses (Outputs)**
Config	query<SBN><FN>Config	report<SBN><FN>Config
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	query<SBN><FN>	report<SBN><FN>

Service Requests (operation:message)
- 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
- query<SBN><FN>ExecutionStatus:<SBN><FN>ExecutionStatusRequestType[1]
- query<SBN><FN>Specs:<SBN><FN>SpecsRequestType[1]
- query<SBN><FN>:<SBN><FN>RequestType [1] [2]

[1]These message types are required for compatibility with the UCS model but are not used by the UMAA specification.
[2]At this time there are no Requests in the specification but when they have been added, this will be the message format.

Service Responses (operation:message)
report<SBN><FN>Config:<SBN><FN>ConfigReportType
report<SBN><FN>CommandStatus:<SBN><FN>CommandStatusType
report<SBN><FN>CommandAck:<SBN><FN>CommandAckReportType
report<SBN><FN>CancelCommandStatus:<SBN><FN>CancelCommandStatusType
report<SBN><FN>ExecutionStatus:<SBN><FN>ExecutionStatusReportType
report<SBN><FN>Specs:<SBN><FN>SpecsReportType
report<SBN><FN>:<SBN><FN>ReportType

where,

- Config (Configuration) Report – 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, maximum allowable radio transmit power.
- Command Status – the current state of a particular command (either control or configuration)
- Command – the ability to influence or direct the behavior of a resource during operation of a particular task. Attributes are variable. Examples include a vehicleâ€™s speed, engine RPM, antenna raising/lowering, controlling a light or gong.
- Command Ack (Acknowledgement) Report – the command currently being executed.
- Cancel – the ability to cancel a particular command that has been issued.
- Execution Status Report – the status related to executing a particular command. Examples associated with a waypoint command include cross track error, time to achieve, distance remaining.
- Specs (Specifications) Report – 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 UMV, cycle time of a radar.
- Report – the current information provided by a resource. Examples include a vehicle speed, rudder angle, current waypoint, 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 service under Maneuver Operations:
UMAA::MO::PrimitiveDriver
Access the Feature Service under Situational Awareness:
UMAA::SA::Feature

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 Report Message Fields:
UMAA::UMAARpt
Access the common UMAA Position2D (i.e., latitude and longitude) structure:
UMAA::Measurement::Position2D

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 Sets

Some reports under the UMAA standard utilize Large Sets, which are unordered sets of related data. The purpose of a Large Set is to provide the ability to update one or more elements of the set without having to republish the entire set on the DDS bus and consuming more resources as a set is appended or updated. In a given DDS topic, each element of the set is tracked to its identifier through the use of the <service>SetID identifier (a key). Additionally, users will be able to trace an element in a set by its source attribute (a NumericGUID) to the Service Provider that is generating the report with this set.

When elements of the set are updated, the timestamp of the metadata must be updated as well to signal a change in the set. The element timestamp for the update must be later than the current metadata timestamp. Once the element is updated, the timestamp of the metadata must be updated to a time equal to or later than the timestamp of the individual element update. The set can be updated as a batch (multiple elements in a single "update cycle," as determined by the provider) provided the metadata timestamp is updated to a time that is no earlier than the the most recent timestamp of all element updates in the batch. This allows for a coarse synchronization: data elements with timestamps later than the current metadata timestamp 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 metadata timestamp is advanced beyond the element's timestamp to signal the complete update cycle has finished and consider the set as a whole.

4 Introduction to Coordinate Reference Frames and Position Model

4.1 Platform Reference Frame

In the following Service Definitions we use the parameters yaw, pitch, and roll to define the orientation of the vehicle 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.

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 Platform Orientation

Determining the orientation of the vehicle (Figure 4) with respect to any reference frame is carried out via the following procedure (Figure 5).

1. Align the vehicle's Longitudinal or X axis with the reference frame X axis. In the global reference, this is the North direction.
2. Align the vehicle's down or 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 6).
5. Rotate the vehicle about the vehicle's newly oriented Y axis by the pitch angle (Figure 7).
6. Rotate the vehicle about the vehicle's newly oriented X axis by the roll angle (Figure 8).

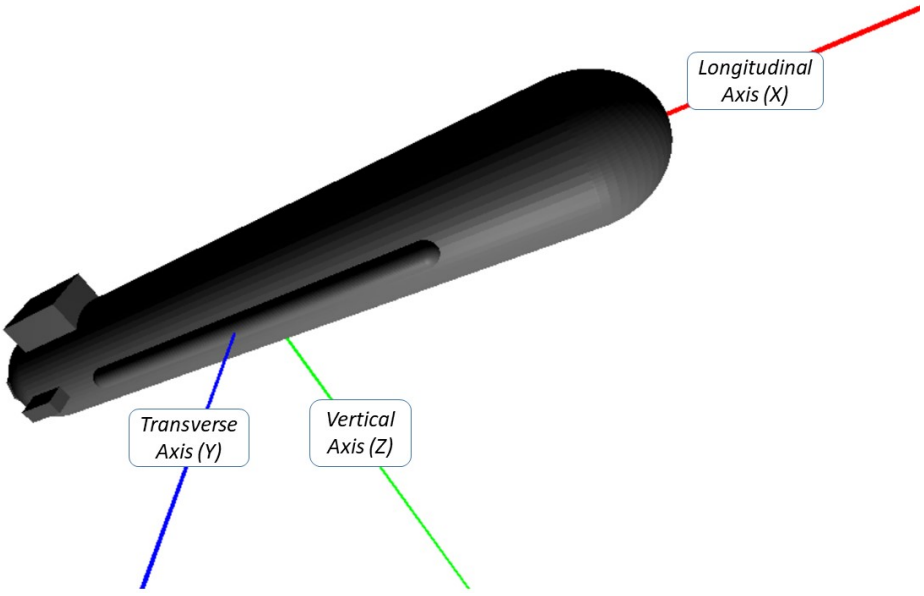

Figure 4: Given a vehicle in arbitrary orientation

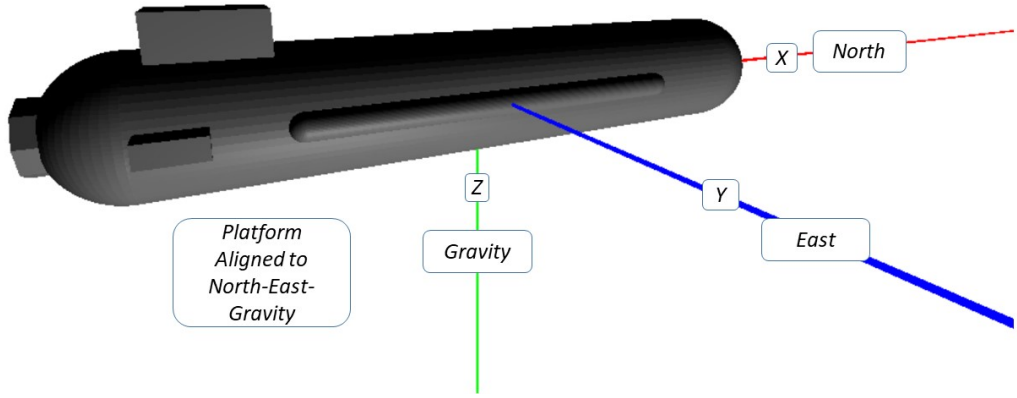

Figure 5: Align the vehicle with the reference frame axes

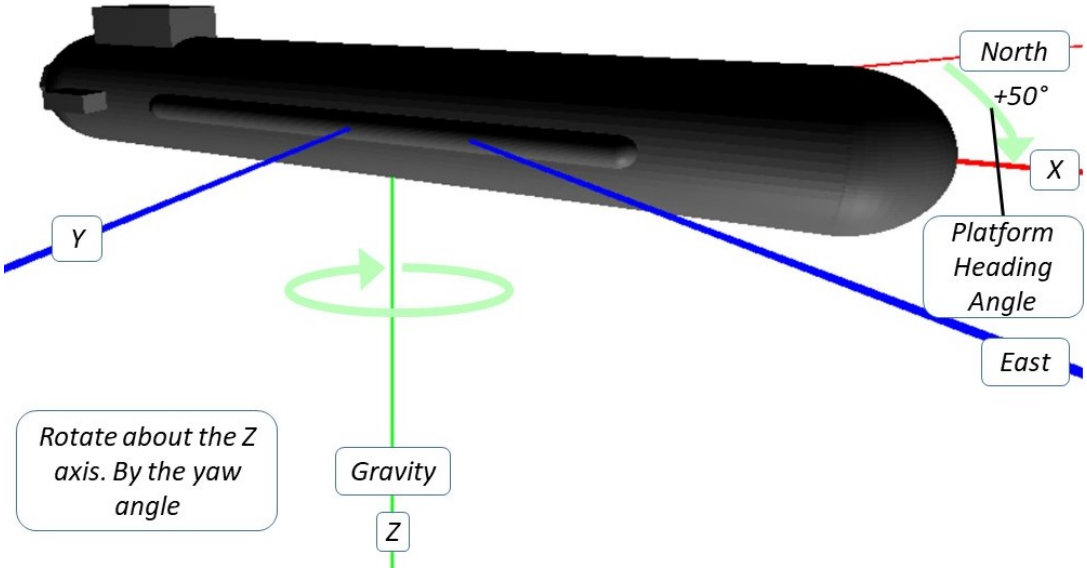

Figure 6: Rotate the vehicle by the Yaw angle

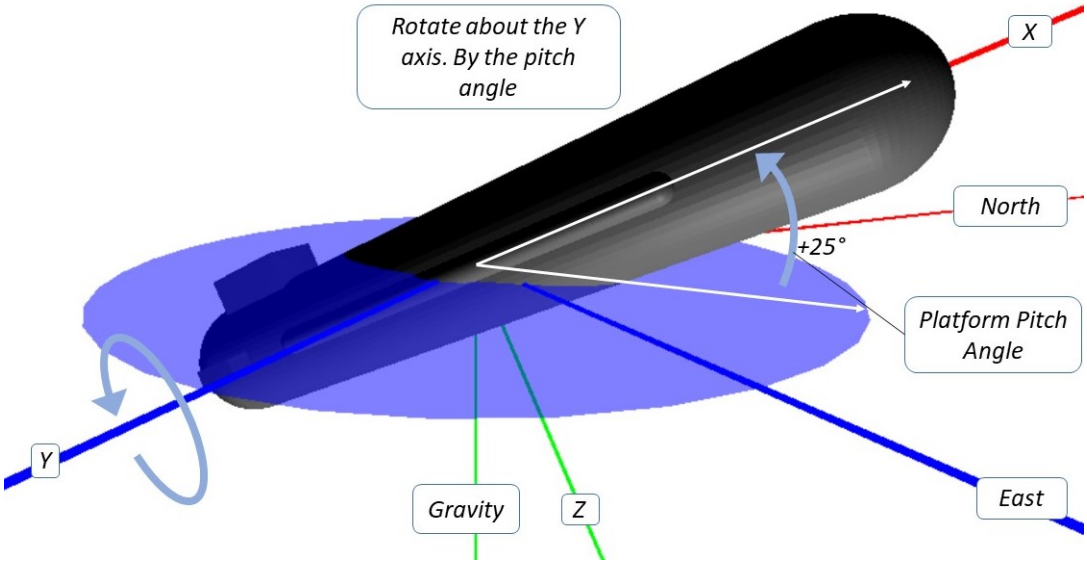

Figure 7: Rotate the vehicle by the Pitch angle

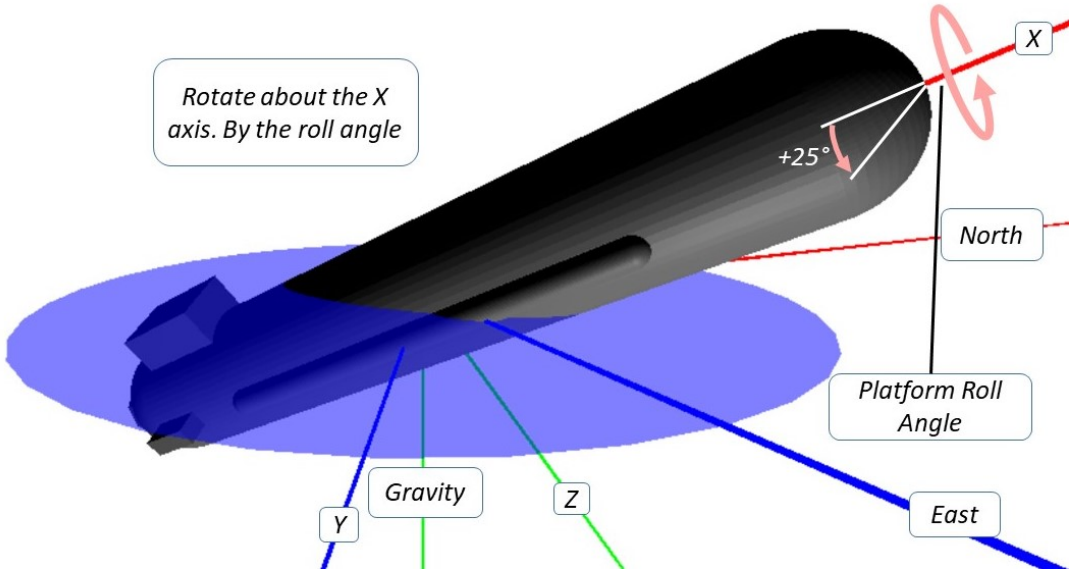

Figure 8: Rotate the vehicle by the Roll angle

4.3 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.

Definitions

- Design Waterline (DWL) - The line representing the waterline on the vehicle at designed load in summer temperature seawater.
- Centerline - The vertical plane passing fore and aft down the center of the ship.
- Aft Perpendicular (AP) - The vertical line passing through the rudder stock.
- Forward Perpendicular (FP) - The vertical line through the intersection of the forward side of the stern with the Design Waterline.
- Amidships - The midpoint between the Forward and Aft Perpendiculars.

Common practice puts the origin at the intersection of the Design Waterline, Centerline, and Amidships (Figure 9).

