An OMG® Command and Control Message Specification™ (C2MS™) Publication

Command and Control Message Specification™ (C2MS™)

Version 1.1

OMG Document Number: dtc/24-06-02
Release Date: May 2024
Normative Reference: https://www.omg.org/spec/C2MS/
Associated Normative Machine Consumable Files:

https://www.omg.org/spec/C2MS/20240501/C2MS.xmi

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web: OMG Specifications, report an issue.

Table of Contents

Tables

Figures

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at https://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL:

https://www.omg.org/spec

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Issue Reporting

The reader is encouraged to report any technical or editing issues/problems with this specification find by completing the Issue Reporting Form listed on the main web: OMG Specifications, report an issue.

Introduction to Specification

The objective of this C2MS (Command and Control Message Specification) standard is to establish common format specifications to allow for common data exchange interfaces for integrating satellite mission ground data system products from multiple vendors and system developers. The formats may be of benefit for system-internal interface definitions and for communications between systems.

This C2MS standard was originally created as part of a National Aeronautics and Space Administration (NASA) project (called the Goddard Mission Services Evolution Center, or "GMSEC") whose mission is to provide a framework to enable flexible and cost-effective means to meet the operational needs of current and future missions. This includes single satellite missions, satellite fleets, and future constellation missions. In order to provide rapid and flexible mission development and continued operations throughout a mission's lifecycle, the GMSEC project architecture incorporates a concept that supports simplified component integration and system communications. (see Figure I-1 Original GMSEC API/Bus Features and Figure I-2 GMSEC Architecture Ground System Software Categories, Notional). The GMSEC architecture is a middleware-based system architecture using standardized messages, the GMSEC application programming interface, and Commercial off the shelf (COTS) and Government off the shelf (GOTS) middleware and functional components. The GMSEC Application Programming Interface (API) enables components to have a uniform interface to the underlying middleware and isolates the components from the middleware (see Figure I-3 GMSEC Middleware Abstraction Layers). Using the standard messages along with the GMSEC API allows a component to be GMSEC compliant and helps it achieve plug and configure compatibility. For a component to be considered GMSEC compliant, it must use the standard set of messages and the GMSEC API.

This document defines the standardized messages developed as part of the GMSEC architecture as well as interaction patterns for their effective usage. The actual GMSEC architecture and software are not part of this OMG standard.

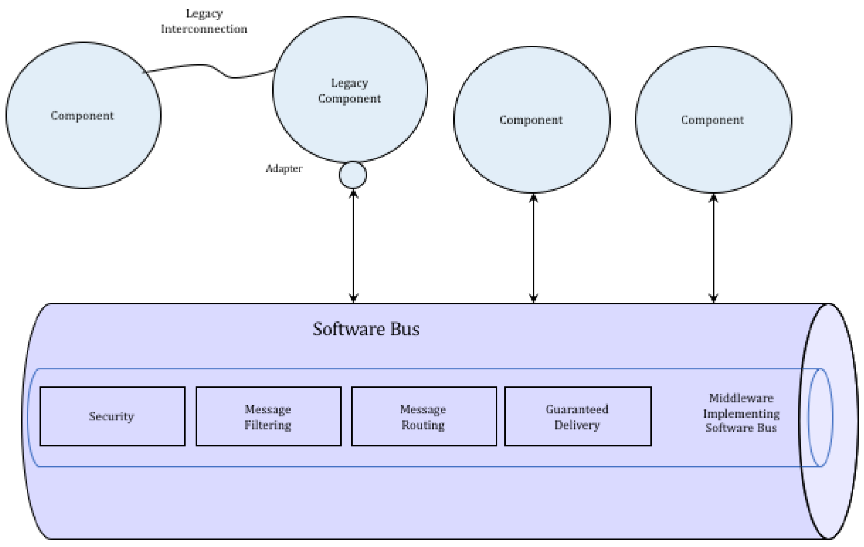


Figure I-1. Original GMSEC API/Bus Features

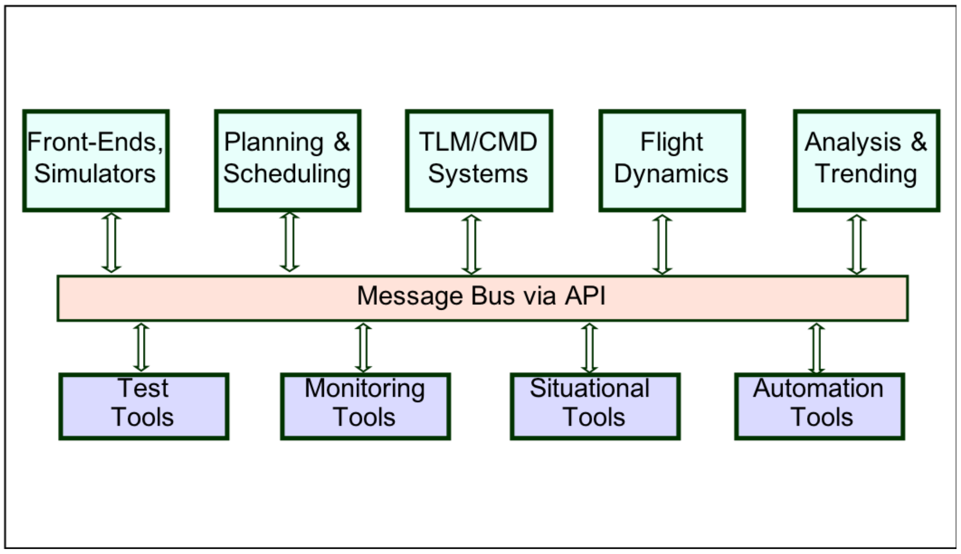


Figure I-2. GMSEC Architecture Ground System Software Categories, Notional

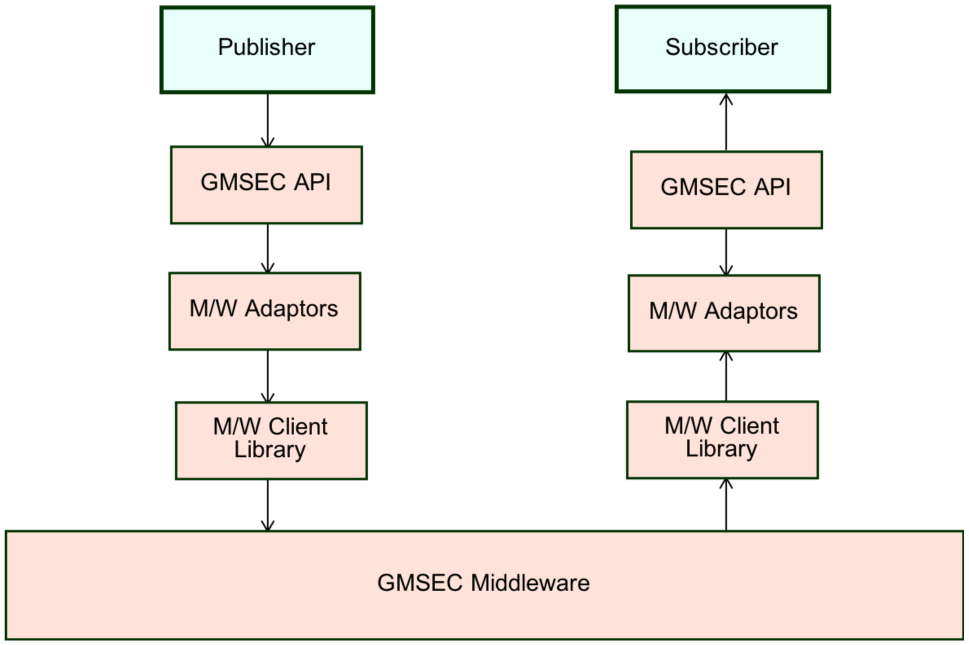


Figure I-3. GMSEC Middleware Abstraction Layers

Applications communicate with one another through the GMSEC API (via the underlying middleware) using standard messages. Each of these messages includes a specific subject that identifies which message standard defines both the content and meaning of the message. This “subject” is sometimes referred to as the “subject name”, “routing header”, or “header 0”. Applications send messages by providing the message and message subject to the GMSEC API, which in turn provides it to the middleware.

The middleware takes the responsibility of routing a copy of any message with that subject that appears on the software bus to the requesting application(s). Applications receive messages by providing the requested message subject via the GMSEC API to the middleware – this is also referred to as “subscribing to a particular subject.” This subject-based message addressing promotes loose coupling: message producers need not know the location, quantity, or platform of message consumers; and message consumers need not know the details of message producers.

The NASA GMSEC Architecture Document (see Section 3.2 Non-Normative References) provides a detailed description of the concepts, high-level design, and application of the GMSEC reference architecture. Other development efforts could result in non-C2MS-compatible systems.

1 Scope

This document, the Command and Control Message Specification (C2MS), is the definition of the standardized messages along with interaction patterns for their use for common interfaces found in typical satellite ground data systems. This promotes platform independence that allows easy plug and play intercommunication among components. The Platform Independent Model (PIM) simply describes the messages and interactions necessary to communicate between components.

Note that NASA has developed a Platform Specific Model (PSM) referred to as GMSEC. The GMSEC API software, selected components, and documentation are distributed by NASA. Others are free to develop alternative PSMs.

2 Conformance

The primary point of conformance is support of the PIM. Conformance to any defined PSM is optional, but if a defined platform is used, such as eXtensible Markup Language (XML) or JavaScript Object Notation (JSON), the implementation must conform to the appropriate PSM. In the event that a PSM does not exist for a specific protocol, implementers are encouraged to define a PSM and submit it for standardization to the OMG.

3 References

3.1 Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

XML Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, and Eve Maler, editors. *Extensible Markup Language (XML) 1.0 (Fourth Edition).* World Wide Web Consortium, 2000. (See http://www.w3.org/TR/2006/REC-xml-20060816.)

ECMA – 404 The JSON Data Interchange Format, 1st Edition / October 2013, ECMA International (See http://www.ecma-international.org/publications/standards/Ecma-404.htm)

CCSDS - In relation to telemetry data formats, particularly the Consultative Committee for Space Data Systems (CCSDS) frames and packets and the contents of the C2MS navigation messages, the CCSDS Recommendations and Reports should be referenced (See http://www.ccsds.org)**.** For navigation data messages, see the following specifications:

- Attitude Data Messages 504.0-B-1
- Orbit Data Messages 502.0-B-2
- Tracking Data Message 503.0.B-1

XTCE - XML Telemetric and Command Exchange (XTCE) - The XML Telemetric and Command Exchange (XTCE) data specification provides an information model for telemetry and command data. This OMG specification defines a standard exchange format for telemetry and commanding that will support the exchange of data through all phases of the satellite, payload, and ground segment lifecycle: system design, development, test, validation, and mission operations (see https://www.omg.org/space/xtce/).

3.2 Non-Normative References

3.2.1 NASA GMSEC Documents

The following GMSEC documents set forth the NASA GMSEC Architecture and the GMSEC Applications Programming Interface User's Guide. Documents for specific GMSEC-compliant software components should be consulted on an individual basis.

- GMSEC Architecture Document, Release 2.8.1, February 2014
- GMSEC API 4.3 User's Guide, May 2017

4 Terms and Definitions

The following table lists terms and descriptions for selected acronyms and abbreviations used in this document.

Table 4-1. Acronyms and Abbreviations

Term	Description
ACK	Acknowledge or Acknowledgement
ANL	Analysis
AOS	Acquisition of Signal
API	Application Programming Interface
APID	CCSDS Application Process Identifier
ARC	Archive
AST	Assessment
C2CX	Component-to-Component Transfer
C2MS	Command and Control Message Specification
CCSDS	Consultative Committee for Space Data Systems
CFG	Configuration
CFG	Configuration Control and Management
CNTL	Control
COTS	Commercial off the shelf
CRYPT	Encryption
CVCDU	Coded Virtual Channel Data Unit
CVT	Current Value Table
DEV	Device
EPH	Ephemeris
EU	Engineering Units
EUI	Extended Unique Identifier
FEP	Front End Processor
GMSEC	Goddard Mission Services Evolution Center
GOTS	Government off the shelf
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
HB	Heartbeat
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol

Term	Description
JSON	JavaScript Object Notation
LOS	Loss of Signal
LRV	Last Received (or recorded) Value
MAC	Media Access Control
MAN	Maneuver Planning
ME	Miscellaneous Element
MEP	Message Exchange Pattern
MON	Monitor
MSG	Message
NAC	Navigation and Control
NASA	National Aeronautics and Space Administration
OD	Orbit Determination
OMG	Object Management Group
OS	Operating System or Systems
PAGE	Paging
PIM	Platform Independent Model
PSM	Platform Specific Model
PTA	Plotting, Trending and Analysis
REQ	Request or Required
RESP	Response
RPY	Replay
RSRC	Resource
RT	Real-time
RULE	Rule-Action
SCH	Schedule or Scheduling
SCID	CCSDS Spacecraft Identifier
SDTF	Space Domain Task Force
SIM	Simulation or Simulator
SQL	Structured Query Language
T&C	Telemetry and Command
TDM	Time Division Multiplexing or Tracking Data Message
TLM	Telemetry
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UTC	Coordinated Universal Time
VCID	CCSDS Virtual Channel Identifier
VER	Verify
Wrt	With respect to
XML	eXtensible Markup Language
XTCE	XML Telemetric and Command Exchange

The following table describes selected terms used in this document. The descriptions below are summaries for quick identification. All terms are defined in more detail elsewhere in this Specification.

Table 4-2. Glossary

Term	Description
C2MS	A Command and Control Message Specification standard to establish common format specifications to allow for common data exchange interfaces for integrating satellite mission ground data system products from multiple vendors and system developers.
Message Exchange Pattern	A description of how C2MS messages can be sent between components.

Term	Description
Message Type	Three fundamental C2MS message types (message, request, and response) are defined in C2MS and can be used in various combinations with one another to create an infinite number of message exchange patterns. In turn, these message exchange patterns can be used in the description of the interfaces for any number of services.
Miscellaneous Elements	Application programs should be able to define their own set of unique elements of the subject name in order to create their own unique subject names. Therefore, a C2MS-defined subject name contains a fixed portion and a variable portion of miscellaneous elements.
OMG	An open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments.
Subject Name	Also known as subject, topic, routing header, and header 0; this is routing information used by the middleware/transport layer and provides for the publish/subscribe architecture.
Tracking	Tracking, one of the message classes composed of aggregated UML classes, is reserved for Application Programming Interface usage. Message classes are composed of aggregated UML classes that show whether the fields are required, optional, tracking, or dependent.

5 Additional Information

5.1 Acknowledgements

The OMG would like to thank the following organization for creating this specification:

NASA Goddard Space Flight Center (GSFC)

6 PIM – C2MS Subject Names and Message Composition

6.1 Overview

The C2MS defines a standard, platform independent model (PIM) for communication among various components. The C2MS model does not presume or try to define a specific system level architecture. Instead, it defines generic concepts such as messages and parameters that are relatively simple to implement; this provides system integrators common ways to connect heterogeneous suites of space related software. The C2MS PIM consists of message classes that define the messages' contents so that the user can create, send, and receive messages.

Each C2MS defined message is composed of three pieces: A C2MS Subject Name that identifies a message, the C2MS Message Header that is common to all messages and a C2MS Message Body Content portion that is unique for each message. In common pub/sub systems, the subject name is used for routing purposes.

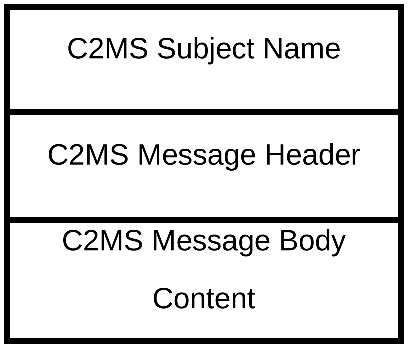


Figure 6-1. C2MS Defined Message

The Message Header and Message Body Content portions are comprised of fields. One of the fields in both message portions is a version number. The version number identifies the iteration of the message definition and is shown below in expanded format.

6.2 Subject Names

Subject names are constructed to provide the subscribing application(s) efficient flexibility to filter messages.

Filtering may occur at two levels: the middleware level and the application level. Filtering takes place at the middleware level based solely upon the subject name. The middleware typically does not extract information from the message contents - it routes messages based solely on the subject contents.

Filtering at the application level takes place based upon the message contents. Applications should be built and subjects defined so as to maximize filtering at the subject level by the middleware. Filtering of messages at the application level may be necessary for some applications, but should be minimized to maximize system efficiency.

The following table suggests some subject name elements (in addition to the message type and subtype) to which a subscribing application could subscribe in order to limit the number of messages that it receives. Not every message or possibility is listed.

Table 6-1. Sample Subject Filtering Items in Addition to Type and Subtype

Message	Subject Filtering Items
Real-time Log Message	• Mission • Constellation • Satellite • Publisher • Occurrence type • Severity
Archive Message Retrieval Request and Response	• Responder • Requestor • Response Status (ACK, Working, Success, Failure, Invalid, Final)
Directive Request and Response	• Requestor • Response Status (ACK, Working, Success, Failure)
Telemetry Messages	• Mission • Constellation • Satellite • Publisher • Telemetry format • Stream mode • Channel or APID
Replay Telemetry Message	• Mission • Constellation • Satellite • Publisher • Telemetry format • Stream mode • Channel or APID
Mnemonic Value Data Message, Request, and Response	• Mnemonic • Requestor
Archive Mnemonic Value Data Message, Request, and Response	• Mission • Constellation • Satellite • Requestor

Message	Subject Filtering Items
Product, Product Request, and Product Response	• Mission • Constellation • Satellite • Publisher • Product Type and Subtype

6.2.1 Characteristics of Subject Names

1. C2MS messages must be easily distinguishable by subject; short elements are encouraged for fast parsing and efficient throughput.

2. In order for the middleware to distinguish standard C2MS messages from other routable messages, an indicator or element should be present in the subject name. Thus, C2MS standardized messages can be filtered and routed apart from other non-C2MS messages.

3. Most applications will filter messages by subject so a common set of filtering parameters would be beneficial. But a common set of parameters may not be applicable to all messages' subject names. Therefore, applications will use the common set of parameters, but also be able to expand upon these common filtering parameters with their own unique filtering parameters.

   Examples of the common filtering parameters include: constellation and/or satellite, message type and subtype. A subject name should contain elements with these common distinguishing characteristics.

4. A means of distinguishing telemetry data streams (or other data/messages streams) is needed at the subject name level so that an application can subscribe to one or more data/message streams.

An application may want to subscribe to a telemetry data stream (or a subset of data streams):

- That has not started flowing yet; that has a single telemetry format or a subset of telemetry formats; that is from a single satellite or a constellation of satellites;
- That contains real-time or playback data;
  or
- That is a subset of mnemonics from a single satellite.

A configuration table of active or future (expected) data streams could assign a unique Stream Identifier (ID) to each data stream that the publisher would then include in the subject name. A subscriber could look up the Stream ID in the table and subscribe to the subject that includes that unique ID.

Furthermore, the publisher of the data may (request to) update the data stream table with the subject by which the data will be published. (This could also be used to distinguish a real-time stream from a playback stream. Subscribers must also determine if they need to unsubscribe once the data stream has ceased.)

6.2.2 Format of C2MS Subject Names

Based on the characteristics of the C2MS Subject Names and on existing subject (topic) conventions, the following subject name format rules are used:

Subject names follow the format of a string of characters separated by the dot ("`.`") character. The character strings separated by the dots are called elements.

- `THIS.IS.A.VALID.SUBJECT` (This subject has 5 elements)

Only alphanumeric (uppercase or lowercase), underscore ("_"), and dash ("-") characters are valid. No invisible or control type characters are used. No element is empty. If an element is required but not applicable for that mission or to that type of message, the publisher inserts "FILL" (no quotes) for that element. For example, most but not all messages are satellite related, so a "Satellite ID" is part of all message's subject names. If a message is not satellite specific, the publisher inserts "FILL" into that portion of the subject name.

- `THIS.IS.NOT.A.VALID.SUBJECT.` (missing element at the end)
- `.THIS.IS.NOT.A.VALID.SUBJECT` (empty element at beginning)
- `THIS.IS..NOT.A.VALID.SUBJECT` (empty element in middle)
- `C2MS.FILL.VALID.SUBJECT` (valid C2MS subject)

Subject names are case-sensitive, so "sat1" is not equal to "SAT1" and neither is equal to "Sat1". Subject names must be at least one character in length.

A C2MS subject is distinguished from other subjects by the first element, which contains the capitalized text "C2MS" (no quotes). Otherwise uppercase or lowercase are valid.

- `C2MS.VALID.SUBJECT` (valid C2MS subject)
- C2ms.not.valid.subject (invalid C2MS subject name, not uppercase "C2MS")

- C2MS.valid.subject (valid C2MS subject, uppercase and lowercase both valid)

An application can subscribe to subjects using various wildcards. Wildcard characters are not used by publishing applications. The wildcard rules are as follows:

An asterisk (*) can take the place of exactly one whole element but not a substring of an element. The asterisk **DOES NOT** have to be the right-most character.

- `THIS.*.VALID` (valid wildcard substitution)
- `THIS.IS*.NOT.VALID` (invalid wildcard substitution)

A greater-than (>) character can appear ONLY as the right-most character of a subject immediately after the element delimiter ("**.**") and will match any subject with one or more elements to the right.

- `THIS.IS.VALID.>` (valid wildcard substitution)
- `THIS.IS.>.NOT.VALID` (invalid wildcard substitution)

A plus (+) character can appear ONLY as the right-most character of a subject immediately after the element delimiter ("**.**") and will match any subject with zero or more elements to the right.

- `THIS.IS.VALID.+` (valid wildcard substitution)
- `THIS.IS.+.NOT.VALID` (invalid wildcard substitution)

The following examples illustrate the use of the "*", ">", and "+" wildcard syntax and the matching semantics.

Table 6-2. Asterisk, Greater Than, and Plus Sign Wildcard Syntax Examples

Example	Comments
`THIS.IS.VALID.*`	Match the first three elements and any fourth element. Here, subjects with more than 4 elements will not match.
`THIS.IS.VALID.>`	As long as the first three elements match, will match any subject of any greater length. Subjects of three elements will not match.

Example	Comments
`THIS.IS.VALID.+`	As long as the first three elements match, will match any subject of any greater or equal length. Subjects of three elements will match.