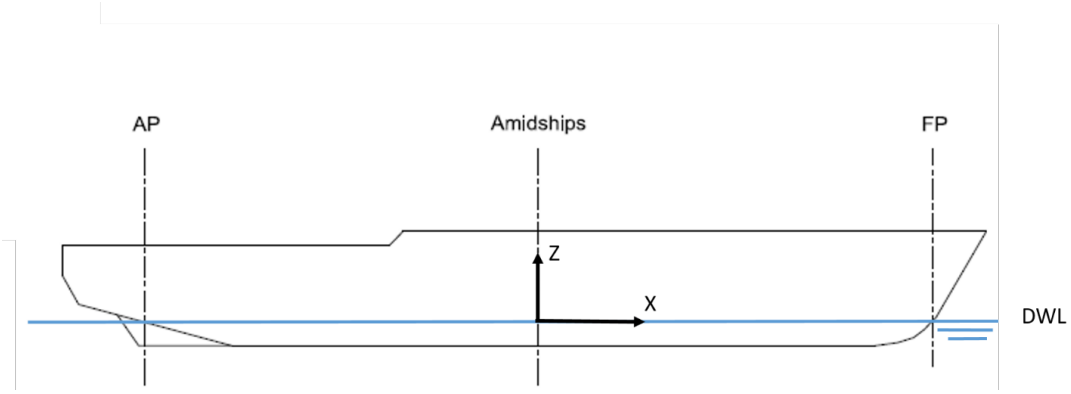

Figure 9: Origin location on a USV as example

For UUVs, common practice puts the origin as in Figure 10:

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

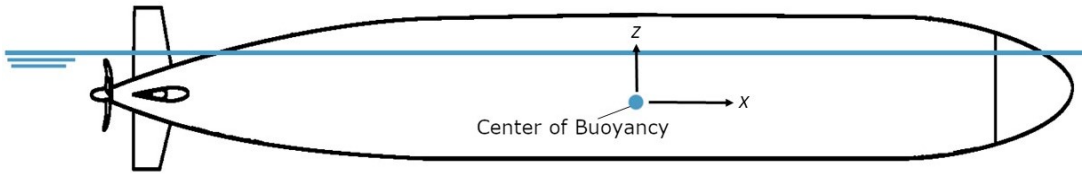

Figure 10: Origin location on a UUV as example

5 Flow Control

5.1 Command / Response

This section defines the flow of control for command/response over the DDS bus. A command/response is used to control 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 unique `sessionID` for the command when commanded. The `sessionID` is used to track the command execution as a key into other command-related messages. 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`. 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 11. 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.

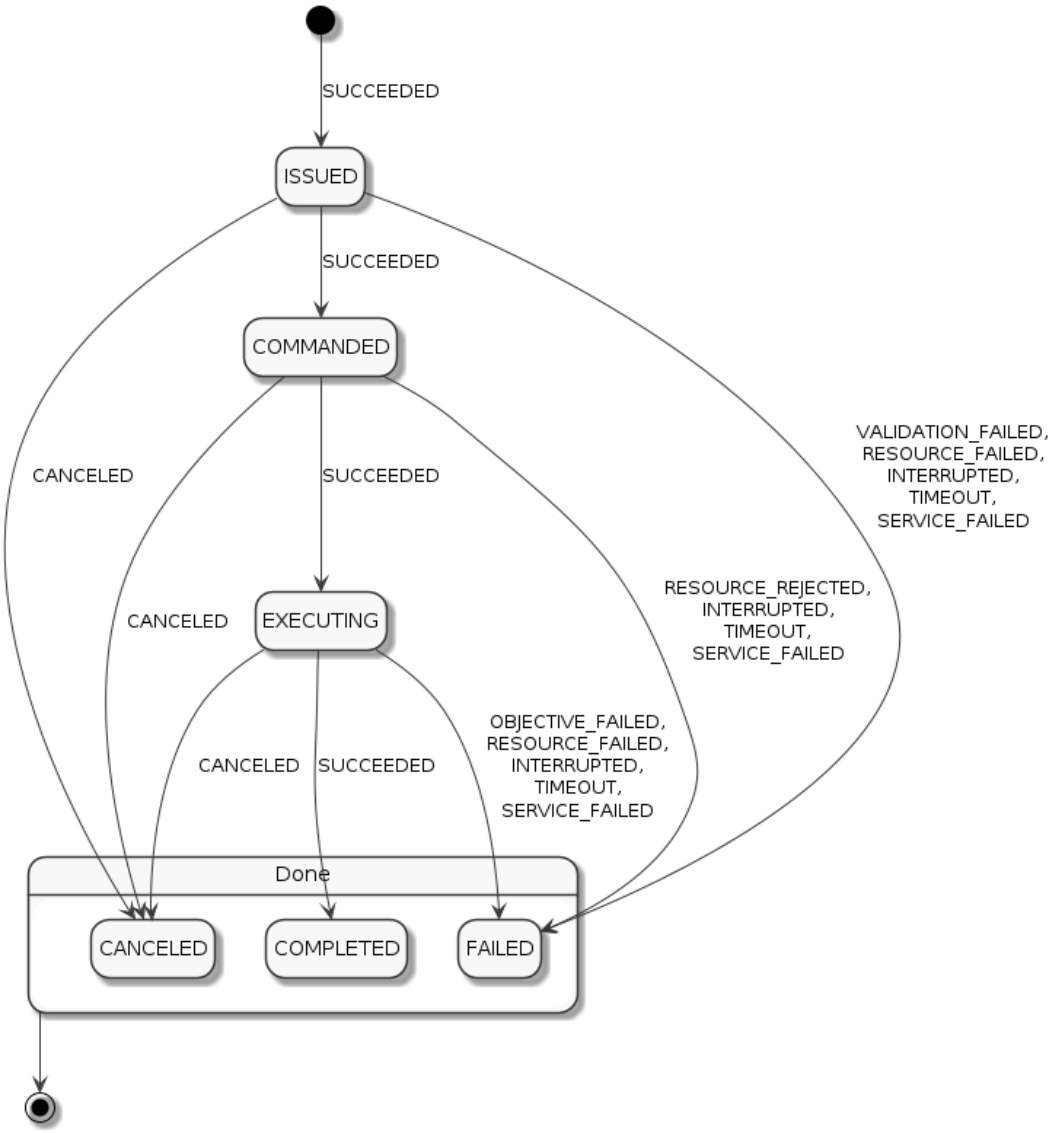

Figure 11: The state transitions of the `commandStatus` as commands are processed. Labels on the arrows represent valid `commandStatusReason` values for each transition.

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 similiar 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 12 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 my not be identical to the overall system execution duration. For example, providers may only be available to execute certain commands during specific phases of a misison or when certain hardware is in specific configurations. This Command Startup Sequence is not required to happend during a system startup phase. The only requirement is 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 anytime the `FunctionCommandType` will no longer be supported. There is no requirement the Command Shutdown Sequence only be performed during a system shutdown phase.

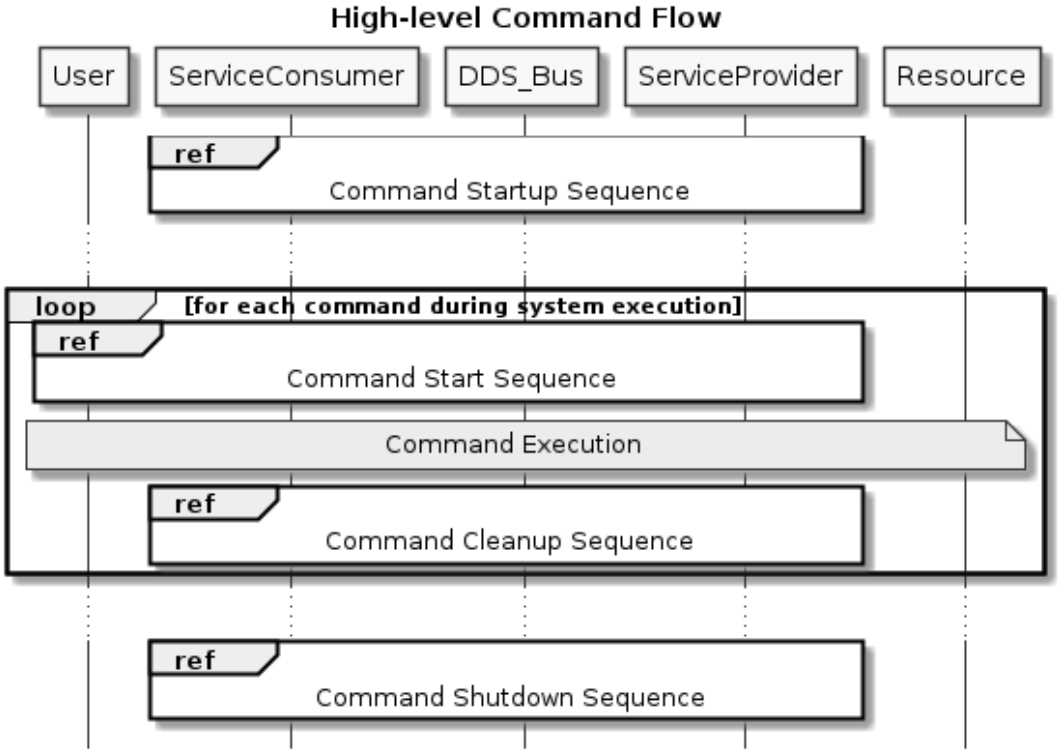

Figure 12: The sequence diagram for the high-level description of a command exeuction.

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 13.

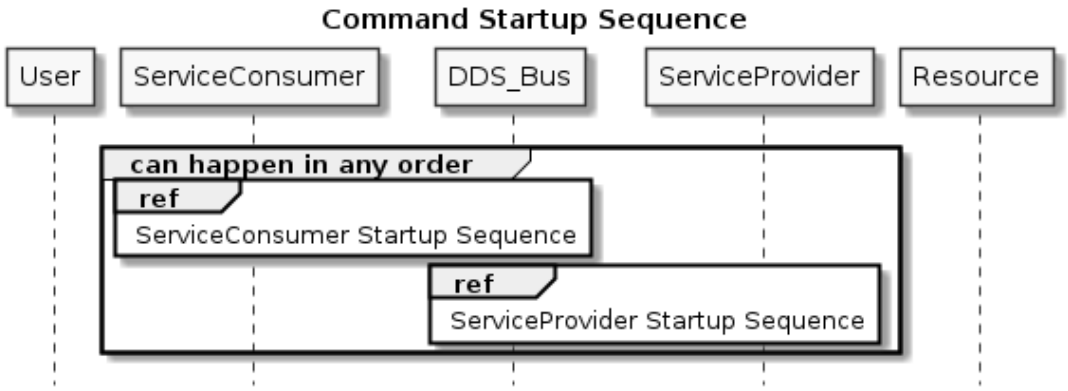

Figure 13: The sequence diagram for command startup.

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 `FunctionExecutionStatus` 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 choses 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 14.

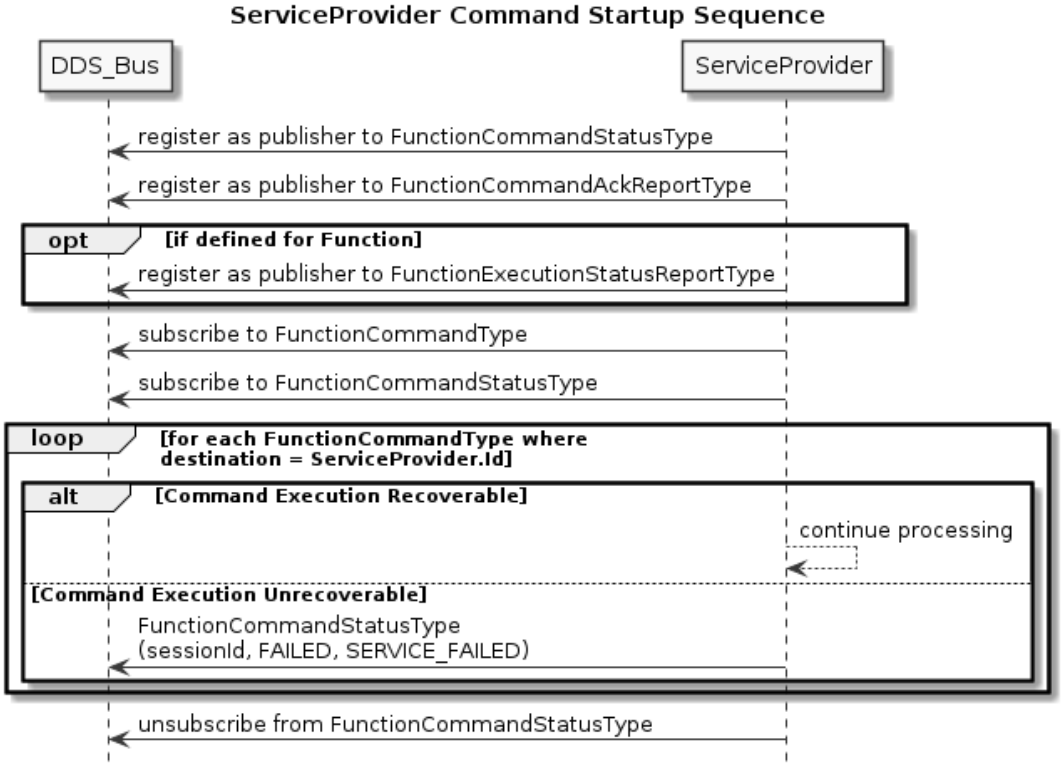

Figure 14: The sequence diagram for command startup for Service Providers.

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 choses 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 15.

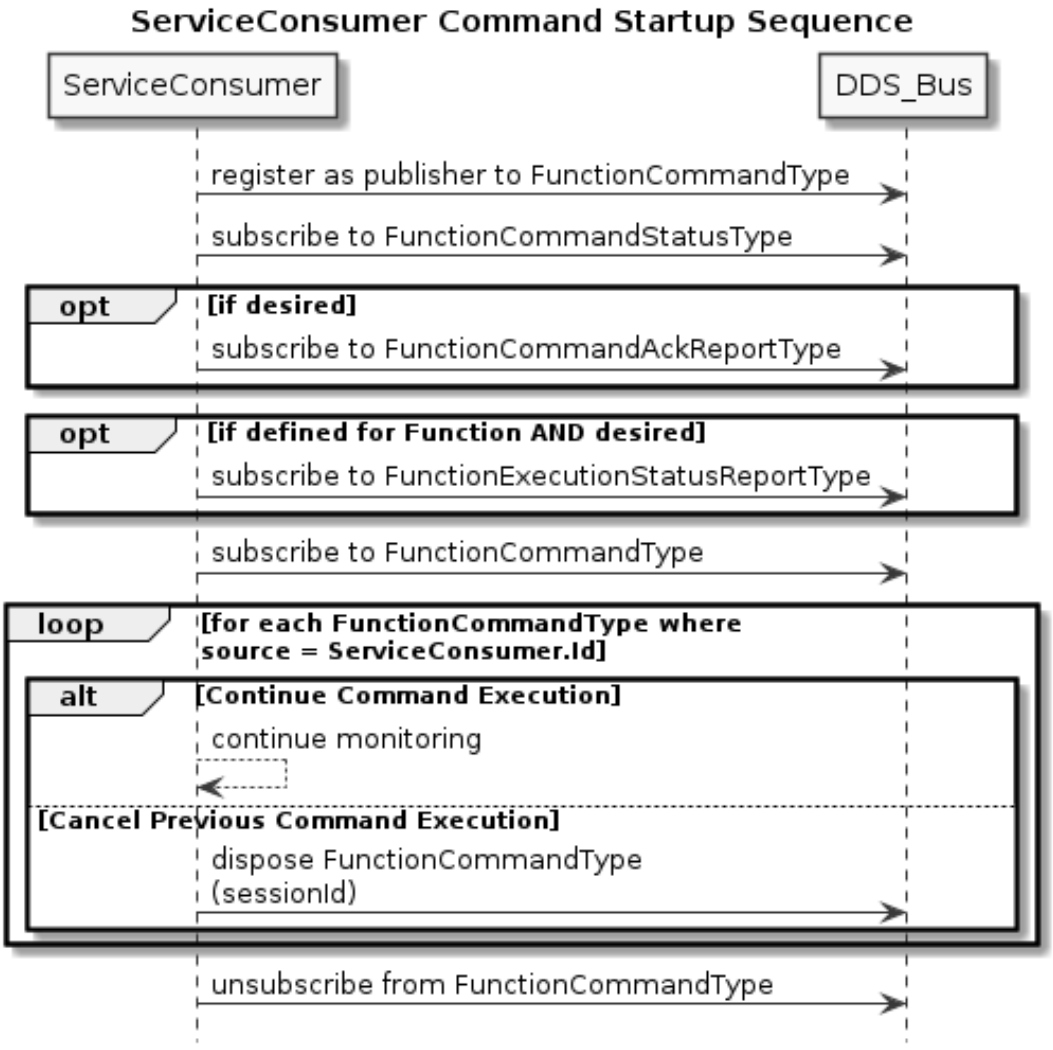

Figure 15: The sequence diagram for command startup for Service Consumers.

5.1.3 Command Execution Sequences

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

5.1.4 Command Start Sequence

The initial start sequence to execute a single 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 the same session ID as the new `FunctionCommandType` and the status of `ISSUED` and reason of `SUCCEEDED` to notify the Service Consumer it has received the new command.

The Command Start Sequence is shown in Figure 16. This pattern will be repeated each time a new command is requested. After the Command Start Sequence, the sequence can take different paths depending on the actual execution of the command. Some possible paths are detailed in the following sections, but they do not enumerate all of the possible execution paths. Other paths (e.g., an objective failing) will follow a similiar pattern to other failures; all are required to follow the state diagram shown in Figure 11 and eventually end with the Command Cleanup Sequence (as shown in Figure 22).

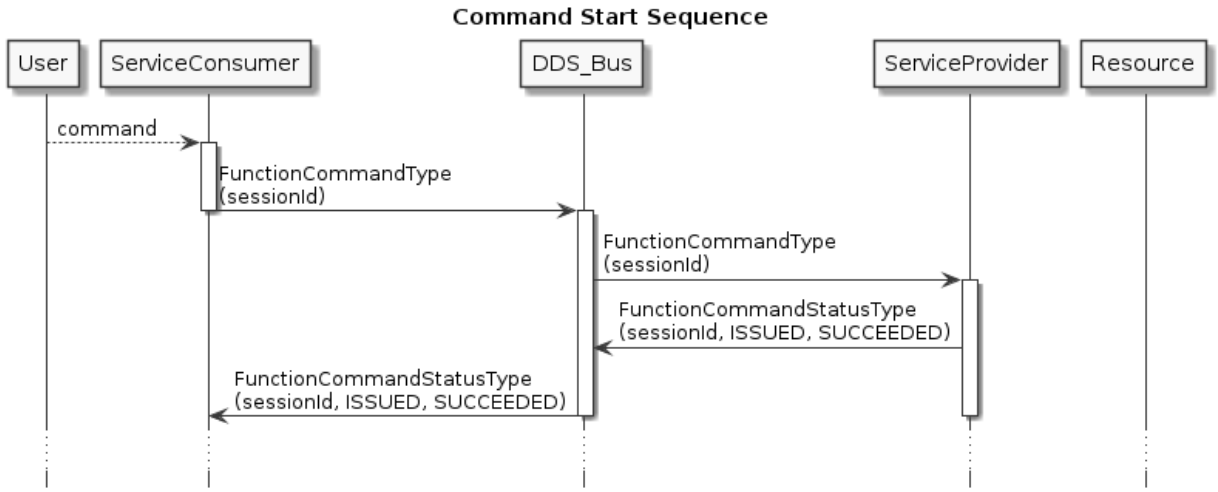

Figure 16: The sequence diagram for the start of a command execution.

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 `EXECUTED` 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 defintions of the command-specific attributes in the execution status report.

The command execution sequence is shown in Figure 17. This sequence holds until the command completes execution.

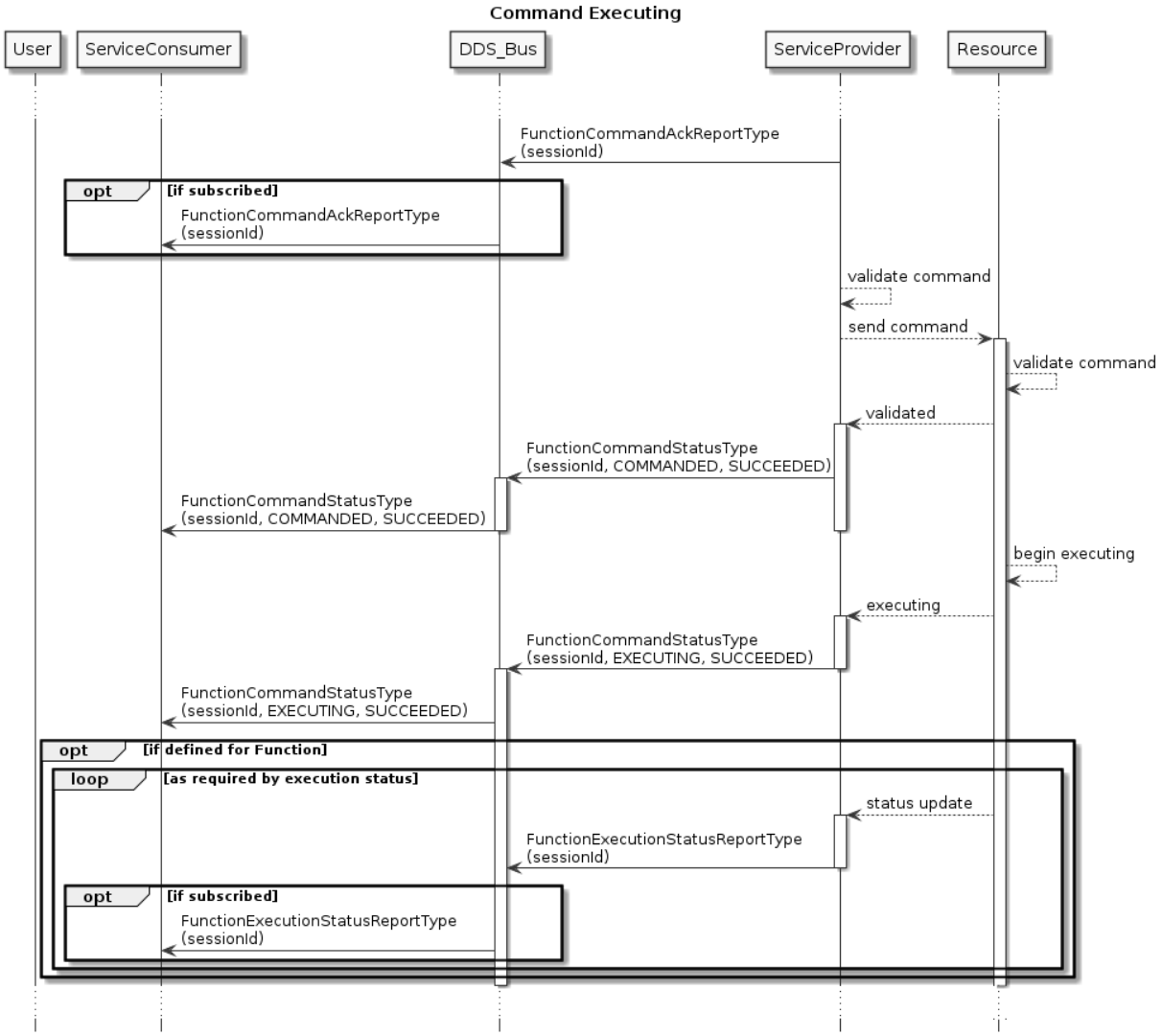

Figure 17: The beginning sequence diagram for a command execution.

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

The Command Execution Success sequence is shown in Figure 18.

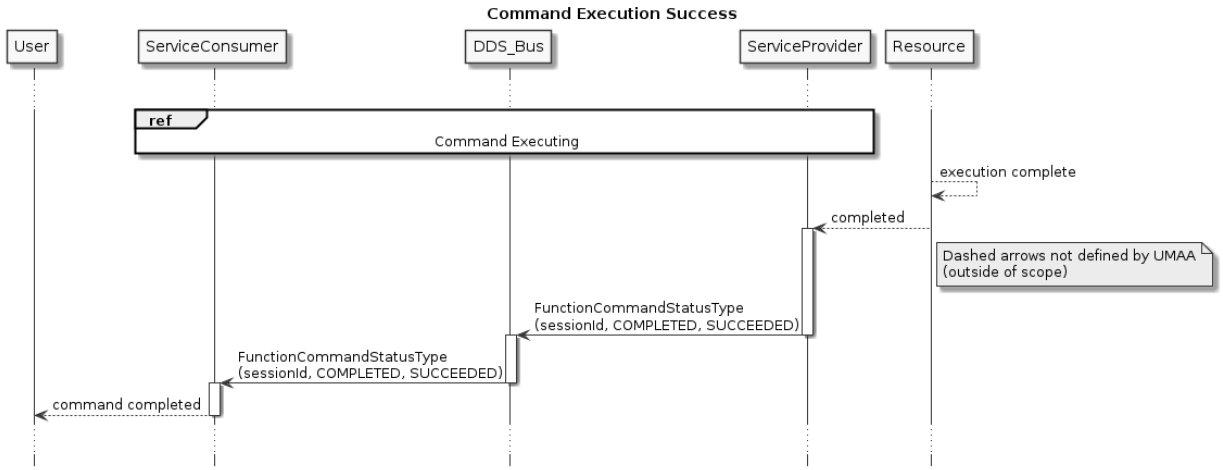

Figure 18: The sequence diagram for a command that completes successfully.

5.1.4.3 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).

The following figures provide examples of cases where a command has failed.

In the first example, the backing Resource has failed and the Service Provider in 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 19.

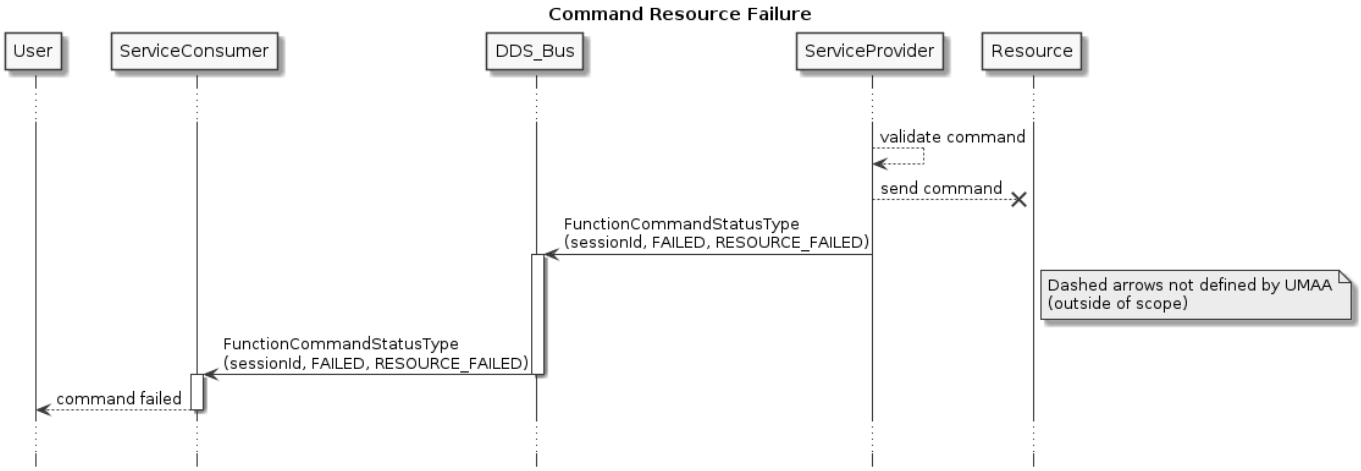

Figure 19: The sequence diagram for a command that fails due to Resource failure.

In the second example, the Resource takes too long to response, 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 20.