Table 6-3. Subject Name Matching Examples

Subject	Matching Subjects	Non-Matching Subjects	Non-Matching Reason
`ONE.TWO.*`	ONE.TWO.THREE	ONE.TWO.THREE.FOUR	Extra element
	ONE.TWO.SEVEN	ONE.TWO	Missing element
	ONE.TWO.TWO	ONE.TWOTHREE.FOUR	Non-matching second element
`ONE.>`	ONE.TWO	TWO.ONE	Position mismatch
	ONE.TWO.THREE	ONE	Missing element
	ONE.TWO.XYZ.FIVE	ONEZ.TWO	Non-matching first element
`ONE.+`	ONE	TWO.ONE	Position mismatch
	ONE.TWO	ONEZ.TWO	Non-matching first element
	ONE.TWO.XYZ.FIVE		

NOTE: As guidance, in order to maximize speed and throughput rates, subject names should be short and not use an extraordinary number of elements.

- The length of an element should not exceed 25 characters, except when it represents a GUID.
- The length of a subject is dependent on the number of elements.

6.2.3 C2MS Subject Name Standard

A few common elements of the subject can be identified that would (nearly) always be included in the subject name. They are:

- Subject standard
- Domain1 and Domain2
- Mission, Constellation, and Satellite IDs
- Message type
- Message subtype

Additionally, application programs should be able to define their own set of unique elements of the subject name in order to create their own unique subject names. Therefore, a C2MS-defined subject name contains a fixed portion and a variable portion of elements, and is defined as follows:

Table 6-4. C2MS Subject Name Definition

Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***					
Specification	DOMAIN1	DOMAIN2	MISSION	CONST	SAT	TYP	SUBTYP	*ME1*	*ME2*	*ME3*	*ME4*	*ME5*	*ME6*

Fixed portion
All elements required

Variable portion
Each message definition determines whether a *Miscellaneous Element* is required or optional

In text format the subject would appear as:

```
SPECIFICATION.DOMAIN1.DOMAIN2.MISSION.CONST.SAT.TYPE.SUBTYPE.ME1.ME2.ME3….
```

The first eight subject elements, the **Specification, Domain1, Domain2, Mission, Const, Sat, Type, and Subtype** are fixed. These elements are always defined the same and are required to be filled in by the publisher/sender of a message.

The elements to the right of the fixed portion of the C2MS subject are the ***Miscellaneous Elements*** and are the variable portion of the subject. The Miscellaneous Elements are message and subscriber specific. That is, the publisher/subscriber (sender/receiver) would predefine or even dynamically create as many Miscellaneous Elements as needed according to their filtering needs. Thus, depending on the message definition, they can have different meanings, can vary in number, and can be either required or optional.

If an element is required but not applicable for that mission or to that type of message, the publisher inserts "FILL" (no quotes) for that element. For example, most, but not all messages are satellite related, so a "Satellite ID" would be part of all message's subject names. If a message were not satellite specific, the publisher would insert "FILL" into that portion of the subject name.

6.2.3.1 Subject Standard Element of the C2MS Subject Name

The Subject Standard element of the subject name identifies the specification used. This interface specification document is the standard by which the C2MS Subject Name is defined and interpreted. For C2MS defined subject names, the first element is always “C2MS”.

A C2MS Subject Name will be distinguished from other subject names by the first element, which shall contain the capitalized text “C2MS” (no quotes). For example:

- `C2MS.VALID.SUBJECT` (valid C2MS subject)
- C2ms.not.valid.subject (invalid C2MS subject name, not uppercase "C2MS")

6.2.3.2 Domain Elements of the C2MS Subject Name

The two domain elements, Domain1 and Domain2, in the subject name allow logical separation of messages and access control rules to mission-specific messages based on physical or logical domains within a mission or enterprise’s architecture. This may include instances of the same mission system operating in different modes, e.g., operations and backup operations. Missions should coordinate their use for messages outside of specific areas as allowed for situational awareness, testing, etc.

6.2.3.3 Mission Elements of the C2MS Subject Name

Three elements comprise the fixed **Mission Elements**. They are described in the table below.

Table 6-5. Descriptions of the Mission Elements of the C2MS Subject Name

Element Name	Value	Description
Mission	[Name of mission]	Name of a mission. E.g., MyMission
Const	[Name of constellation]	Name of constellation, e.g. MyConst1
Sat	[Name of satellite]	Name of a satellite. E.g., MySat1

For single satellite missions, the mission identifier, constellation, and satellite ID may be different, or may be identical. Or the mission may choose to name the satellite by adding “1” to the mission name. For example:

```
C2MS.D1.D2.Sun.C1.HELIO.MSG.LOG…
```
or,
```
C2MS.D1.D2.Sun.C1.SUN.MSG.LOG…
```
or,
```
C2MS.D1.D2.Sun.SUN1.SUN1.MSG.LOG…
```

Some missions may consist of a fleet or a constellation of satellites or multiple constellations and can be distinguished in the following manner similar to a product manufacturer’s model and serial number identification:

```
C2MS.D1.D2.Moon.Const1.LUNAR1
C2MS.D1.D2.Moon.Const1.LUNAR2
C2MS.D1.D2.Moon.Const1.LUNAR3

C2MS.D1.D2.Mars.C1.11
C2MS.D1.D2.Mars.C1.12
```

```
C2MS.D1.D2.Mars.C1.13

C2MS.D1.D2.MSN1.CONST-A.Sat1
C2MS.D1.D2.MSN1.CONST-B.Sat1
C2MS.D1.D2.MSN1.CONST-A.Sat2
C2MS.D1.D2.MSN1.CONST-B.Sat2
C2MS.D1.D2.MSN1.CONST-A.Sat3
C2MS.D1.D2.MSN1.CONST-B.Sat3
```

It is important to note that the subject naming convention for the "Sat" element of the subject does not have to refer to a single physical satellite, though that may be a common way of using the "Sat" element. "Sat" could refer to the physical satellite (perhaps by flight model number), to a logical satellite name such as "CONTROLLER", "PRIME", "EAST", "RING-A1", and "SPARE2".

Creative use of the "Mission", "Const", and "Sat" elements to refer to a physical, logical, group (subset), or entire constellation of satellites is possible and permits great latitude for categorization and unique identification of assets.

In generic terms, the Mission Elements are simply the taxonomy of classifying groups (or sets) and group members (elements of the sets).

For services not associated with a constellation or a satellite, place the service name in the "Sat" element.

6.2.3.4 Message Elements of the C2MS Subject Name

Two elements comprise the fixed **Message Elements**.

The **Type** element is used to describe the kind of message communication used within C2MS. The available values are **Request**, **Response**, and **Message**. A **Message** is published without a required or expected response, though it may cause an action when received. A **Request** message is used to request a specific action or information from a data provider or product generator. A **Request** message may or may not require a **Response** message. Message types are discussed in detail in Section 6.4 C2MS Messages: Their Characteristics and Interactions.

The **Subtype** element contains the ID or name of the message definition. The **Message Elements** are summarized in the table below.

Table 6-6. Descriptions of the Message Elements of the C2MS Subject Name

Element Name	Value	Description
Type Kind or intention of communication	**MSG**	Message
	REQ	Request
	RESP	Response
Subtype Name or ID of C2MS Defined Message	**AMSG**	Archive Message Retrieval
	AMVAL	Archive Mnemonic Value Retrieval
	CMD	Command
	CMDECHO	Command Echo
	CFG	Configurtion Status
	CNTL	Control
	DEV	Device

Element Name	Value	Description
	DIR	Directive
	HB	Heartbeat
	LOG	Log (or event)
	NDM	Navigation Data Message
	MVAL	Mnemonic Value
	PROD	Product
	RSRC	Resource
	RTLM	Replay Telemetry
	SERV	Simple Service
	TLMPKT	Telemetry CCSDS Packet
	DEPRECATED TLMFRAME	Telemetry CCSDS Frame
	TLMCCSDS-CADUFRAME	Telemetry CCSDS CADU Frame
	TLMCCSDS-TRANSFERFRAME	Telemetry CCSDS Transfer Frame
	TLMTDM	Telemetry TDM Frame
	TLMPROC	Telemetry Processed Frame

6.2.3.5 *Miscellaneous Elements* of the C2MS Subject Name

Each individual message definition in Section 8 specifies whether a Miscellaneous Element is required or optional and how those fields should be populated. Some typical uses of the miscellaneous elements are shown in the table below and following text.

Table 6-7. Message Type Determines Content of the *Miscellaneous Elements*

		Miscellaneous Elements			
Message Type	**Meaning**	***ME1***	***ME2***	***ME3***	***ME4…***
REQ	**Request**	Responder	Undefined		
RESP	**Response**	Requestor	Status	Undefined	Undefined
MSG	**Message**	Publisher	Message specific		

If TYPE = REQ, then	***ME1*** = component (or group or service) name of responder
	ME2, ME3*,* ... undefined
If TYPE = RESP then	***ME1*** = component (or group) name of requestor
	ME2 = status of the request
	ME3, ME4*,* … undefined
If TYPE = MSG, then	***ME1*** = component (or group) name of publisher
	ME2, ME3*,* ... message specific

The other primary factor that determines the value of the ***Miscellaneous Elements*** is the Message Subtype – the abbreviated name of the message. The ***Miscellaneous Elements*** are further defined in Section 8 PIM – Message Definitions, where the C2MS messages are defined.

For the *ME1* variable element, group names can also apply. Components may organize themselves into logical associations or groups (pre-defined or dynamically formed) for more creative forms of communication. For example, any one member of a group may need to send or receive a message to/from all members of the group.

Using group names will make message subscribing much simpler. This concept is nearly identical to that described in Section 6.2.3.2 Domain Elements of the C2MS Subject Name, where the “Sat” element could be used to refer to a physical, logical, group, or constellation of satellites.

6.3 Message Fields

6.3.1 Field Names

Field names represent the name of an item contained in a C2MS Message. A generalized field-naming convention has been implemented that adheres to the following rules:

- For each message, all field names are unique.
- Field names consist of alphanumeric characters, dashes (“-”), and dots/periods (“.”).
- Alpha characters are capitalized.

6.3.2 Required, Optional and Dependent Fields

Fields are classified as Required, Optional, or Dependent. The required fields must be present in order to be compliant with C2MS.

An optional field may or may not be included in a message. Optional fields may be useful to the Receiver and may be implemented as necessary. Software components, missions, or interface definitions may determine if these fields are required for their particular needs and applications.

Dependent fields are optional fields that become required based on the presence or values of other fields in the message. This information will be documented in the specific Message section where applicable.

Field information tables for each message include a column, "Presence" that will indicate a R, O, or D as appropriate for the given field.

UML diagrams throughout this document use multiplicity designators to indicate if a field is required or optional. Default multiplicity is '1' if not shown and indicates a required field. In the figure below, EVENT-TIME is a required field, using default multiplicity of '1', while SPACECRAFT-TIME is optional, as shown by a multiplicity of either '0' or '1'.