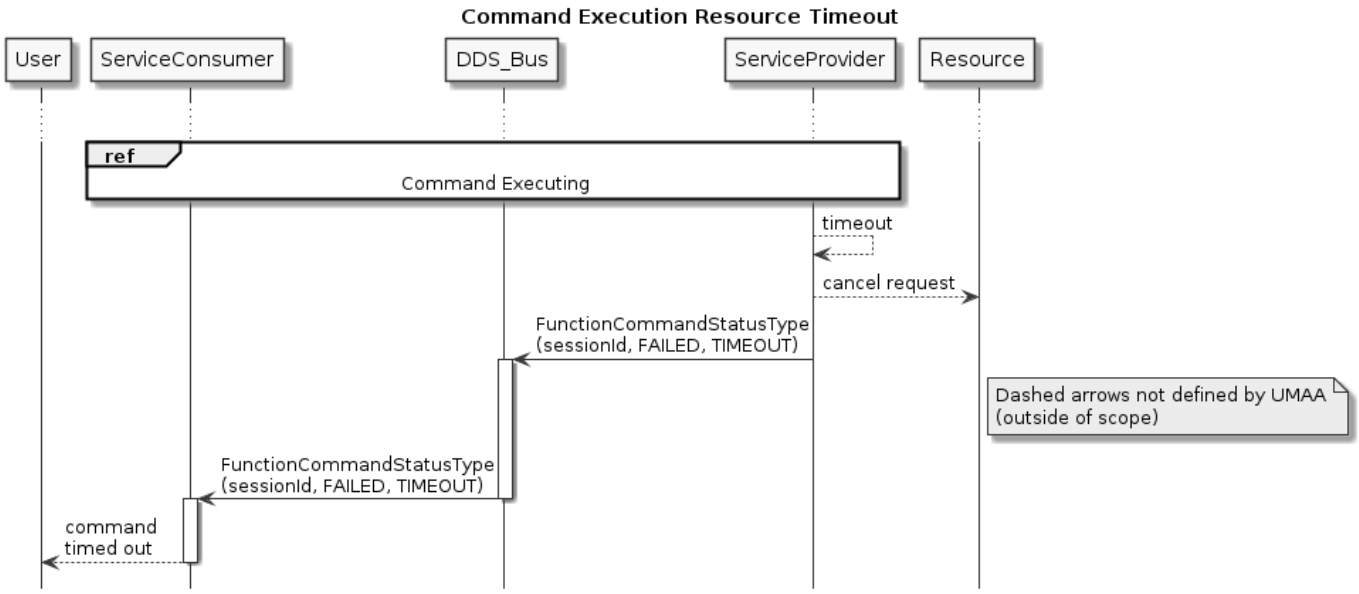

Figure 20: The sequence diagram for a command that times out before completing.

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.

5.1.4.4 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 the existing `FunctionCommandType` from the DDS bus before the execution is complete. When notified of the command disposal, 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` and then dispose the `FunctionCommandStatusType` and `FunctionCommandAckReportType` and, if defined for the Function service, the `FunctionExecutionStatusReportType` from the bus. This is shown in Figure 21. If the command cannot be canceled the Service Provider can continue to update the command status until the execution is completed, reporting `FunctionCommandStatusType` with a status of `COMPLETED` and a reason of `SUCCEEDED`, and then dispose the `FunctionCommandStatusType` and `FunctionCommandAckReportType` and, if defined for the Function service, the `FunctionExecutionStatusReportType` from the DDS bus.

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

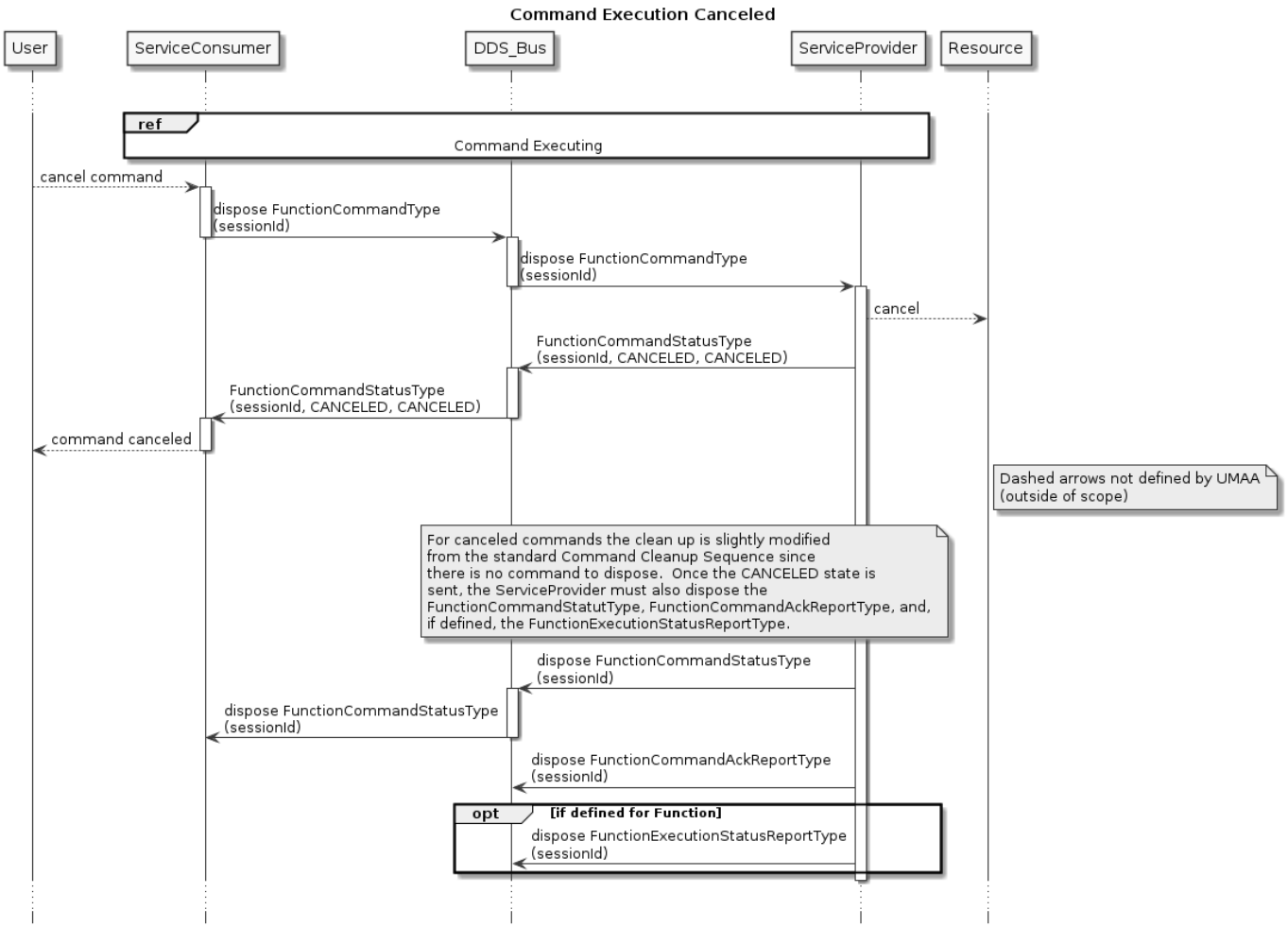

Figure 21: The sequence diagram for a command that is canceled by the Service Consumer before the Service Provider is able to complete it.

5.1.5 Command Cleanup

The Service Consumer and Service Provider are responsible for disposing corresponding data published to the DDS bus when the command is no longer active. With the exception of a canceled command, the signal that a `FunctionCommandType` can be disposed is when the `FunctionCommandStatusType` reports a terminal state (`COMPLETED` or `FAILED`)[3]. In turn, the signal that a `FunctionCommandStatusType`, `FunctionCommandAckReportType`, and if defined for the Function service, the `FunctionExecutionStatusReportType` can be disposed is when the corresponding `FunctionCommandType` has been disposed. This is shown in Figure 22.

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

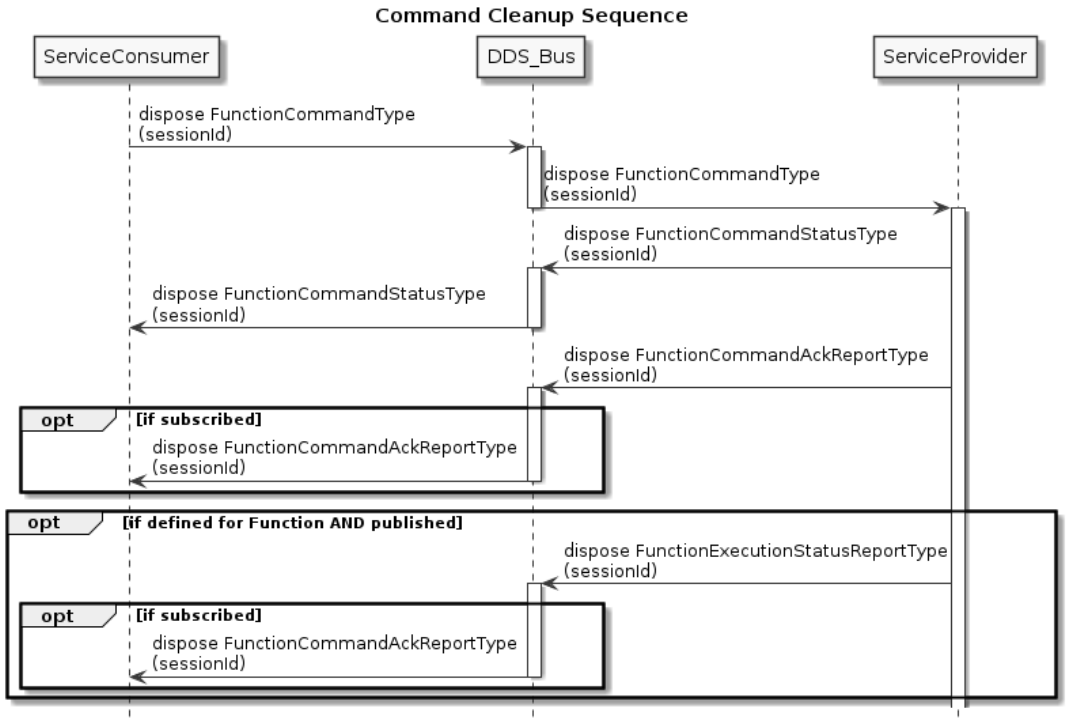

Figure 22: The sequence diagram showing cleanup of the bus when a command has been completed and the Service Consumer no longer wishes to maintain the commanded state.

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 shutdown in any order. The sequence diagram is shown in Figure 23.

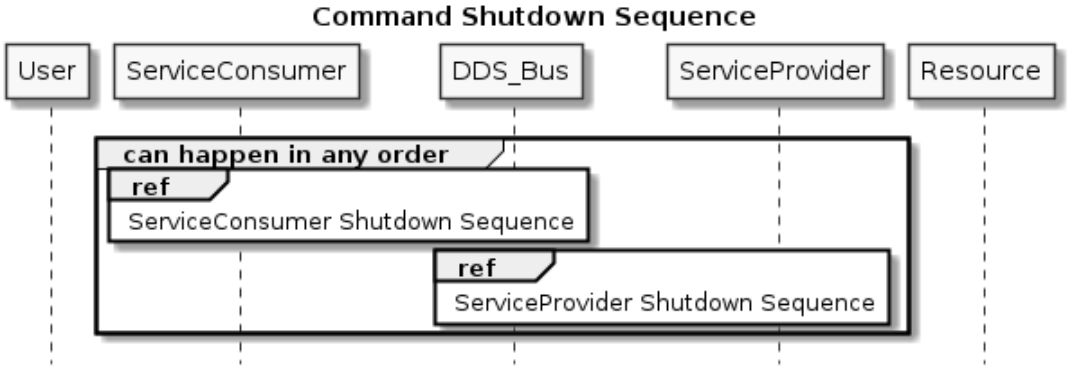

Figure 23: The sequence diagram for command shutdown.

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 24.

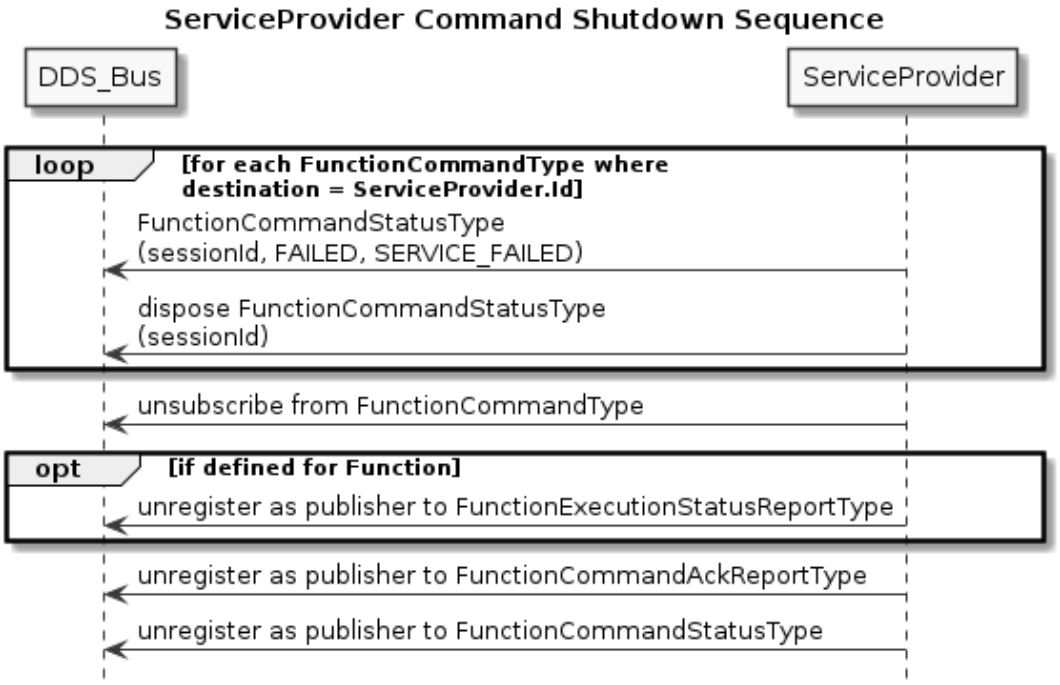

Figure 24: The sequence diagram for command shutdown for Service Providers.

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 25.