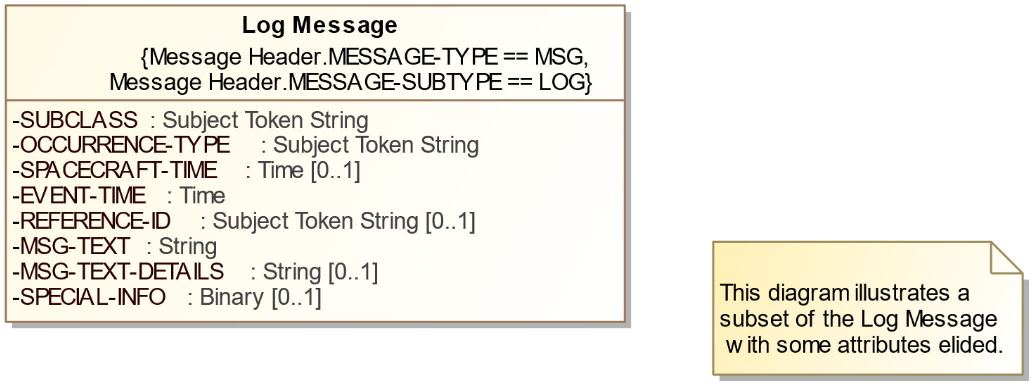

Log Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == LOG}
-SUBCLASS : Subject Token String
-OCCURRENCE-TYPE : Subject Token String
-SPACECRAFT-TIME : Time [0..1]
-EVENT-TIME : Time
-REFERENCE-ID : Subject Token String [0..1]
-MSG-TEXT : String
-MSG-TEXT-DETAILS : String [0..1]
-SPECIAL-INFO : Binary [0..1]
This diagram illustrates a subset of the Log Message with some attributes elided.

Figure 6-2. Message Field Multiplicity

6.3.3 Series Fields

Some fields contain a series of items. In this case another preceding field designates the number of items in the series, as shown in the following example:

```
NUM-OF-MNEMONICS
MNEMONIC.n.NAME
MNEMONIC.n.NUM-OF-SAMPLES
MNEMONIC.n.SAMPLE.m.TIME-STAMP
MNEMONIC.n.SAMPLE.m.RAW-VALUE
```

Where “NUM-OF-“ is the standard prefix of the field name. The two suffixes in this example, “MNEMONICS” and “SAMPLES”, are then used in the singular form to describe the series of field names that follows. In the above example, if NUM-OF-MNEMONICS = 2 and MNEMONIC.n.NUM-OF-SAMPLES = 3, the actual message would contain the field names as follows:

```
NUM-OF-MNEMONICS
MNEMONIC.1.NAME
MNEMONIC.1.NUM-OF-SAMPLES
MNEMONIC.1.SAMPLE.1.TIME-STAMP
MNEMONIC.1.SAMPLE.1.RAW-VALUE
MNEMONIC.1.SAMPLE.2.TIME-STAMP
MNEMONIC.1.SAMPLE.2.RAW-VALUE
MNEMONIC.1.SAMPLE.3.TIME-STAMP
MNEMONIC.1.SAMPLE.3.RAW-VALUE
MNEMONIC.2.NAME
MNEMONIC.2.NUM-OF-SAMPLES
MNEMONIC.2.SAMPLE.1.TIME-STAMP
MNEMONIC.2.SAMPLE.1.RAW-VALUE
MNEMONIC.2.SAMPLE.2.TIME-STAMP
MNEMONIC.2.SAMPLE.2.RAW-VALUE
MNEMONIC.2.SAMPLE.3.TIME-STAMP
MNEMONIC.2.SAMPLE.3.RAW-VALUE
```

Note that “`n`” starts with “1”.

Finally, note that this could have been represented using array notation, such as MNEMONIC[1].SAMPLE[2].RAW-VALUE, but is represented throughout this document in the notation described in this section for continuity with earlier iterations of this specification.

In UML diagrams throughout this document, each series is represented as shown in the figure below.

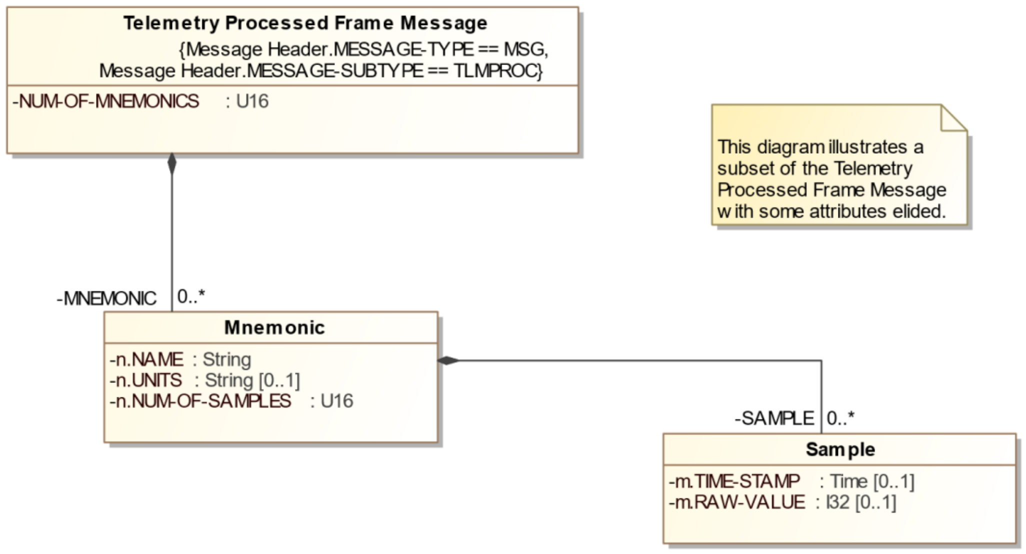

Telemetry Processed Frame Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == TLMPROC}
-NUM-OF-MNEMONICS : U16
This diagram illustrates a subset of the Telemetry Processed Frame Message with some attributes elided.
-MNEMONIC
0..*
Mnemonic
-n.NAME : String
-n.UNITS : String [0..1]
-n.NUM-OF-SAMPLES : U16
-SAMPLE
0..*
Sample
-m.TIME-STAMP : Time [0..1]
-m.RAW-VALUE : I32 [0..1]

Figure 6-3. Message Series Fields

Here, the Telemetry Processed Frame Message contains a series of MNEMONIC items, each one starting with the 'n' notation, so that the full name resolves to "MNEMONIC.n.", such as "MNEMONIC.n.NAME. Note that as shown in this diagram, each Telemetry Processed Frame Messages contains a series of 0 or more MNEMONICS, making this series optional, and in turn, each MNEMONIC optionally contains a series of SAMPLES.

6.3.4 Tracking Fields

All messages contain tracking fields that are reserved for use by the implementing software and thus any user supplied data in these fields may be overwritten. Tracking field information will be documented in the specific Message section where applicable.

6.3.5 Value

Some fields must contain specific values to be C2MS compliant. If no value is specified, the value of the Field Name is variable; however, it must conform to the specified Type. See the Type description in the next section.

6.3.6 Type

The Field Type is the data type. Cross-platform compatibility is achieved using the defined field types listed below. The intention is for the client application to not have to deal with byte-swapping or other number format changes. Type definitions are based on the Institute of Electrical and Electronics Engineers (IEEE) standards. Time field types are based on the ISO 8601 standards.

Following each message diagram, additional information is presented for each field in the message in the format of a table that adds information about the values and any important notes about the field.

Table 6-8. Field Type Definitions

Field Type	Definition	Range/Comments
Binary [Blob]	0 or more of any combination of bytes, integers, floating points, doubles, time, and strings.	Its structure may be dependent upon message type, message subtype, or application generating the message.
Boolean (1)	False/true, no/yes	[0, 1]

Field Type	Definition	Range/Comments
Character	Native single ASCII character representation	[0, 127]
F32 Float (3)	32-bit single precision floating point representation	32 bits composed of 23 bits for the fraction, 8 bits for the exponent, and 1 sign bit. (See IEEE 754)
F64 Double (3)	64-bit double precision (extended) floating point representation	64 bits composed of 52 bits for the fraction, 11 bits for the exponent, and 1 sign bit. (See IEEE 754)
GUID	Globally Unique ID. A String that uniquely identifies an item from all others	C2MS recommendation is to use the format specified in RFC 4122, which will likely be the required format in a future version of C2MS. Values of this type must comply with Subject Token String restrictions as GUIDs are sometimes used in Message Subject Elements. C2MS treats GUIDs as case sensitive for GUID lookup/matching.
Subject Token String	Any combination of uppercase or lower case alphanumeric, "-" (dash), and "_" (underscore) characters.	This field type defines fields in the format of subject name elements. Must be at least one character in length
I16 Short (3)	16-bit signed integer representation	[-2^{15}, $2^{15} - 1$] 16 bits.
I32 Long (3)	32-bit signed integer representation	[-2^{31}, $2^{31} - 1$] 32 bits.
I64 Longlong (2,3)	64-bit signed integer representation	[-2^{63}, $2^{63} - 1$] 64 bits.
String	0 or more ASCII characters	Also, see Subject Token String.
Time	String representation of time	See Table 8-2 Ordinal Date and Time Field Type Definition
U16 UShort (3)	16-bit unsigned integer representation	[0, $2^{16} - 1$] 16 bits, no sign bit
U32 ULong (3)	32-bit unsigned integer representation	[0, $2^{32} - 1$] 32 bits, no sign bit
U64 ULonglong (3)	64-bit unsigned integer representation	[0, $2^{64} - 1$] 64 bits, no sign bit
Variable	Field could be any data type	User needs to ascertain the data type of the field prior to accessing the value (e.g. with a function call)

Notes:

1. "Boolean" - The C2MS defines the *value* of the Boolean field to be 0 (zero) or 1. The *description* or meaning of the value can take various forms, such as no/yes, false/true, disabled/enabled, in-limits/out-of-limits, active/static, and so on. It is important to take into account that some programming and scripting languages (e.g., Java), schemas, commercial products, and custom software will only interpret the *value* of a Boolean field to be false/true.

2. Field types larger than 32 bits may not be available on 32-bit architecture platforms.

3. In support of both 32-bit and 64-bit architecture platforms for various equipment manufacturers, these ambiguous terms are targeted for future deprecation.

The following table describes some of the commonly used time formats. The "Time" format is the only data type specified in C2MS messages. Other formats are included for reference. Time formats generally are based on Coordinated Universal Time (UTC).

Table 6-9. Ordinal Date and Time Field Type Definition

Field Type	Definition	Range
Time	**Note: This time type can represent either an absolute or relative time.** Time in the form: [+, -] YYYY-DDD-hh:mm:ss[.ff…] Where: "+" and "-" are leading signs used for relative (duration) times. If the sign is omitted, absolute time is indicated. Relative (duration) times are given in the same format, but leading fields may be omitted rather than being set to zeros. For example, +0000-000-00:12:21 may be abbreviated +12:21 Absolute times are UTC based. Applications need to convert from the string format to native system representations.	
	"YYYY" is the year.	0000 to 9999
	"DDD" is the day number of the year as 001-365/366 in absolute time or as the number of days offset in relative time as 000-364. The following examples illustrate the extreme values of DDD for absolute time: 2031-001:09.00.00 2031-365:12.30.00 (non Leap Year) 2032-366:04.15.00 (Leap Year) and relative time: +0001-000:12.00.00 (1 yrs 12 hrs) +0000-364:08.00.00 (364 days, 8 hrs)	001 to 365/366 (absolute time) or 0-364 (relative time)
	"hh" is hours of day on a 24-hour clock.	00 to 23
	"mm" is the minutes of hour.	00 to 59
	"ss" is the seconds of minute.	00 to 60 (allowing for leap seconds)
	"ff…" is the base-ten fractional seconds. Fractional seconds are considered optional. If present, the number of digits can vary in length from 1 to 6.	0 to 999999

6.3.7 STREAM-MODE Usage Caution

Several messages in this specification support a STREAM-MODE field that is used to indicate whether the message is to be interpreted as a Real-time (RT), Replay (RPY), Simulated (SIM) or Test (TEST) message.

The use of STREAM-MODE for anything other than Real-time (RT) may be appropriate for small-scale or experimental satellite operations but is strongly discouraged in a large formal enterprise.

It is recommended to use environmental separation for performing replay, test, and simulation and to avoid flowing replay, test or simulation messages in the operational enterprise, where they might be misinterpreted by message recipients as real-world.

It is further recommended to enforce Real-time (RT) as the only valid STREAM-MODE value in such enterprises through enterprise logic.

Finally, note that when using environmental separation to perform, for example, simulation, it is preferred to mark all messages that flow in the simulation environment as Real-time (RT). In an enterprise simulation environment, marking messages with a STREAM-MODE of Simulation (SIM) likely will not exercise the system in exactly the same way as the operational system due to differences in message subject element values and probable differences in message handling.

6.4 C2MS Messages: Their Characteristics and Interactions

6.4.1 C2MS Message Type Overview

Three types or classes of messages have been defined within the C2MS. The type of the message identifies the kind of communication for which the actual message is being used. The three types of messages are as follows:

- **Message** (MSG) or generic message
- **Request** (REQ) message
- **Response** (RESP) message

Typically, the generic **Message** is published without a required or expected response (though it may cause an action when received). The **Request** message is used to request a specific action or information from a service or data provider, or product generator. The **Request** message may or may not require a reply message. If so, the **Response** message is used.

In the C2MS Subject Name Definition (see Section 6.2.3 C2MS Subject Name Standard), the type of message (**MSG**, **REQ**, and **RESP**) is specified in the **Type** element.

These three fundamental C2MS message types can be used in various combinations with one another to create an infinite number of message exchange patterns. In turn, these message exchange patterns can be used in the description of the interfaces for any number of services. A basic set of C2MS Message Exchange Patterns (MEPs) is described in Section 7 PIM – Message Exchange Patterns.

In the C2MS message type discussion that immediately follows, the terms **REQ**, **RESP**, and **MSG** will be used in reference to the type of message. The terms message, request, and response will refer to actual messages.

6.4.2 C2MS MSG Message Type Details

The MSG message type is the basic message type used to convey any kind of information. This information can be normative or critical. It can be used by itself or in combination with the other message types of REQ and RESP. When used by itself as a single message it is commonly sent for informational purposes.

Perhaps the two most common uses of the MSG type are the C2MS Log message and the Component-to-Component Transfer (C2CX) Heartbeat message. The Log message is primarily informational. It is typically published / sent by all components with no further regard or accounting by the sender. Other components will subscribe to the Log messages; some without regard to their source. No requests are made for the messages and no direct linkage exists between the publishers and subscribers.

Likewise, the C2CX Heartbeat message is pumped out periodically with no regard for its destination or subscriber. Other components will subscribe to the Heartbeat messages and use them to monitor the active status of the publishers. Again, there is no request made for the messages, no direct linkage exists between the senders and subscribers, and no interest on the part of the sender to receive a response or feedback.

A final example of the uncoupled use of the MSG message type is as a stream of data messages, most typically for a telemetry data stream. In this circumstance, one message follows another in a continuous stream of MSG messages. The stream of MSG messages can be solicited or unsolicited. An example of unsolicited MSG messages is a real-time telemetry data provider that automatically publishes CCSDS formatted frames or packets onto the network upon receiving them from a front-end provider. A solicited MSG data stream can occur following a Request/Response message exchange for a replay of stored telemetry data.

When the MSG type is used in conjunction with the other message types of REQ and RESP, it will either precede (instigate) a Request/Response message exchange, or follow the exchange such as:

- `MSG-REQ-RESP`
- `REQ-RESP-MSG`

The details of these Message Exchange Patterns are discussed in later sections.

6.4.2.1 C2MS REQ Message Type Details

The REQ message type is typically used to make a request of another component. The request could be for data, a product, service, or to take some action. The request may or may not require a response. For example, one component may request another component take some action and does not need to know immediately if the action was successful or not, as it may not care or it may discover the result by other means; or it may take a considerable amount of time for the action to be completed.

A common use of the REQ message type will be in conjunction with the Directive Request message. The Directive Request message typically asks another component (in its own language syntax, or perhaps an operations language meta-model) to perform some action. If a response is required, it is specified within the content of the Directive Request message.

6.4.2.2 C2MS RESP Message Type Details

The RESP message type is used in response to a REQ message type. It is never used in the initiation of a message exchange, only in response. For example, when sending a Directive Response message, the TYPE and SUBTYPE elements of the subject name will be RESP and DIR, respectively.

Within all response messages is a “Response Status” substructure to provide status information on the request. Depending on the status code (in the RESPONSE-STATUS field) within the structure, the response message can take on a number of different meanings. Further information on the status of the request can be found in the optional RETURN-VALUE field of the structure. This structure and the possible status codes are shown in the following table.

Table 6-10. Response Status Substructure

STRUCTURE: Response Status			
RESPONSE-STATUS	**Value**	**Description**	Identifies the status of the Request Message that was processed.
	1	Acknowledgement	
	2	Working / Keep Alive	
	3	Successful Completion	
	4	Failed Completion	

STRUCTURE: Response Status			
	5	Invalid Request	
	6	Final Message	
TIME-COMPLETED			Time application completed processing the request
RETURN-VALUE			Return value or status based on the RESPONSE-STATUS. Used to provide function call status or error code in the case of failed completion

Depending on the type of messages and the service/data the responder is providing, there may be a need for more than one Response message to be returned to the requestor.

For example, the responder may initially return a Response message with a RESPONSE-STATUS of “Working/Keep Alive”. This status indicates the message has been received and that the request is still active or being processed but is not yet complete. The responder may periodically return this same status until the processing has been completed.

At this point a Response message is returned with a status of either “Successful” or “Failed” in the RESPONSE-STATUS field. Thus, a series of Response messages may be returned to the requestor. Each return status is explained in greater detail below. Due to the uncertain amount of time required to process requests, there is no defined interval between the Request and (multiple) Response messages. Discussion of each status code follows.

Acknowledgement

- Meaning - The Request Message was received. No action has yet been taken on the Request Message.
- Sequencing - This could be the first of a series of Response messages, or be the one and only final message in the case where the message exchange pattern is Request/Acknowledgement (ACK). Only 1 ACK status would normally be returned.

Working/Keep Alive

- Meaning - The request has been received and is actively being processed.
- Sequencing - This status could initially be returned or could be the second, following an ACK status. This status could be returned a multiple number of times. It should not be the last response message returned.

Successful Completion

- Meaning - The request was valid and has been processed in a successful manner.
- Sequencing - This status could be initially returned, or it could follow either of the two statuses above. It will also be the last status returned. If initially returned, the responder was able to complete the request in a timely manner and immediately return a response. In other cases, the responder required a lengthier time period to complete the request.

Failed Completion

- Meaning - The request was valid, processing was initiated, but the responder was unable for any number of reasons to fully and successfully complete the request.

- Sequencing - This status could initially be returned (the 1st and only status), or could follow an ACK or Working/Keep Alive status. It will not follow a Successful status. It will be the last status returned.

Invalid Request

- Meaning - The request message was unable to be fully interpreted for processing. There could be missing parameters, inconsistencies, or incorrect values.

- Sequencing - This status could be the first and only one returned. It could also follow an ACK. It will be the last status returned.

Final Message

- Meaning - This is the last and final message in a series of messages. This status code has been included to provide an unmistakable indication that this is indeed the final message in a series.

- Sequencing - This status will never be the first one returned and will always follow previous Response messages, either a series of data value messages, or other Response messages.

A summary of the sequencing of the statuses is shown in the following table.

Table 6-11. Sequence of Response Status

		Sequence		
Status Code		Initial Status	Intermediate Status	Final Status
1	**Acknowledge**	Yes	No	Yes
2	**Working/Keep Alive**	Yes	Yes	No
3	**Successful Completion**	Yes	No	Yes
4	**Failed Completion**	Yes	No	Yes
5	**Invalid Request**	Yes	No	Yes
6	**Final Message**	No	No	Yes

This page intentionally left blank.

7 PIM – Message Exchange Patterns

C2MS message exchange patterns define common interactions and activities associated with creating and using C2MS messages. The following figure shows a UML use case diagram of these message exchange patterns.

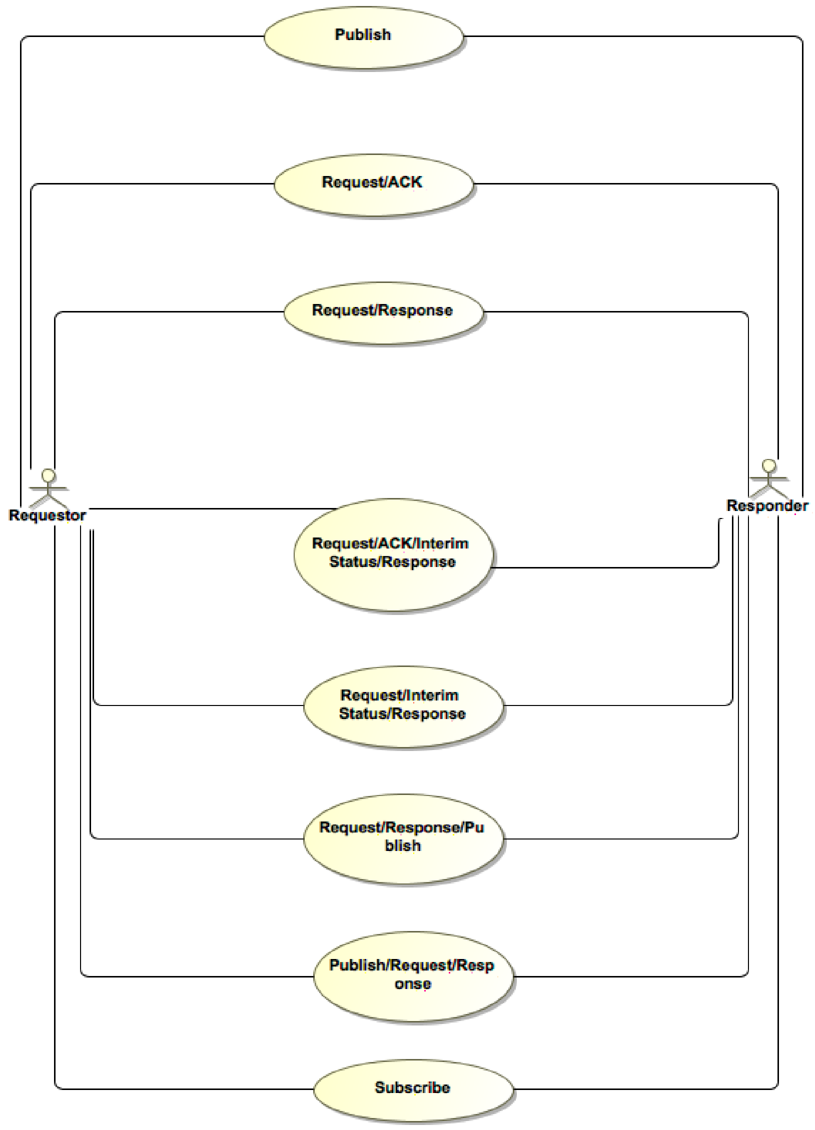


Figure 7-1. C2MS Message Exchange Patterns Diagram

Actors

Requestor – The component initiating the exchange of messages.

Responder – The component responding to the initial message from the requestor.