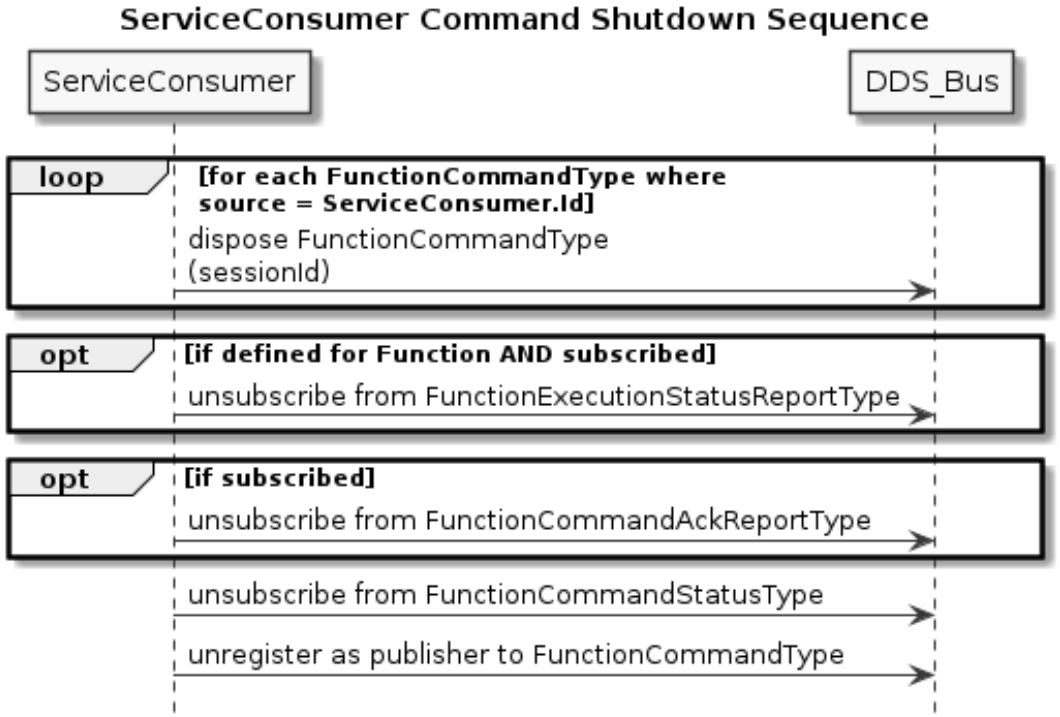

Figure 25: The sequence diagram for command shutdown for Service Consumers.

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.

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. Additionally, 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 external Resource). In all implementations, however, 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

In the case where there is no specific query data (i.e., the service is always just providing the current data to the bus) the sequence of exchanges is show in Figure 26.

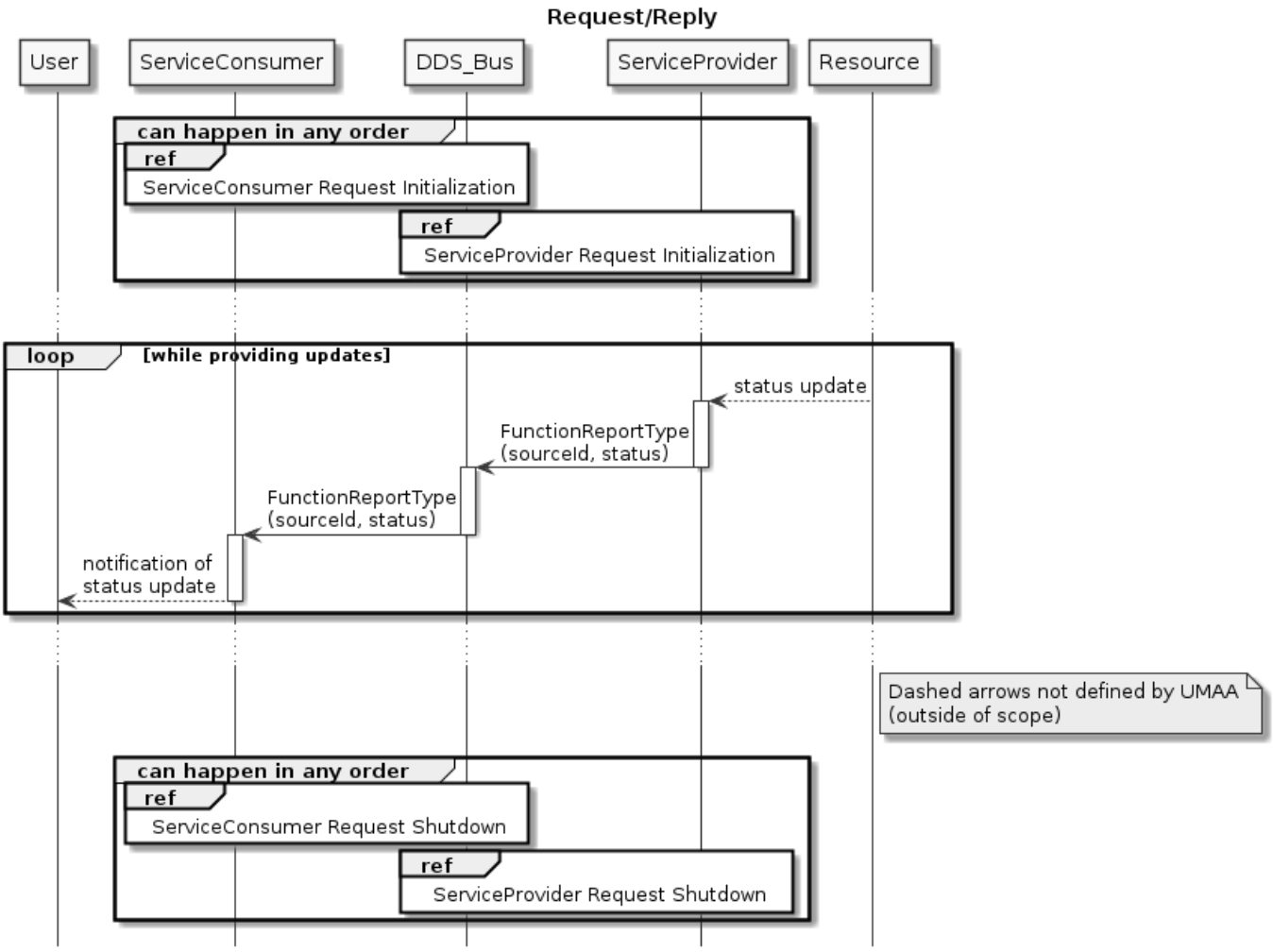

Figure 26: The sequence diagram for a request/reply for report data that does not require any specific query data.

5.2.1.1 Service Provider Startup Sequence The Service Provider registers as a publisher of `FunctionReportTypes` to be able to respond to requests. The Service Provider must also handle reports that exist on the bus from a previous instantiation, either by providing an immediate update or, if the status is unrecoverable, disposing of the old `FunctionReportType`. This is shown in Figure 27.

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

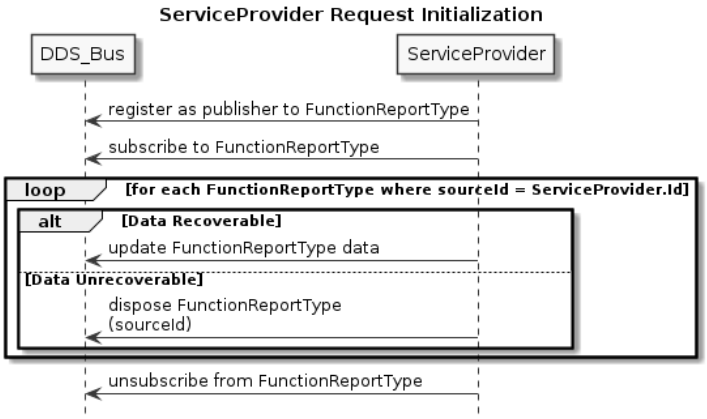

Figure 27: The sequence diagram for initialization of a Service Provider to provide FunctionReportTypes.

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 28.

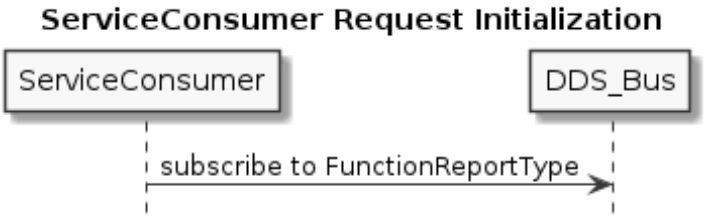

Figure 28: The sequence diagram for initialization of a Service Consumer to request FunctionReportTypes.

5.2.1.3 Service Provider Shutdown To no longer provide `FunctionReportTypes`, the Service Provider disposes the `FunctionReportType` and unregisters as a publisher of the data as shown in Figure 29.

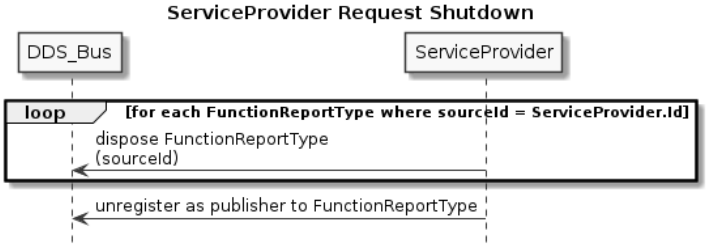

Figure 29: The sequence diagram for shutdown of a Service Provider.

5.2.1.4 Service Consumer Shutdown To no longer request `FunctionReportTypes`, the Service Consumer unsubscribes from `FunctionReportType` as shown in Figure 30.

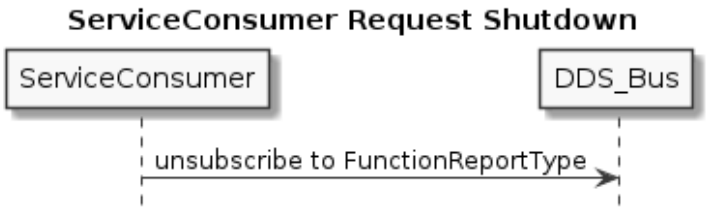

Figure 30: The sequence diagram for shutdown of a Service Consumer.

5.2.2 Request/Reply with Query Data

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

6 Mission Management (MM) Services and Interfaces

6.1 Services and Interfaces

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

Each interface topic is referenced by a UMAA service and is defined as either an input or output interface.
Attributes ending in one or more asterisk(s) denote the following:
* = Key (annotated with @key in IDL file, vendors may use different notation to indicate a key field)
†= Optional (annotated with @optional in IDL file, vendors may use different notation to indicate an optional field)

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

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

Operations without DDS Topics

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 MissionExecutionControl

The purpose of this service is to provide operations and interfaces required to manage the execution of Mission Plans.

Table 6: MissionExecutionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionExecution	reportMissionExecutionCommandStatus
queryMissionExecutionCommandAck⊕	reportMissionExecutionCommandAck
cancelMissionExecutionCommand⊕	reportMissionExecutionCancelCommandStatus⊕

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

6.1.1.1 reportMissionExecutionCommandAck

Description: This operation is used to report the current command executing the mission state.

Namespace: UMAA::MM::MissionExecutionControl

Topic: MissionExecutionCommandAckReport

Data Type: MissionExecutionCommandAckReportType

Table 7: MissionExecutionCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
state	TaskStateEnumType	A desired state of the current mission
missionID*	NumericGUID	A current mission identification

6.1.1.2 reportMissionExecutionCommandStatus

Description: This operation is used to report the current status of executing the commanded mission state.

Namespace: UMAA::MM::MissionExecutionControl

Topic: MissionExecutionCommandStatus

Data Type: MissionExecutionCommandStatusType

Table 8: MissionExecutionCommandStatusType Message Definition

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

6.1.1.3 setMissionExecution

Description: This operation is used to set the current state of a mission plan.

Namespace: UMAA::MM::MissionExecutionControl

Topic: MissionExecutionCommand

Data Type: MissionExecutionCommandType

Table 9: MissionExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
state	TaskStateEnumType	A desired state of the current mission
missionID*	NumericGUID	A current mission identification

6.1.2 MissionExecutionStatus

The purpose of this service is to provide operations and interfaces required to retrieve the current mission execution status.

Table 10: MissionExecutionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMissionExecution⊕	reportMissionExecution

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

6.1.2.1 reportMissionExecution

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

Namespace: UMAA::MM::MissionExecutionStatus

Topic: MissionExecutionReport

Data Type: MissionExecutionReportType

Table 11: MissionExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
missionStatuses	sequence<MissionStatusType>	the current status of the mission specified by the associated mission plan

6.1.3 MissionPlanAssignmentStatus

The purpose of this service is to provide operations and interfaces required to report the assignment of MissionPlans to specific resources.

Table 12: MissionPlanAssignmentStatus 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::MissionPlanAssignmentStatus

Topic: MissionPlanAssignmentReport

Data Type: MissionPlanAssignmentReportType

Table 13: MissionPlanAssignmentReportType Message Definition

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

6.1.4 MissionPlanCatalogControl

The purpose of this service is to provide operations and interfaces required to manage the Mission Plan Catalog.

Table 14: MissionPlanCatalogControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setMissionPlanCatalog	reportMissionPlanCatalogCommandStatus
queryMissionPlanCatalogCommandAck⊕	reportMissionPlanCatalogCommandAck
cancelMissionPlanCatalogCommand⊕	reportMissionPlanCatalogCancelCommandStatus⊕

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

6.1.4.1 reportMissionPlanCatalogCommandAck

Description: This operation is used to report the commanded mission plan and its metadata the Mission Plan catalog.

Namespace: UMAA::MM::MissionPlanCatalogControl

Topic: MissionPlanCatalogCommandAckReport

Data Type: MissionPlanCatalogCommandAckReportType

Table 15: MissionPlanCatalogCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
action	PlanActionEnumType	The desired action to be taken on the mission plan in the mission plan catalog.
allocationStatus	BooleanEnumType	availability of the mission plan
assignedResource	NumericGUID	the resource that the mission plan is assigned to
domain	DomainEnumType	domain that the mission plan is planned for.
format	StringShortDescription	Indicating a file based or message based type of mission plan

Attribute Name	Attribute Type	Attribute Description
name	StringShortDescription	name of the mission plan
missionID*	NumericGUID	Identifies the associated mission within the mission plan.

6.1.4.2 reportMissionPlanCatalogCommandStatus

Description: This operation is used to report the status of the Mission Plan catalog.