Message Exchange Pattern Sequence Diagrams

Section 6 described some specific ways in which the three C2MS message types of REQ, RESP, and MSG can be used. In fact, these three message types can be combined to create an endless number of message exchange patterns (MEPs). Fortunately, a limited number of MEPs can be identified to satisfy practically all interaction requirements of software applications involving service consumers and providers.

Each MEP is detailed in its own section that includes a description, a usage, and a sequence diagram depicting the interactions.

Table 7-1, Table 7-2, Table 7-3, and Table 7-4 below show the currently defined C2MS message exchange patterns. For the legend for these tables, see Table 7-5.

The following table describes the Publish message exchange pattern, the simplest. For legend, see Table 7-5 below. Note that the "#" column corresponds to the sub-sections that follow. For example, Publish (#1) is described in section 7.1.

Table 7-1. C2MS Message Exchange Pattern 1 (Publish)

Pattern	#	Description / Use	MSG Sequence Direction (Wrt Initiator) and Message Types Used	Fault Message	Examples
Publish	1	Publish a single message, for any purpose, with no follow up required	Out: MSG or REQ	None	1. Send a Log or Heartbeat message 2. Send a Directive to a component for execution; no response is necessary

The following table describes the currently defined Request / Response Theme message exchange patterns. For legend, see Table 7-5 below.

Table 7-2. C2MS Message Exchange Patterns 2 – 6 (Request / Response Theme)

Pattern	#	Description / Use	MSG Sequence Direction (Wrt Initiator) and Message Types Used	Fault Message	Examples
Request / ACK	2	Publish a message and receive an acknowledgement	Out: REQ In: RESP (ACK)	None	1. A component sends a request and needs to know if it was received. 2. One component pings other components to test their responsiveness
Request/ Response	3	For Requests that can be fulfilled with a single Response message	Out: REQ In: RESP (Status)	Response (Status)	Request a product, data, or a service from another component and receive the result in the single RESP message.
Request/ ACK/ Response	4	The requestor requires an acknowledgement to the Request, then a Response.	Out: REQ In: RESP (ACK) In: RESP (Status)	Response (Status)	The initial Request message is responded to with a Response message having a status = ACK; then the Response message (with appropriate status) will follow.
Request/ ACK / Interim Status/ Response	5	For requests that take an extended time, initially provide an ACK to the Request, then periodic status updates, and then the final response message.	Out: REQ In: RESP (ACK) In… RESP (Working)… In: RESP (Status)	Response (Status)	A request for a product is responded to with an ACK, then any number of "working" messages as the product is generated, ending with a final Response message containing the product.

Pattern	#	Description / Use	MSG Sequence Direction (Wrt Initiator) and Message Types Used	Fault Message	Examples
Request/ Interim Status/ Response	6	Identical to the Request/ ACK /Interim Status/Response pattern but with no ACK message.	Out: REQ In-in... RESP (Working ...) In: RESP (Status)	Response (Status)	A request for a product is responded to with any number of "working" messages as the product is generated, ending with a final RESP message containing the product.

The following table describes the currently defined Triad Theme Patterns message exchange patterns. For legend, see Table 7-5 below.

Table 7-3. C2MS Message Exchange Patterns 7 – 8 (Triad Theme Patterns)

Pattern	#	Description / Use	MSG Sequence Direction (Wrt Initiator) and Message Types Used	Fault Message	Examples
TRIAD 1 **Request/ Response/ Publish**	7	For requests that either take a long time, or that require a subsequent message, and no interim status updates are required. (Combination of Request/Response and Publish).	Out: REQ In: RESP(Working or Status) In: RESP or MSG	Response (Status)	A request for a product is responded to with the RESP message. Later, when the product is generated or made available, it is sent with a RESP or MSG type. The requestor does not require any interim status messages.
TRIAD 2 **Publish/ Request/ Response**	8	Send notification that Requests can be accepted. Then accept request(s) for that product/service. (Combination of Publish and Request/Response).	Out: MSG In: REQ Out: RESP (Status)	Response (Status)	Provider sends message announcing availability of product. Consumers use the Request/Response interaction pattern to then request and receive the product.

The following table describes the currently defined Subscription message exchange pattern. For legend, see Table 7-5 below.

Table 7-4. C2MS Message Exchange Pattern 9 (Subscription Pattern)

Pattern	#	Description / Use	MSG Sequence Direction (Wrt Initiator) and Message Types Used	Fault Message	Examples
Subscription • **Subscribe** • **Data stream** • **Unsubscribe**	9	Subscribe and unsubscribe to data, products, or message streams. (Combination of Request/Response, Publish and Request/Response.)	Out-In: REQ - RESP (Status) In... MSG ... Out-in: REQ - RESP (Status)	Response (Status)	Use Request/Response to subscribe to a series of messages such as a telemetry data stream or mnemonic set. The stream is subscribed to and ingested. Later, unsubscribe to the messages.

For terms and meanings, see Table 7-5 below.

Table 7-5. Legend for Table 7-1, Table 7-2, Table 7-3, and Table 7-4

Term	Meaning
Wrt	With respect to
REQ, RESP, and MSG	Are message types
"…"	Indicates any number of these messages from 0 to n
ACK, Working, Status	• RESP (ACK): indicates an ACK message in the form of a Response message with a status code of "ACK" in the RESPONSE-STATUS field. • RESP (Working): indicates an interim status message in the form of a Response message with a status code of "Working" in the RESPONSE-STATUS field. • RESP (Status): indicates a final Response message in the message exchange pattern. o For successful exchanges, the status code is either "Success" or "Final Message" in the RESPONSE-STATUS field. o For fault messages, the status code is either "Failure" or "Invalid" in the RESPONSE-STATUS field.

7.1 Publish

DESCRIPTION

The simplest message exchange pattern involves no obvious exchange with the sender. The sender publishes a message, and from the perspective of the sender, the exchange is complete. The publisher has no more interest in the message and has no need to follow up on its progress. Some other component will subscribe to the message and process it according to the subscriber's requirements.

Either of two C2MS message types can be used for this pattern; either a C2MS MSG type or REQ type message. In the case of a REQ message type, the sender may request an action but not require knowledge of the result via a Response message, which will be indicated in the Request message.

USAGE

Most typically, this pattern is seen when a software component publishes a C2MS Log message using the MSG message type. The Log message and this exchange pattern provide a simple, one-way means of information distribution.

Also, data streams of telemetry will be published using the C2MS MSG type. Again, the data is published unidirectional with subscribers ingesting the messages and processing them as programmed.

A second message type, REQ, can also be used for this one-way pattern. In this case, a component will send a Request message but has no need for follow up. No acknowledgement or response is required. An indicator is present in the Request message on whether or not a response is required.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Publish message exchange pattern.

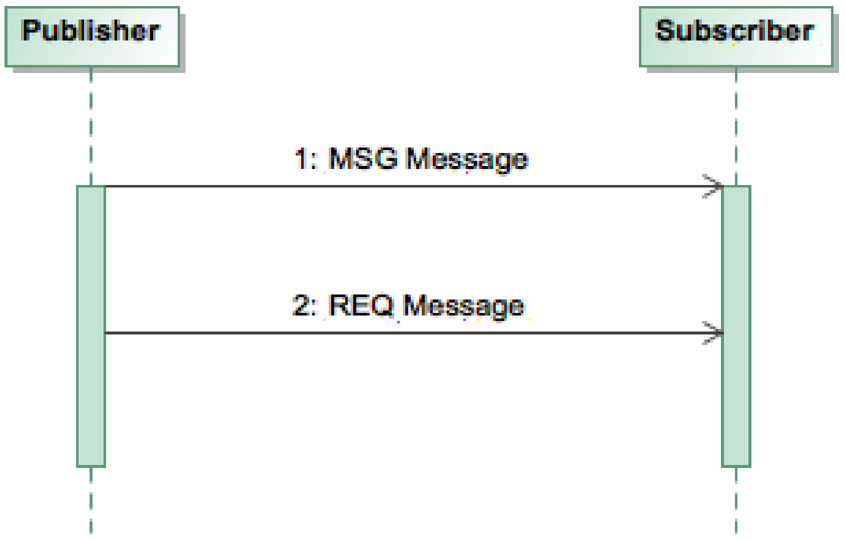


Figure 7-2. Sequence Diagram of Publish

7.2 Request/Acknowledgement

DESCRIPTION

The Request/ACK message exchange pattern consists of a published Request message and a returned Response message. The RESPONSE-STATUS field of the Response message contains a value of "Acknowledgement". No further messages will be returned to the sender.

USAGE

A component sends a Request message and needs to know if it was received. No further information is necessary. Most likely, the sender of the request will need to take subsequent action if the message was not received. Therefore, the Request/ACK pattern can provide confirmation the message was received when no other information is necessary.

Another possible usage is to "ping" other components. One component may need to take a roll call of members in its group. This could be accomplished with a Request message (e.g., Directive Request) that requires all recipients to return a Response message with the status of ACK.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Request/ACK message exchange pattern.

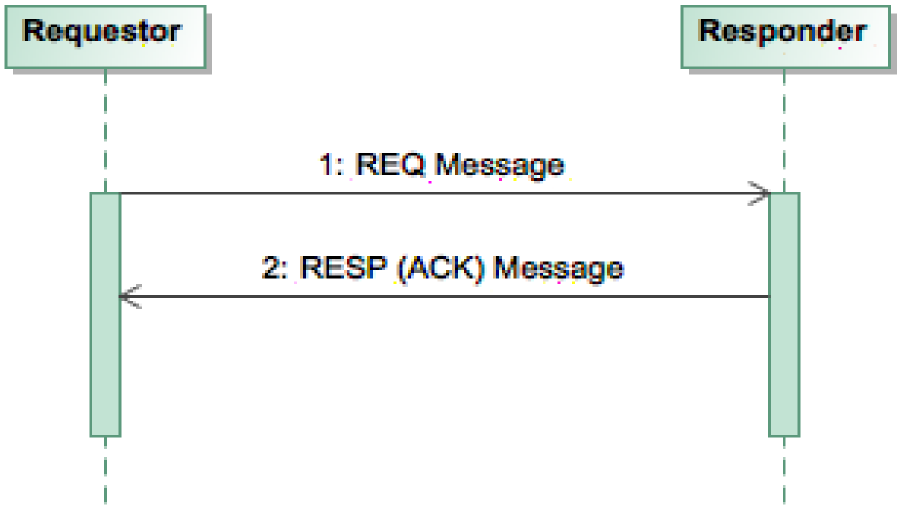


Figure 7-3. Sequence Diagram of Request/ACK

7.3 Request/Response

DESCRIPTION

The Request/Response message exchange pattern is a common one-for-one message exchange. This MEP takes the previously described Request/ACK one step further. In this case the Response message will contain an informative status code on the results of the request. See Section 6.4.2.2 C2MS RESP Message Type Details for a discussion on the possible status codes for a Response message. The Response message may also contain the resultant data and information, either within the message or via reference. Many C2MS message definitions are paired in the Request/Response fashion.

USAGE

Components that need to know the results and/or require information from a request will use the Request/Response MEP. Requests can be made for almost anything, including the following:

Telemetry and Command

- Replay Telemetry Request/Response
- Mnemonic Value Request/Response
- Archive Mnemonic Value Request/Response
- Command Request/Response

Product and Services

- Product Request/Response

Function Specific

- Directive Request/Response

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Request/Response message exchange pattern.

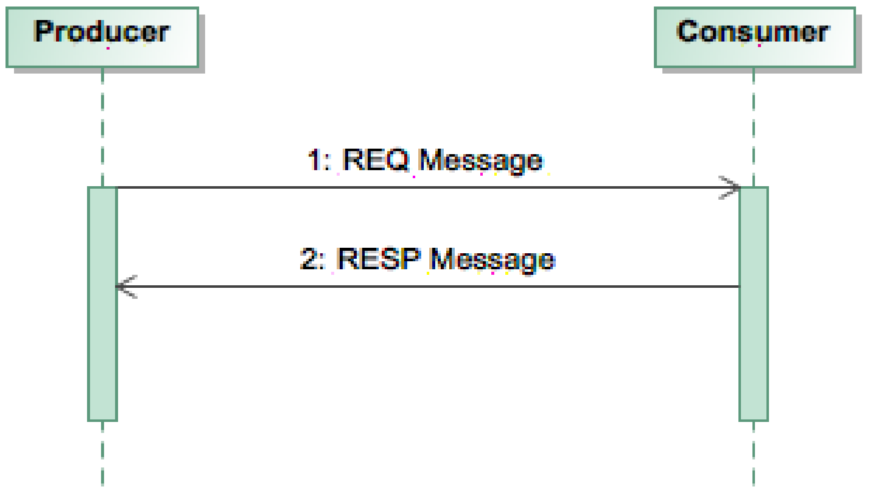


Figure 7-4. Sequence Diagram of Request/Response Message Exchange Pattern

7.4 Request/Acknowledgement/Response

DESCRIPTION

The Request/ACK/Response message exchange pattern is a combination of the Request/ACK and the Request/Response MEPs. In this case, the requestor requires confirmation that the request was received, final status information on the results of the request, and most likely information generated from processing the request.

USAGE

Usage is similar to the Request/Response MEP.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Request/ACK/Response message exchange pattern.

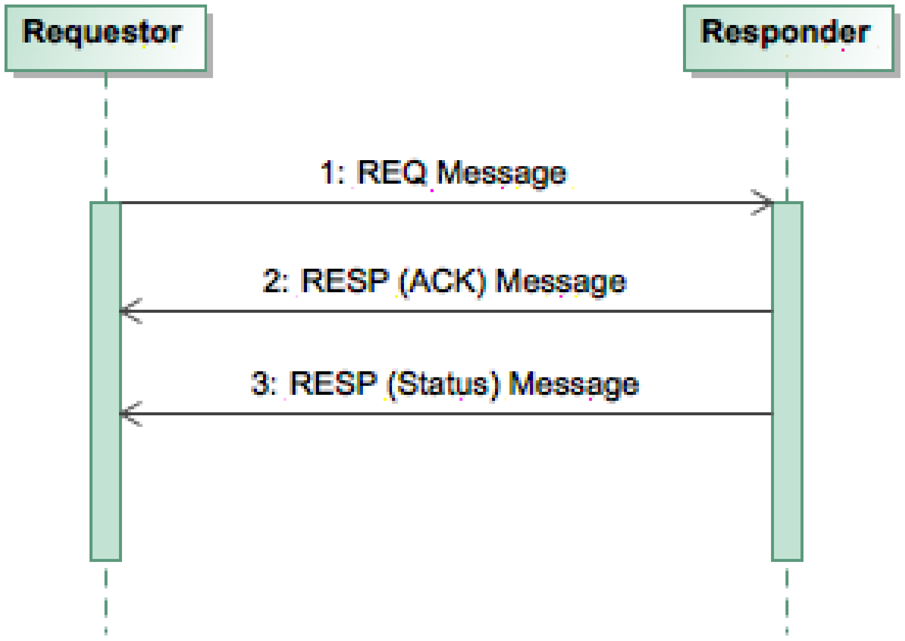


Figure 7-5. Sequence Diagram of Request/ACK/Response Message Exchange Pattern

7.5 Request/Acknowledgement/Interim Status/Response

DESCRIPTION

For some requests, an extended period of time may be required to complete the action or task. Additionally, the requestor may want to be kept abreast of the progress of such a request. In this case, the Request/ACK/Interim Status/Response message exchange pattern is appropriate. This pattern provides for the following responses:

- Initial Response message with an Acknowledgement (ACK) status
- Any number of interim Response messages with a status of "Working/Keep Alive"
- A final Response message with the status of the request and possibly information generated from processing the request.

The number and frequency of the interim Response messages are left up to the interacting components to be determined prior to execution.

For example, depending on the request and the resources required by the responder, it could take seconds, minutes, hours, or even days to complete a request. The requestor may want to be kept informed with periodic Response messages (with a status of "Working") to be assured that the request has not been lost, dropped, or forgotten.

Thus, the requestor can be kept informed over an extended period of time that a final response is forthcoming. If the interim status Response messages cease, the requestor can determine what subsequent action to take.

USAGE

A component may request a telemetry data product – for example, an archived data set or a data plot. The provider of this data product may need to first validate the request, and then request the necessary data from another data provider. This could take the form of another Request/Response MEP with another component. Once the data set has been retrieved, the data plot can be generated and finally provided back to the original requestor. In this instance, seconds may transpire while the original request ripples through a system generating other product and service requests.

A second, longer-duration example is a request made for a product that is not yet available and requires ancillary data that won't be available for some time.

For example, a request may be issued for a satellite contact schedule, tracking data, an activity plan, or for the set of available resources. These products may be generated on a scheduled, periodic basis, after the completion of a pass, or only after some other product has been generated in the future.

The provider to the original request may store or queue the request to later be acted upon when other data products become available. In the meantime, the responder will issue a "Working/Keep Alive" status in interim Response messages periodically until the request can be satisfied. If some link in the sequence chain of dependent product generation is broken – for example a necessary product is not forthcoming – then a fault Response message can be issued. In either case, whether the request can be satisfied or not, a final Response message can be issued and the requestor can determine appropriate actions.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Request/ACK/Interim Status/Response message exchange pattern.

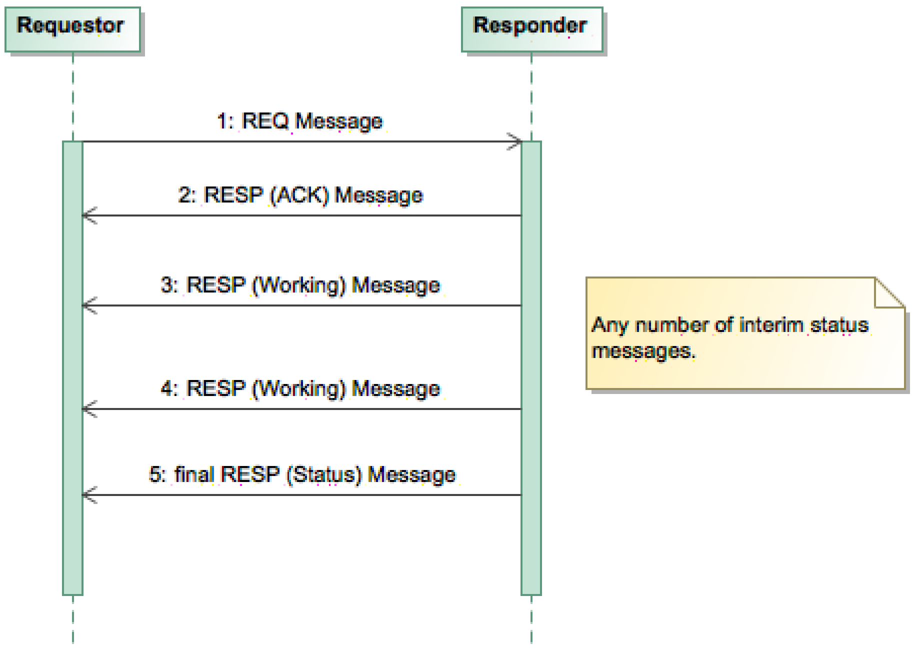

Requestor
Responder
1: REQ Message
2: RESP (ACK) Message
3: RESP (Working) Message
Any number of interim status messages.
4: RESP (Working) Message
5: final RESP (Status) Message

Figure 7-6. Sequence Diagram of Request/ACK/Interim Status/Response Message Exchange Pattern

7.6 Request/Interim Status/Response

DESCRIPTION

The Request/Interim Status/Response message exchange pattern is an abbreviated form of the Request/ACK/Interim Status/Response MEP. No ACK Response message is provided in the sequence, only interim status messages and a final Response message.

USAGE

Usage is similar to the previous Request/ACK/Interim Status/Response MEP.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Request/Interim Status/Response message exchange pattern.

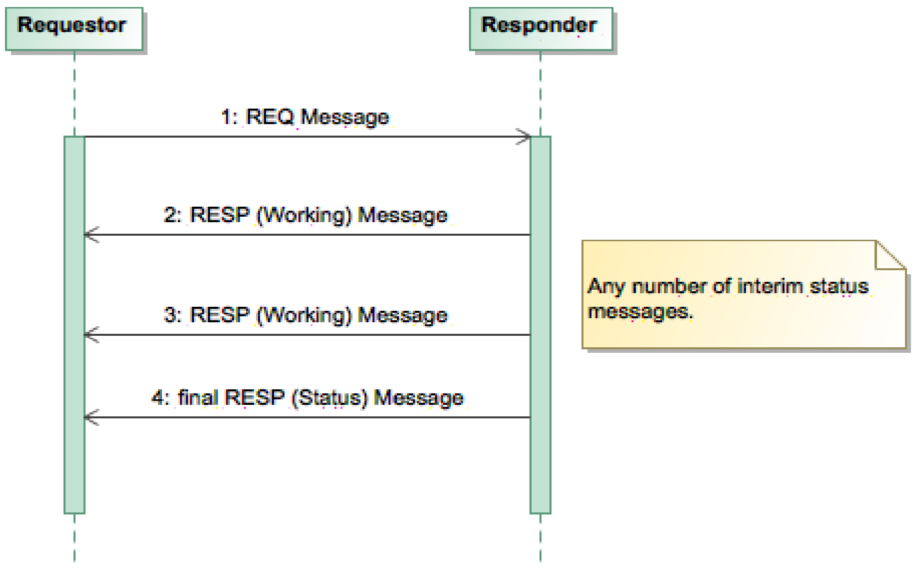

Requestor
Responder
1: REQ Message
2: RESP (Working) Message
Any number of interim status messages.
3: RESP (Working) Message
4: final RESP (Status) Message

Figure 7-7. Sequence Diagram of Request/Interim Status/Response Message Exchange Pattern

7.7 Request/Response/Publish

DESCRIPTION

The message exchange pattern of Request/Response/Publish will typically be used where requests could take a long time to fulfill, or require a subsequent message after the Request/Response interaction. They also do not require any interim status messages. The Request/Response/Publish MEP can be thought of as a combination of the Request/Response and Publish MEPs.

The Request/Response/Publish MEP will be used where the provider of a service or product can respond fairly quickly with an indication that the request can be satisfied, but cannot provide the results within the Response message itself.

If the request can be satisfied, by indicating a “Successful” or “Working” status within the Response message, the requestor knows a subsequent message will follow. The subsequent message will contain information about the results of the request.

The subsequent message could be a Log message, one of the aforementioned data messages, a Product message, or another Response message that is better suited to contain the requested information. In some cases, only one subsequent message will follow. In other cases, a stream of messages (RESP or MSG) may be required to complete the data request.

Note that this MEP differs from a subscription MEP. A subscription MEP remains open and messages will be published until the subscription is cancelled. The Request/Response/Publish MEP is a one-time request for information, data, service, or product that may take one or more messages to fulfill.