Namespace: UMAA::MM::MissionPlanCatalogControl

Topic: MissionPlanCatalogCommandStatus

Data Type: MissionPlanCatalogCommandStatusType

Table 16: MissionPlanCatalogCommandStatusType Message Definition

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

6.1.4.3 setMissionPlanCatalog

Description: This operation is used to manage a mission plan in the Mission Plan Catalog.

Namespace: UMAA::MM::MissionPlanCatalogControl

Topic: MissionPlanCatalogCommand

Data Type: MissionPlanCatalogCommandType

Table 17: MissionPlanCatalogCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
action	PlanActionEnumType	The desired action to be taken on the mission plan in the mission plan catalog.
allocationStatus	BooleanEnumType	availability of the mission plan
assignedResource	NumericGUID	the resource that the mission plan is assigned to
domain	DomainEnumType	domain that the mission plan is planned for.
format	StringShortDescription	Indicating a file based or message based type of mission plan
name	StringShortDescription	name of the mission plan
missionID*	NumericGUID	Identifies the associated mission within the mission plan.

6.1.5 MissionPlanStatus

The purpose of this service is to provide operations and interfaces required to provide the Mission Plan.

Table 18: MissionPlanStatus Operations

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

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

6.1.5.1 reportMissionPlan

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

Namespace: UMAA::MM::MissionPlanStatus

Topic: MissionPlanReport

Data Type: MissionPlanReportType

Table 19: MissionPlanReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
dependentMissionIDs	sequence<NumericGUID>	List of missions that must be adjudicated (e.g., completed, cancelled, etc.) before the referenced mission can begin execution.
endTime	DateTime	Absolute time for which mission must end. An optional tolerance for the end time may be defined.
endTimeTolerance†	DateTime_Tolerance	The earliest absolute time the mission can end and the latest absolute time the mission can end.
missionDescription	StringShortDescription	A brief description of the mission.
missionName	StringShortDescription	A short name for the mission.
startTime	DateTime	Absolute time for which mission must begin. An optional tolerance for the start time may be defined.
startTimeTolerance†	DateTime_Tolerance	The earliest absolute time the mission can start and the latest absolute time the mission can start.
taskPlans	sequence<TaskPlanType>	List of task plans associated with the mission.
missionID*	NumericGUID	Unique identifier for the mission.

6.1.6 MissionSummaryStatus

The purpose of this service is to provide operations and interfaces required to retrieve a summary of the current mission status.

Table 20: MissionSummaryStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryMissionSummary⊕	reportMissionSummary

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

6.1.6.1 reportMissionSummary

Description: This operation is a response to the request for the summary of current mission plans.

Namespace: UMAA::MM::MissionSummaryStatus

Topic: MissionSummaryReport

Data Type: MissionSummaryReportType

Table 21: MissionSummaryReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
missionSummaries	sequence<MissionSummaryType>	The mission summaries.

6.1.7 ObjectiveExecutionControl

The purpose of this service is to provide operations and interfaces required to manage the execution of Objectives for a particular Mission and Task.

Table 22: 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 ⊕.

6.1.7.1 reportObjectiveExecutionCommandAck

Description: This operation is used to report the current commanded objective state.

Namespace: UMAA::MM::ObjectiveExecutionControl

Topic: ObjectiveExecutionCommandAckReport

Data Type: ObjectiveExecutionCommandAckReportType

Table 23: ObjectiveExecutionCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
state	TaskStateEnumType	A desired state of the current objective
missionID*	NumericGUID	A current mission identification
objectiveID*	NumericGUID	A current objective identification within the current task
taskID*	NumericGUID	A current task identification within the current mission

6.1.7.2 reportObjectiveExecutionCommandStatus

Description: This operation is used to report the current status of executing the commanded objective state.

Namespace: UMAA::MM::ObjectiveExecutionControl

Topic: ObjectiveExecutionCommandStatus

Data Type: ObjectiveExecutionCommandStatusType

Table 24: ObjectiveExecutionCommandStatusType Message Definition

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

6.1.7.3 setObjectiveExecution

Description: This operation is used to set the current state of an objective within a mission plan.

Namespace: UMAA::MM::ObjectiveExecutionControl

Topic: ObjectiveExecutionCommand

Data Type: ObjectiveExecutionCommandType

Table 25: ObjectiveExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
state	TaskStateEnumType	A desired state of the current objective
missionID*	NumericGUID	A current mission identification
objectiveID*	NumericGUID	A current objective identification within the current task

Attribute Name	Attribute Type	Attribute Description
taskID*	NumericGUID	A current task identification within the current mission

6.1.8 ObjectiveExecutionStatus

The purpose of this service is to provide operations and interfaces required to retrieve the execution status of each Objective for a particular Mission and Task.

Table 26: ObjectiveExecutionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryObjectiveExecutionStatus⊕	reportObjectiveExecutionStatus

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

6.1.8.1 reportObjectiveExecutionStatus

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

Namespace: UMAA::MM::ObjectiveExecutionStatus

Topic: ObjectiveExecutionReport

Data Type: ObjectiveExecutionReportType

Table 27: ObjectiveExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
objectiveStatuses	sequence<ObjectiveStatusType>	the current status of the objective specified by the associated objective.

6.1.9 ObjectivePlanAssignmentStatus

The purpose of this service is to provide operations and interfaces required to report the assignment of Objectives to specific resources.

Table 28: ObjectivePlanAssignmentStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryObjectivePlanAssignment⊕	reportObjectivePlanAssignment

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

6.1.9.1 reportObjectivePlanAssignment

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

Namespace: UMAA::MM::ObjectivePlanAssignmentStatus

Topic: ObjectivePlanAssignmentReport

Data Type: ObjectivePlanAssignmentReportType

Table 29: ObjectivePlanAssignmentReportType Message Definition

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

6.1.10 TaskExecutionControl

The purpose of this service is to provide operations and interfaces required to manage the execution of tasks for a particular Mission.

Table 30: TaskExecutionControl Operations

Service Requests (Inputs)	Service Responses (Outputs)
setTaskExecution	reportTaskExecutionCommandStatus
queryTaskExecutionCommandAck⊕	reportTaskExecutionCommandAck
cancelTaskExecutionCommand⊕	reportTaskExecutionCancelCommandStatus⊕

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

6.1.10.1 reportTaskExecutionCommandAck

Description: This operation is used to report the current commanded task state.

Namespace: UMAA::MM::TaskExecutionControl

Topic: TaskExecutionCommandAckReport

Data Type: TaskExecutionCommandAckReportType

Table 31: TaskExecutionCommandAckReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommandStatusBase		
missionID*	NumericGUID	A current mission identification
taskID*	NumericGUID	A current task identification within the current mission

6.1.10.2 reportTaskExecutionCommandStatus

Description: This operation is used to report the current status of executing the commanded task state.

Namespace: UMAA::MM::TaskExecutionControl

Topic: TaskExecutionCommandStatus

Data Type: TaskExecutionCommandStatusType

Table 32: TaskExecutionCommandStatusType Message Definition

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

6.1.10.3 setTaskExecution

Description: This operation is used to set the current state of a task within a mission plan.

Namespace: UMAA::MM::TaskExecutionControl

Topic: TaskExecutionCommand

Data Type: TaskExecutionCommandType

Table 33: TaskExecutionCommandType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAACommand		
missionID*	NumericGUID	A current mission identification
taskID*	NumericGUID	A current task identification within the current mission

6.1.11 TaskExecutionStatus

The purpose of this service is to provide operations and interfaces required to retrieve the execution status of each Task for a particular Mission.

Table 34: TaskExecutionStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryTaskExecution⊕	reportTaskExecution

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

6.1.11.1 reportTaskExecution

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

Namespace: UMAA::MM::TaskExecutionStatus

Topic: TaskExecutionReport

Data Type: TaskExecutionReportType

Table 35: TaskExecutionReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
taskStatuses	sequence<TaskStatusType>	the current status of a task specified by the associated task.

6.1.12 TaskPlanAssignmentStatus

The purpose of this service is to provide operations and interfaces required to report the assignment of Tasks to specific resources.

Table 36: TaskPlanAssignmentStatus Operations

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

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

6.1.12.1 reportTaskPlanAssignment

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

Namespace: UMAA::MM::TaskPlanAssignmentStatus

Topic: TaskPlanAssignmentReport

Data Type: TaskPlanAssignmentReportType

Table 37: TaskPlanAssignmentReportType Message Definition

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

6.1.13 WaterspacePlanAssignmentStatus

The purpose of this service is to provide operations and interfaces required to report the assignment of Missions, Tasks, Objectives to a specific WaterspacePlan.

Table 38: WaterspacePlanAssignmentStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryWaterspacePlanAssignment⊕	reportWaterspacePlanAssignment

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

6.1.13.1 reportWaterspacePlanAssignment

Description: This operation is a response to the request for the waterspace operational areas. Each operational area consists of one or more zones that are assigned to an objective, task, or mission. Waterspace zones at the objective and task level cannot conflict with the waterspace plans at the task and mission level, respectively.

Namespace: UMAA::MM::WaterspacePlanAssignmentStatus

Topic: WaterspacePlanAssignmentReport

Data Type: WaterspacePlanAssignmentReportType

Table 39: WaterspacePlanAssignmentReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
missionID*	NumericGUID	Defines the mission that is associated with the operational area
objectiveID*	NumericGUID	Defines the objective that is associated with the operational area
taskID*	NumericGUID	Defines the task that is associated with the operational area
waterspacePlanID*	NumericGUID	Defines an identifier associated with each define operational area

6.1.14 WaterspacePlanStatus

Intended to define valid keep in and keep out zone(s) for a vehicle. Zones are defined by a polygon that defines the zone, the zone type (keep in or keep out), and a schedule for the when the zone is valid. The keep out zones should overlap with a keep in zone and define subareas within the larger keep in zone that are excluded from the operation area (e.g., an island within an operation area)

Table 40: WaterspacePlanStatus Operations

Service Requests (Inputs)	Service Responses (Outputs)
queryWaterspacePlan⊕	reportWaterspacePlan

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

6.1.14.1 reportWaterspacePlan

Description: This operation is a response to the request for the waterspace zones.

Namespace: UMAA::MM::WaterspacePlanStatus

Topic: WaterspacePlanReport

Data Type: WaterspacePlanReportType

Table 41: WaterspacePlanReportType Message Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::UMAAStatus		
waterspacePlanName	StringShortDescription	The name of the current waterspace zones.
zones	sequence<PlanningZoneType>	The zone described by the waterspace plan.
waterspacePlanID*	NumericGUID	The unique identifier of the waterspace plan.

6.2 Common Data Types

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

6.2.1 UCSMDEInterfaceSet

Namespace: UMAA::UCSMDEInterfaceSet

Description: Defines the common UCSMDE Interface Set Message Fields.

Table 42: UCSMDEInterfaceSet Structure Definition

Attribute Name	Attribute Type	Attribute Description
timeStamp	DateTime	The time at which the data was derived.

6.2.2 UMAACommand

Namespace: UMAA::UMAACommand

Description: Defines the common UMAA Command Message Fields.

Table 43: 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 44: 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 45: 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 46: 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 47: 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 Altitude_AGL

Namespace: UMAA::Common::Measurement::Altitude_AGL

Description: Altitude_AGL specifies the entity's height above terrain, as reported by a radar system.

Table 48: Altitude_AGL Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	RadarHeight	Specifies the entity's height above terrain, as reported by a radar system.

6.2.8 Altitude_HAE

Namespace: UMAA::Common::Measurement::Altitude_HAE

Description: Altitude_HAE specifies the entity's height above the reference ellipsoid.

Table 49: Altitude_HAE Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	EllipsoidalHeight	Specifies the entityâ€™s height above the reference ellipsoid.

6.2.9 Altitude_MSL

Namespace: UMAA::Common::Measurement::Altitude_MSL

Description: Altitude_MSL specifies the entity's height above the geoid.

Table 50: Altitude_MSL Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude	MSLHeight	Specifies the entity's height above the geoid.

6.2.10 ChargingObjectiveType

Namespace: UMAA::MM::BaseType::ChargingObjectiveType

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

Table 51: ChargingObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
chargeLevel	Power_Percent	Specifies the charge level that must be obtained during charging

6.2.11 CommsLinkObjectiveType

Namespace: UMAA::MM::BaseType::CommsLinkObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required for establishing/maintaining a communications link.

Table 52: CommsLinkObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		

6.2.12 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 53: ContingencyObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
altitude	Altitude_MSL	Specifies the distance along the vehicle path to the waypoint
altitudeAGL	Altitude_AGL	altitudeAGL that specifies the distance above ground level
altitudeASF	Distance_ASF	Specifies the distance above sea level
behavior	ContingencyBehaviorEnumType	contingencyLevel defines the system contingency level for the mission
depth	Distance_BSL	Specifies the distance of the waypoint below sea level
DTEDAltitude	Distance	Specifies DTED altitude at the origin
mode	StringShortDescription	specifies the navigation mode that the unmanned platform operating in
position	Position2D	Specifies the location for the vehicle to be at in case of emergency or lost communication
radius	Distance	Specifies the radius the unmanned platform 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 unmanned platform
vehicleRunTime	Duration_Hours	Specifies the duration that the unmanned platform can go

6.2.13 DateTime_Tolerance

Namespace: UMAA::Common::Measurement::DateTime_Tolerance

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

attribute.

Table 54: DateTime_Tolerance Structure Definition

Attribute Name	Attribute Type	Attribute Description
lowerLimit	DateTime	specifies the minimum value of the time point.
stepSize	DateTime	specifies the difference between allowable time values between lowerLimit and upperLimit.
upperLimit	DateTime	specifies the maximum value of time point.

6.2.14 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 55: DeploymentObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
altitude	Altitude_MSL	Specifies the distance along the vehicle path to the waypoint.
altitudeAGL	Altitude_AGL	Specifies the distance above ground level
altitudeASF	Distance_ASF	Specifies the distance above sea level
heading	Heading_TrueNorth_Angle	Specifies the heading to be maintained during deployment
position	Position2D	Specifies the location for deploying the vehicle
releaseDepth	Distance_BSL	Specifies the distance below sea level for releasing the vehicle
speed	GroundSpeed	Specifies the speed to be maintained during deployment

6.2.15 GeodeticLatitude

Namespace: UMAA::Common::Measurement::GeodeticLatitude

Description: GeodeticLatitude specifies the angle between the normal and the equatorial plane of the ellipsoid. The Latitude specifies the north-south position of a point.

Table 56: GeodeticLatitude Structure Definition

Attribute Name	Attribute Type	Attribute Description
latitude	GeodeticLatitude	GeodeticLatitude specifies the angle between the normal and the equatorial plane of the ellipsoid. The Latitude specifies the north-south position of a point.

6.2.16 GeodeticLongitude

Namespace: UMAA::Common::Measurement::GeodeticLongitude

Description: GeodeticLongitude specifies the angular measurement of a location east or west of the prime meridian of the reference ellipsoid.

Table 57: GeodeticLongitude Structure Definition

Attribute Name	Attribute Type	Attribute Description
longitude	GeodeticLongitude	GeodeticLongitude specifies the angular measurement of a location east or west of the prime meridian of the reference ellipsoid.

6.2.17 LoiterObjectiveType

Namespace: UMAA::MM::BaseType::LoiterObjectiveType

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

Table 58: LoiterObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
altitude†	Altitude_MSL	Defines the altitude for loitering.
altitudeAGL†	Altitude_AGL	Defines the altitude for loitering in an above ground level reference frame.
altitudeASF†	Distance_ASF	Defines the altitude for loitering in an above sea floor reference frame.
autoHeading†	BooleanEnumType	Describes whether there is an auto heading set.
bearing†	Heading_TrueNorth_Angle	Defines the bearing that the vehicle must maintain for loitering
depth†	Distance_BSL	Defines the depth for loitering in a below sea level reference frame.
direction†	WaterTurnDirectionEnumType	Defines the direction for loitering.
duration	Duration_Hours	Defines the duration for loitering.
heading†	Heading_TrueNorth_Angle	Defines the heading that the vehicle must maintain for loitering.
hoverKind†	HoverKindEnumType	Defines the hover pattern for loitering.
length†	Distance	Describes the length of the loiter pattern.
position	Position3D_WGS84	Defines the reference position for loitering.
speed	Speed	Defines the speed used to maneuver the vehicle to the reference position.
type	LoiterKindEnumType	Defines the type of loitering action.
width	Distance	Describes the width of the loiter pattern.

6.2.18 MissionStatusType

Namespace: UMAA::MM::MissionExecutionStatus::MissionStatusType

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

Table 59: MissionStatusType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime	DateTime	Specifies the end time for the mission associated with missionID
feedback	StringShortDescription	reason for why the vehicle system rejects the Mission/Task/Status plan.
startTime	DateTime	Specifies the start time for the mission associated with missionID
state	TaskStateEnumType	Specifies the current state of the mission specified by the associated mission plan.
missionID*	NumericGUID	An identification of the mission

6.2.19 MissionSummaryType

Namespace: UMAA::MM::BaseType::MissionSummaryType

Description: This structure is used to summarize the mission.

Table 60: MissionSummaryType Structure Definition

Attribute Name	Attribute Type	Attribute Description
missionDescription	StringShortDescription	The description of the mission.
missionName	StringShortDescription	The name of the mission.
startTime	DateTime	The start time of the mission.
state	TaskStateEnumType	The current state of the mission.
missionID*	NumericGUID	The unique identifier of the mission.

6.2.20 ObjectiveStatusType

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

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

Table 61: ObjectiveStatusType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime	DateTime	Specifies the end time for the objective associated with missionID,taskID,objectiveID.
startTime	DateTime	Specifies the start time for the objective associated with missionID,taskID,objectiveID.
state	TaskStateEnumType	Specifies the current state of the objective specified by the associated objective.

Attribute Name	Attribute Type	Attribute Description
missionID*	NumericGUID	An identification of the mission
objectiveID*	NumericGUID	objectiveID identifies the associated objective within the mission plan
taskID*	NumericGUID	An identification of the associated task within the mission plan

6.2.21 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 62: ObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
approvalRequired	BooleanEnumType	An indication whether approval is required for the specified objective within a mission
childObjectiveIDs	sequence<NumericGUID>	If the objective is decomposed into lower level objectives, specifies one or more unique identifiers of the children objectives that decompose the objective.
dependentObjectiveIDs	sequence<NumericGUID>	List of objectives that must be adjudicated (e.g., completed, cancelled, etc.) before the referenced objective can begin execution.
endTime	DateTime	Absolute time for which task must end. An optional tolerance for the end time may be defined.
endTimeTolerance†	DateTime_Tolerance	The earliest absolute time the task can end and the latest absolute time the mission can end.
name	StringShortDescription	A short name for the objective.
preferredResourceIDs	sequence<NumericGUID>	If defined, specifies the unique identifier of the preferred resource(s) to execute objective.
priority	Order	Specifies the execution priority for the objective
startTime	DateTime	Absolute time for which task must begin. An optional tolerance for the start time may be defined.
startTimeTolerance†	DateTime_Tolerance	The earliest absolute time the task can start and the latest absolute time the mission can start.
objectiveID*	NumericGUID	Unique identifier for the objective within a mission.
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.

Table 63: ObjectiveType Children

Type Name	Type Description
ChargingObjectiveType	This structure is used to described a clearly defined goal specifying the action(s) required for vehicle charging.
CommsLinkObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for establishing/maintaining a communications link.
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.
LoiterObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for loitering.
PassiveLoiterObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for passive loitering.
ProductDisseminateFileObjectiveType	This structure is used to define goal specifying the action(s) required for sensor data dissemination.
ProductExploitationObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required to exploit a given payload sensor product or set of products.
RecoveryObjectiveType	This structure is used to describe a clearly defined goal specifying the action(s) required for vehicle recovery.
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.

6.2.22 Orientation3D

Namespace: UMAA::Common::Measurement::Orientation3D

Description: Orientation3D specifies the orientation of the platform in the order yaw, pitch, roll. The angles are given in a locally level, North-East-Down coordinate system centered on the platform.

Table 64: Orientation3D Structure Definition

Attribute Name	Attribute Type	Attribute Description
pitchY	Pitch_HalfAngle	pitchY 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.
rollX	Roll_Angle	rollX specifies the platform's rotation about the longitudinal axis (e.g. the axis through the body of an aircraft from tail to nose) in a locally level, North-East-Down coordinate system centered on the platform. 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.

Attribute Name	Attribute Type	Attribute Description
yawZ	Yaw_PosAngle	yawZ specifies the platform's rotation about the vertical axis (e.g. the axis from top to bottom through an aircraft) in a locally level, North-East-Down coordinate system centered on the platform. By this definition, yaw is zero when the platform is oriented toward true North and is equivalent to true North referenced heading. The measurement is stated in radians between -pi and pi.

6.2.23 PassiveLoiterObjectiveType

Namespace: UMAA::MM::BaseType::PassiveLoiterObjectiveType

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

Table 65: PassiveLoiterObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
captureRadius	Distance	Defines the capture radius that specifies the minimum distance from the reference position the vehicle must achieve while maneuvering to it.
depth†	Distance_BSL	Defines the depth for loitering
driftRadius†	Distance	Defines the drift radius that specifies the maximum distance from the reference position the vehicle is allowed to drift.
duration†	Duration_Seconds	Defines the duration for loitering
heading†	RelativeAngle	Defines the heading that the vehicle must maintain for loitering
headingReference†	HeadingReferenceEnumType	Defines the reference frame for the required vehicle heading
position	Position2DTime	Defines the reference position for loitering
speed	GroundSpeed	Defines the speed used to maneuver the vehicle to the reference position

6.2.24 PlanningZoneType

Namespace: UMAA::MM::BaseType::PlanningZoneType

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

Table 66: PlanningZoneType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime†	DateTime	Defines the end time for the zone. If not defined means active indefinitely.
startTime†	DateTime	Defines the start time for the zone. If not defined means immediately active.

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

6.2.25 Polygon_Volume

Namespace: UMAA::Common::Measurement::Polygon_Volume

Description: Realizes VolumeType: an entity that describes a quantity of three-dimensional space enclosed by some closed boundary.

Table 67: Polygon_Volume Structure Definition

Attribute Name	Attribute Type	Attribute Description
ceiling	Altitude_MSL	Describes the plane relative to the mean sea level that intersects the highest point or plane of the polygon.
floor	Altitude_MSL	Describes the plane relative to the mean sea level that intersects the lowest point or plane of the polygon.
lineKind	LineSegmentEnumType	Describes the type of line used to represent the polygon.
referencePoints	sequence<Position2D>	Describes a reference point for the polygon.

6.2.26 Position2D

Namespace: UMAA::Common::Measurement::Position2D

Description: Position2D specifies a location on the surface of the Earth.

Table 68: Position2D Structure Definition

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

6.2.27 Position2DTime

Namespace: UMAA::Common::Measurement::Position2DTime

Description: Position2DTime specifies a location on the surface of the Earth at a given point in time.

Table 69: Position2DTime Structure Definition

Attribute Name	Attribute Type	Attribute Description
geodeticPosition	Position2D	geodeticPosition specifies a location on the surface of the Earth.
timeAtPosition†	DateTime	timeAtPosition specifies the date and time when this position is considered valid or an associated measurement was made.

6.2.28 Position3D_PlatformXYZ

Namespace: UMAA::Common::Measurement::Position3D_PlatformXYZ

Description: Position3D_PlatformXYZ specifies a location on a Cartesian coordinate system relative to the origin of the platform.

Table 70: Position3D_PlatformXYZ Structure Definition

Attribute Name	Attribute Type	Attribute Description
xA	Forward	xA specifies the X-axis position which is in the forward (toward the nose) direction.
yA	Right	yA specifies the Y-axis position which is in the right (starboard) direction.
zA	Down	zA specifies the Z-axis position which is in the down (toward the center of the Earth) direction.

6.2.29 Position3D_WGS84

Namespace: UMAA::Common::Measurement::Position3D_WGS84

Description: Position3D_WGS84 specifies a location relative to the WGS-84 ellipsoid.

Table 71: Position3D_WGS84 Structure Definition

Attribute Name	Attribute Type	Attribute Description
geodeticPosition	Position2D	geodeticPosition specifies a location on the surface of the Earth.
heightAboveEllipsoid	Altitude_HAE	heightAboveEllipsoid specifies the height above the WGS-84 ellipsoid.

6.2.30 ProductDisseminateFileObjectiveType

Namespace: UMAA::MM::BaseType::ProductDisseminateFileObjectiveType

Description: This structure is used to define goal specifying the action(s) required for sensor data dissemination.

Table 72: ProductDisseminateFileObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
destinationID	NumericGUID	Specifies the ID of destination resource
sourceURI	UniformResourceIdentifier	Specifies the location of the sensor product data file

6.2.31 ProductExploitationObjectiveType

Namespace: UMAA::MM::BaseType::ProductExploitationObjectiveType

Description: This structure is used to describe a clearly defined goal specifying the action(s) required to exploit a given payload sensor product or set of products.

Table 73: ProductExploitationObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
sourceURI	UniformResourceIdentifier	Specifies the location of the sensor product data file

6.2.32 Quaternion

Namespace: BasicTypes::Quaternion

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

Table 74: Quaternion Structure Definition

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

6.2.33 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 75: RecoveryObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
altitude	Altitude_MSL	Specifies the distance along the vehicle path to the waypoint

Attribute Name	Attribute Type	Attribute Description
altitudeAGL	Altitude_AGL	Specifies the distance above ground level
altitudeASF	Distance_ASF	Specifies the distance above sea level
position	Position2D	Specifies the location for recovering the vehicle

6.2.34 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 76: RouteObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
mode	HeightModeEnumType	describe whether the route is an altitude or depth mode
routeDescription	StringShortDescription	description of a route
waypoints	sequence<WaypointType>	Specifies the route the vehicle is to travel

6.2.35 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 77: StationkeepObjectiveType Structure Definition

Attribute Name	Attribute Type	Attribute Description
Additional fields included from UMAA::MM::BaseType::ObjectiveType		
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†	Duration_Seconds	Defines duration for station keeping

6.2.36 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 78: TaskPlanType Structure Definition

Attribute Name	Attribute Type	Attribute Description
dependentTaskIDs	sequence<NumericGUID>	List of tasks that must be adjudicated (e.g., completed, cancelled, etc.) before the referenced task can begin execution.
endTime	DateTime	Absolute time for which task must end. An optional tolerance for the end time may be defined.
endTimeTolerance†	DateTime_Tolerance	The earliest absolute time the task can end and the latest absolute time the mission can end.
name	StringShortDescription	A short name for the task.
objectives	sequence<ObjectiveType>	List of objectives associated with the task.
startTime	DateTime	Absolute time for which task must begin. An optional tolerance for the start time may be defined.
startTimeTolerance†	DateTime_Tolerance	The earliest absolute time the task can start and the latest absolute time the mission can start.
taskID*	NumericGUID	Unique identifier for the task within a mission.

6.2.37 TaskStatusType

Namespace: UMAA::MM::TaskExecutionStatus::TaskStatusType

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

Table 79: TaskStatusType Structure Definition

Attribute Name	Attribute Type	Attribute Description
endTime	DateTime	Specifies the end time for the task associated with missionID,taskID.
startTime	DateTime	Specifies the start time for the task associated with missionID,taskID.
state	TaskStateEnumType	Specifies the current state of the task specified by the associated task plan.
missionID*	NumericGUID	An identification of the mission
taskID*	NumericGUID	An identification of the task within the mission

6.2.38 WaypointType

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

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

Table 80: WaypointType Structure Definition

Attribute Name	Attribute Type	Attribute Description
altitude†	Altitude_MSL	Specifies the distance along the vehicle path to the waypoint
altitudeAGL†	Altitude_AGL	Specifies the distance above ground level

Attribute Name	Attribute Type	Attribute Description
altitudeASF†	Distance_ASF	Specifies the distance above sea level
attitude†	Orientation3D	includes yaw, pitch, roll that vehicle should assume at the given waypoint
depth†	Distance_BSL	Specifies the distance of the waypoint below sea level
endTime†	DateTime	Specifies the end of the valid time period
globalPosition†	Position2D	Specifies the global location of the waypoint (in latitude/longitude)
localPosition†	Position3D_PlatformXYZ	Specifies the local position of the waypoint (in x, y, z)
mode†	HeightModeEnumType	describe whether the waypoint is an altitude or depth mode
pathTolerance†	Distance	the current tolerance of the path measured by distance
speed†	Speed_LocalWaterMass	Specifies the speed to be maintained traveling to the waypoint
startTime†	DateTime	Specifies the beginning of the valid time period
waypointTolerance†	Distance	the current tolerance of the waypoint measured by distance.
waypointType	WaypointKindEnumType	A type of waypoint
waypointID*	NumericGUID	An unique identification of the waypoint

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 81: BearingAngleEnumType Enumeration

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

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 82: CommandStatusReasonEnumType Enumeration

Enumeration Value	Description
CANCELED	Indicates a transition to the CANCELED state when the command is canceled successfully.
VALIDATION_FAILED	Indicates a transition to the FAILED state when the command contains missing, out-of-bounds, or otherwise invalid parameters.
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.
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.
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.
INTERRUPTED	Indicates a transition to the FAILED state when the command has been interrupted by a higher priority process.
TIMEOUT	Indicates a transition to the FAILED state when the command is not acknowledged within some defined time bound.
SUCCEEDED	Indicates the conditions to proceed to this state have been met and a normal state transition has occurred.

6.3.3 ContingencyBehaviorEnumType

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

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

Table 83: ContingencyBehaviorEnumType Enumeration

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

6.3.4 DomainEnumType

Namespace: UMAA::Common::MaritimeEnumeration::DomainEnumType

Description: A mutually exclusive set of values that defines the area or region in which an unmanned platform operates in.

Table 84: DomainEnumType Enumeration

Enumeration Value	Description
AIR	Air
GROUND	Surface, ground
SURFACE	Surface, water
UNDERSEA	Undersea

6.3.5 HeadingReferenceEnumType

Namespace: UMAA::Common::MaritimeEnumeration::HeadingReferenceEnumType

Description: Defines a mutually exclusive set of values for the heading reference angle.

Table 85: HeadingReferenceEnumType Enumeration

Enumeration Value	Description
MAGNETIC_NORTH	Angle is relative to magnetic north
TRUE_NORTH	Angle is relative to true north
WIND_DIRECTION	Angle is relative to wind direction

6.3.6 HeightModeEnumType

Namespace: UMAA::Common::MaritimeEnumeration::HeightModeEnumType

Description: An enumeration that is used to set which height mode the unmanned vehicle should be operated in under the specified route.

Table 86: HeightModeEnumType Enumeration

Enumeration Value	Description
ALTITUDE	Height value is distance above sea-floor.
DEPTH	Height value is distance below sea surface.

6.3.7 HoverKindEnumType

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

Description: A mutually exclusive set of values that defines the loitering priority of the unmanned platform.

Table 87: HoverKindEnumType Enumeration

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

6.3.8 LineSegmentEnumType

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

Description: LineSegmentEnumTypeLDM is a Realization of LineSegmentEnumType which is a mutually exclusive set of values that defines the line segment types used for navigation.

Table 88: 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).

6.3.9 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 89: CommandStatusEnumType Enumeration

Enumeration Value	Description
FAILED	The command has been attempted, but was not successful.

Enumeration Value	Description
COMPLETED	The command has been completed successfully.
ISSUED	The command has been issued to the resource (typically a sensor or streaming device), but processing has not yet commenced.
COMMANDED	The command has been placed in the resource's command queue but has not yet been accepted.
EXECUTING	The command is being performed by the resource and has not yet been completed.
CANCELED	The command was canceled by the requestor before the command completed successfully.

6.3.10 LoiterKindEnumType

Namespace: UMAA::Common::MaritimeEnumeration::LoiterKindEnumType

Description: A Realization of LoiterKindEnumType which is a mutually exclusive set of values that defines the types of loiter patterns performed by a vehicle.

Table 90: LoiterKindEnumType Enumeration

Enumeration Value	Description
CIRCLE	The loiter type is a circular pattern.
RACETRACK	The loiter type is a racetrack pattern.

6.3.11 TaskStateEnumType

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

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

Table 91: TaskStateEnumType Enumeration

Enumeration Value	Description
PLANNED_PENDING_APPROVAL	The desired/reported state of the task is that a detailed mission plan with route has been created from the allocated tasks and approval is pending.
AWAITING_EXECUTION_APPROVAL	The desired/reported state of the task is that execution approval has been requested.
ALLOCATED	The desired/reported state of the task is that it has been allocated. The task allocation has been completed and approval has been granted.
EXECUTION_APPROVED	The desired/reported state of the task is that it has been approved for execution.
CANCELED	The desired/reported state of the task is that it has been cancelled.
COMPLETED	The desired/reported state of the task is that it has 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.).

Enumeration Value	Description
DROPPED	The desired/reported state of the task is that it has been dropped; the task is subject for reallocation within the UxS node.
PLANNED	The desired/reported state of the task is that it has been planned, indicating that the task is part of an approved and active detailed mission plan.
PROPOSED	The desired/reported state of the task is that it has been proposed to an allocation service.
QUEUED	The desired/reported state of the task is that it has been queued for execution.
EXECUTING	The desired/reported state of the task is that it 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 task. Once a task transitions to this state, it can no longer be reallocated to another UxS or vehicle unless it transitions to the FAILED state.
FAILED	The desired/reported state of the task is that it has failed. The UxS node has determined that no vehicles within the UxS can achieve the task.
UNALLOCATED	The desired/reported state of the task is that it is unallocated. The task could not be allocated to a system, or the task has just been created.
ALLOCATED_PENDING_APPROVAL	The desired/reported state of the task is that task allocation has been completed and approval is pending.
AWAITING_MISSION_PLAN	Used at initial state when there is no mission plan reported.
PAUSED	Used to pause the execution of an approved mission plan, approved mission task, or approved mission objective.
QUEUING	Used when mission plan, mission task, or mission objective is being queued (e.g., uploading to vehicle) for execution
PLANNING	Used when mission plan, mission task, or mission objective is still in the planning state.

6.3.12 WaypointKindEnumType

Namespace: UMAA::Common::MaritimeEnumeration::WaypointKindEnumType

Description: A mutually exclusive set of values that defines the types of waypoints in the route.

Table 92: WaypointKindEnumType Enumeration

Enumeration Value	Description
APPROACH_FINAL_POINT	Approach final point
APPROACH_INITIAL_POINT	Approach initial point
LAUNCH	Launch
LOITER	Loiter
NAV_ONLY	Navigation only
NAV_TARGET	Navigation target
OTHER	Other
RECOVERY	Recovery
RENDEZVOUS	Rendezvous

6.3.13 PlanActionEnumType

Namespace: UMAA::Common::Enumeration::PlanActionEnumType

Description: PlanActionEnumTypeLDM is a Realization of PlanActionEnumType which is a mutually exclusive set of values that defines types of actions for plan maintenance and management.

Table 93: PlanActionEnumType Enumeration

Enumeration Value	Description
ABORT_MISSION_LOAD_PLAN	The current mission should be aborted and the specified plan loaded in its place.
ADD_PLAN	The specified plan should be added to the mission plan repository.
DELETE_PLAN	The specified plan should be deleted from the mission plan repository.
DOWNLOAD_PLAN	The specified plan should be downloaded from the air vehicle to the mission plan repository.
RETRIEVE_PLAN	The specified plan should be retrieved from the mission plan repository.
UPDATE_PLAN	The specified plan should be updated in the mission plan repository.
UPLOAD_PLAN	The specified plan should be uploaded to the air vehicle.

6.3.14 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 94: WaterTurnDirectionEnumType Enumeration

Enumeration Value	Description
NO_VALID_TURN_DIRECTION	No valid turn direction is specified for the vehicle.
LEFT_TURN	The vehicle will make left turns.
RIGHT_TURN	The vehicle will make right turns.
VEHICLE_SPECIFIC	The vehicle will make turns as dictated by the vehicle's specific behavior.
INTO_THE_CURRENT	The vehicle will make turns into the current.
INTO_THE_WIND	The vehicle will make turns into the wind.

6.3.15 ZoneKindEnumType

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

Description: Defines a mutually exclusive set of zone kinds.

Table 95: ZoneKindEnumType Enumeration

Enumeration Value	Description
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 96: Type Definitions

Type Name	Primitive Type	Range of Values	Description
Angle	double	fractionDigits=3 maxInclusive=3.141592653589 7932384626433832795 minInclusive=-3.141592653589 7931264626433832795 units=Radian referenceFrame=Counting	Angle 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	units=N/A minInclusive=N/A maxInclusive=N/A fractionDigits=N/A length=N/A	BooleanEnumTypeLDM is a Realization of BooleanEnumType which is 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=0 maxInclusive=18446744073709 500000 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	units=Meter minInclusive=0 maxInclusive=401056000 fractionDigits=3	This type stores a distance in meters.
Distance_ASF	double	units=Meter minInclusive=0 maxInclusive=401056000 fractionDigits=3	The altitude or distance above the sea floor in meters.
Distance_BSL	double	units=Meter minInclusive=0 maxInclusive=10000 fractionDigits=3	The distance below sea level in meters.
Down	double	axisAbbrev=Z axisDirection=down axisUnit=Meter maximumValue=50000 minimumValue=-50000 rangeMeaning=exact resolution=0.001	The Down axis is used for measuring position and increases in magnitude as values extend toward the center of the Earth. Down measurements are expressed in meters.
Duration_Hours	double	units=Hour minInclusive=0 maxInclusive=10505 fractionDigits=3	Represents a time duration in hours.

Type Name	Primitive Type	Range of Values	Description
Duration_Seconds	double	units=Seconds minInclusive=0 maxInclusive=37817280 fractionDigits=6	Represents a time duration in seconds.
EllipsoidalHeight	double	axisAbbrev=h axisDirection=up axisUnit=Meter maximumValue=700000 minimumValue=-10000 rangeMeaning=exact resolution=0.001	The EllipsoidalHeight axis is used for measuring position and increases in magnitude as values extend away from the center of the Earth. EllipsoidalHeight measurements are expressed in meters.
Forward	double	axisAbbrev=X axisDirection=fore axisUnit=Meter maximumValue=20000000 minimumValue=-20000000 rangeMeaning=exact resolution=0.001	The Forward axis is used for measuring position and increases in magnitude as position extends out the "front" of the reference body. Forward 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	The Latitude axis is 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	The Longitude axis is 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	units=MeterPerSecond minInclusive=0 maxInclusive=200 fractionDigits=6	This type stores speed in meters/s.
Heading_TrueNorth_Angle	double	units=Radian referenceFrame=TrueNorth minInclusive=-3.1415926535897932626433832795 maxInclusive=3.1415926535897932626433832795 fractionDigits=3	Describes heading as a value between -pi and pi with respect to True North.
MSLHeight	double	axisDirection=up axisUnit=Meter maximumValue=700000 minimumValue=-10000 rangeMeaning=exact resolution=0.001	The MSLHeight axis is used for measuring position and increases in magnitude as values extend away from the center of the Earth. MSLHeight measurements are expressed in meters.

Type Name	Primitive Type	Range of Values	Description
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
Order	long	units=N/A minInclusive=0 maxInclusive=2147483647 fractionDigits=0	Represents nonnegative integers.
Pitch_HalfAngle	double	fractionDigits=3 maxInclusive=1.570796326794 8966192313216916398 minInclusive=-1.570796326794 8966192313216916398 units=Radian referenceFrame=PlatformNED	Pitch_HalfAngle 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.
Power_Percent	double	units=Percent minInclusive=0 maxInclusive=1000 fractionDigits=3	Defines a percentage 100% = 100.0. Values greater than 100% are allowed.
RadarHeight	double	axisDirection=up axisUnit=Meter maximumValue=700000 minimumValue=-10000 rangeMeaning=exact resolution=0.001	The RadarHeight axis is used for measuring position and increases in magnitude as values extend away from the center of the Earth. RadarHeight measurements are expressed in meters.
RelativeAngle	double	fractionDigits=3 maxInclusive=3.141592653589 7932384626433832795 minInclusive=-3.141592653589 7931264626433832795 units=Radian referenceFrame=Counting	RelativeAngle specifies the angle between two intersecting rays. The measurement is stated in radians between -pi and pi.
Right	double	axisAbbrev=Y axisDirection=starboard axisUnit=Meter maximumValue=20000000 minimumValue=-20000000 rangeMeaning=exact resolution=0.001	The Right axis is used for measuring position and increases in magnitude as position extends out the "right" of the reference body. Right measurements are expressed in meters.
Roll_Angle	double	fractionDigits=3 maxInclusive=3.141592653589 7932384626433832795 minInclusive=-3.141592653589 7931264626433832795 units=Radian referenceFrame=PlatformNED	Roll_Angle 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.

Type Name	Primitive Type	Range of Values	Description
Speed	double	units=MeterPerSecond minInclusive=0 maxInclusive=299792458 fractionDigits=6	This type stores speed in meters/s.
Speed_LocalWaterMass	double	units=MeterPerSecond minInclusive=0 maxInclusive=299792458 fractionDigits=6	This type stores speed in meters/s.
StringLongDescription	string	fractionDigits=N/A length=4095 maxExclusive=N/A maxInclusive=N/A minExclusive=N/A minInclusive=N/A units=N/A	Represents a long format description.
StringShortDescription	string	fractionDigits=N/A length=1023 maxExclusive=N/A maxInclusive=N/A minExclusive=N/A minInclusive=N/A units=N/A	Represents a short format description.
UniformResourceIdentifier	string	fractionDigits=N/A length=2047 maxExclusive=N/A maxInclusive=N/A minExclusive=N/A minInclusive=N/A units=N/A	Represents a Uniform Resource Identifier (URI).
Yaw_PosAngle	double	fractionDigits=3 maxInclusive=6.283185307179586364925286766559 minInclusive=0 units=Radian referenceFrame=PlatformNED	Yaw_PosAngle specifies the platform's rotation about the Z axis of its body axis system (PlatformXYZ) relative to its velocity vector in the X-Y plane of its body axis system. Yaw is positive in a clockwise direction. The measurement is stated in radians between 0 and 2 pi.

A Appendices

A.1 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
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
WMO	World Meteorological Organization