USAGE

A user requests a data product from a product provider. The provider is able to satisfy the request and this is conveyed through the Product Request/Product Response interaction. When the requested product is generated or available, it will be published with the Product Message.

A set of historical mnemonic values is requested from a data provider. The Archive Mnemonic Value Request message allows a number of delivery options, including the option to receive archived mnemonic data as a stream of messages. The subscriber may prefer to receive and process archived data in the same manner as real-time data, i.e., as a stream of messages.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Request/Response/Publish message exchange pattern.

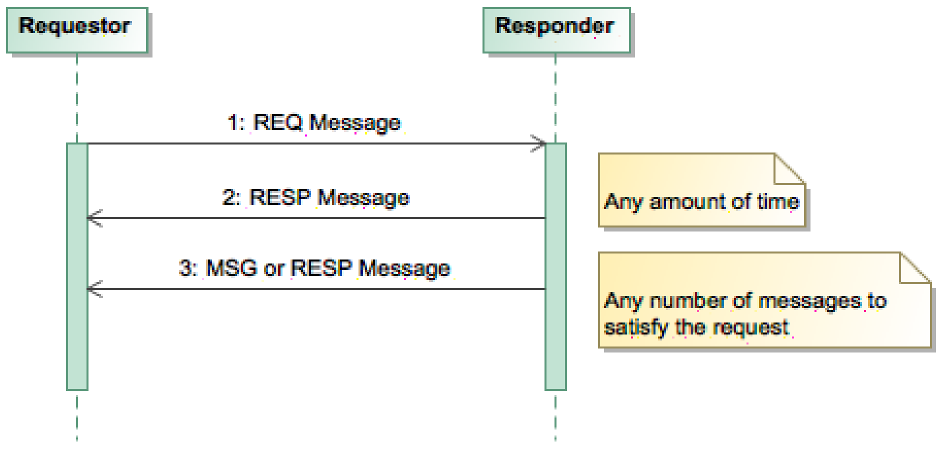


Figure 7-8. Sequence Diagram of Triad 1: Request/Response/Publish Message Exchange Pattern

7.8 Publish/Request/Response

DESCRIPTION

The Publish/Request/Response message exchange pattern is a combination of a Publish MEP and a Request/Response MEP. The initial Publish of a MSG message will instigate a follow up Request/Response interaction. This Publish/Request/Response MEP will typically be initiated by a service or product provider. The data or product provider will publish a MSG message to notify interested subscribers that a product, service or data is now available. Subscribers to the MSG message can then request and receive the product through the Request/Response interaction.

USAGE

A schedule product producer has just completed the compilation of an operational schedule for the next day. The producer issues a Log message that contains information on the type of product and how/where to acquire it. The schedule execution component has previously subscribed for this particular message.

When the message is received and parsed, the schedule execution component issues a Product Request message to the producer for the operational schedule product. The product is received within the Response message and readied for the next day's operations.

Note that an alternate methodology for this interaction could be accomplished with a Subscription MEP discussed in Section 7.9. In this case, the producer of the schedule product would provide a subscription service. Parties interested in knowing when a schedule has been generated and is available could subscribe with the producer. The producer would automatically provide the product when available for any subscriber.

A second scenario could involve a mnemonic data provider. In this example, the data provider has just completed "scrubbing" the data (merging, gap filling, and correcting) and publishes a message to that effect. Users interested in clean data can then initiate the Request/Response interaction for the scrubbed archived mnemonic data.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the Publish/Request/Response message exchange pattern.

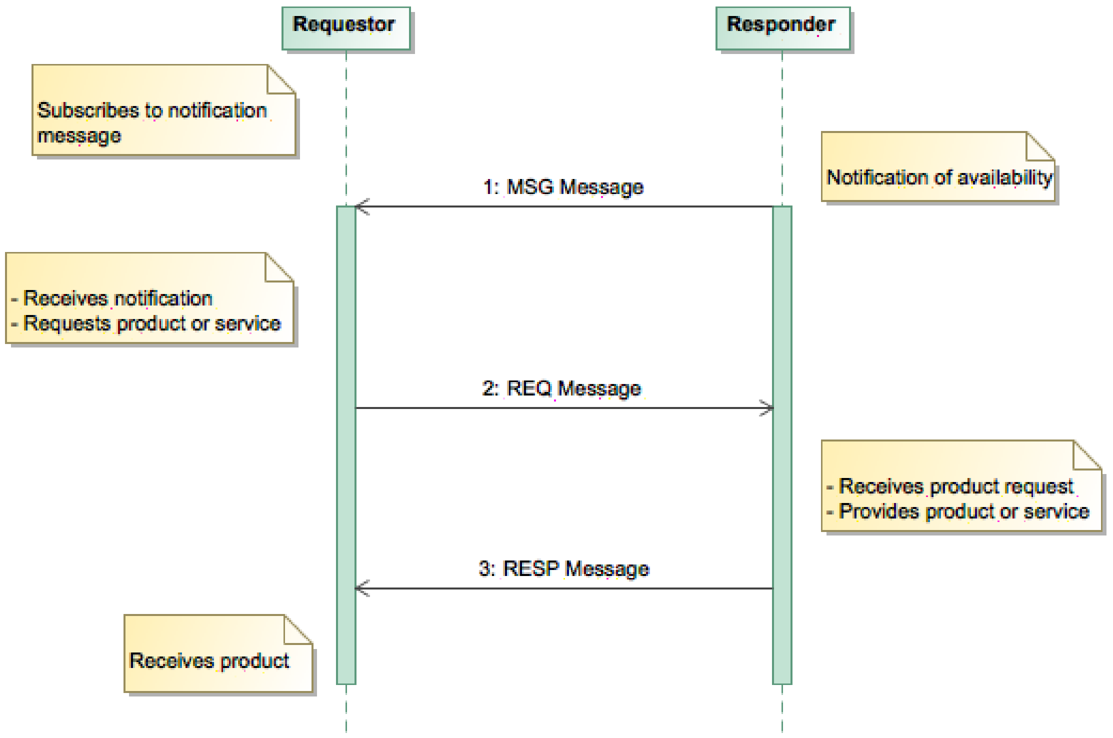

Requestor
Responder
Subscribes to notification message
Notification of availability
1: MSG Message
- Receives notification
- Requests product or service
2: REQ Message
- Receives product request
- Provides product or service
3: RESP Message
Receives product

Figure 7-9. Sequence Diagram of Triad 2: Publish/Request/Response Message Exchange Pattern

7.9 Subscription

DESCRIPTION

The subscription message exchange pattern is used to provide a continuous data or product delivery service. The data or product can take the form of one or a series of messages. The steps for a subscription will typically be:

Request/Response – REQ/RESP message pairs are used to request the data or product that is desired

Publish - the MSG type messages are used by the publisher to distribute the specified data

Request/Response - REQ/RESP message pairs are used to unsubscribe from the data

The subscription will remain active until the subscriber cancels it. There is no restriction on the timing of the MSG messages, nor on the number of messages. The subscription could result in one MSG message or a set of MSG messages. Additionally, the set of MSG messages could be periodically repeated.

The subscriber is free to cancel the subscription at any time; however, the provider can also terminate the subscription for its own reasons.

USAGE

A subscriber requests a specific set of real-time mnemonic data values from a data provider. The provider will respond with a Response message indicating success or failure of the request. If the request was successful, mnemonic data values will be published. The subscription will remain open and on the next pass the requestor will again receive the specified real-time mnemonic data. The data will continue to be published for each pass until the requestor unsubscribes from the data. The subscriber can request the termination of the data values at any time.

In other scenarios, the subscriber may subscribe to previously known or predefined data sets already being published, rather than request specific data sets. Or, if enough information about a data set is made known and available, a consumer may simply read or ingest the data stream messages without even subscribing. There can be multiple subscribers for the data sets and products.

SEQUENCE DIAGRAM

The following figure shows a UML sequence diagram for the subscription message exchange pattern.

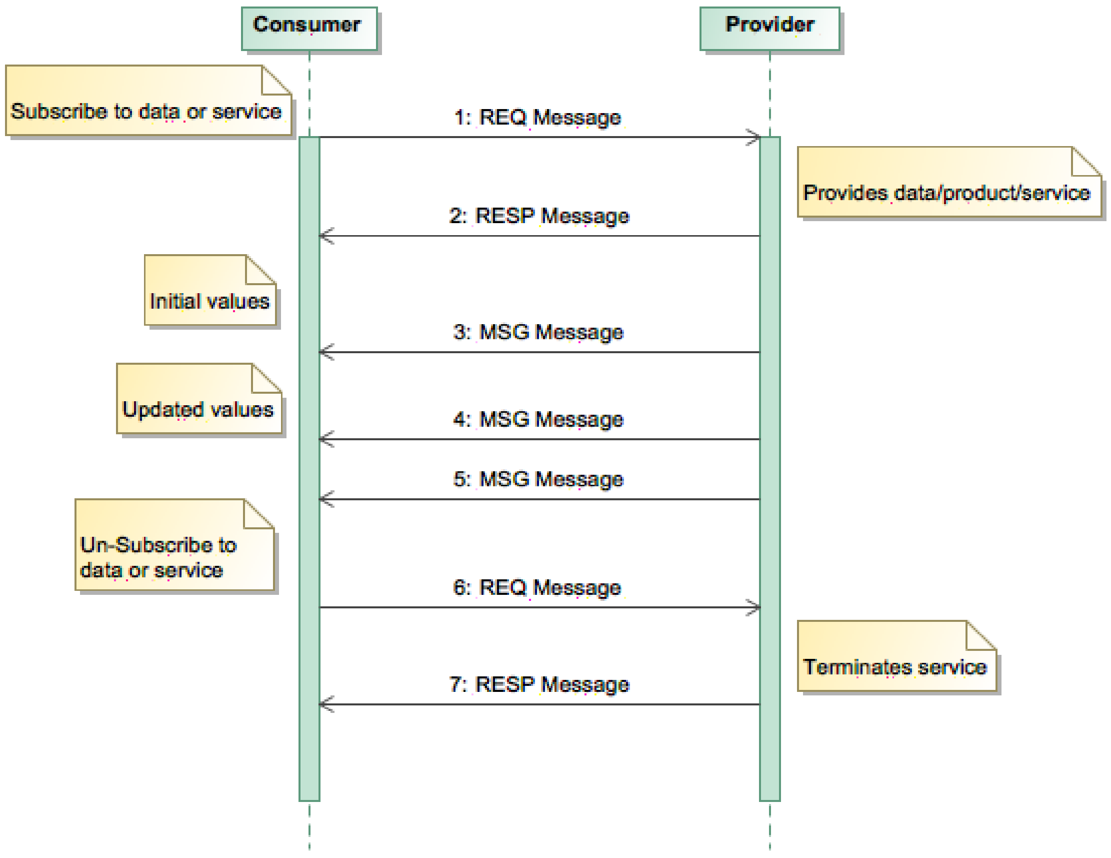


Figure 7-10. Sequence Diagram of Subscription Message Exchange Pattern

8 PIM – Message Definitions

This C2 Message Specification Document contains the standard set of defined messages. Each standard message is composed of a C2MS Message Header section and a Message Contents section. Additionally, each message defines the subject names associated with the message.

These messages are described in detail in the following sections. For the diagrams of the messages, UML classes are used. The field names, field types, and field values are also shown or each message.

8.1 C2MS Message Envelope

C2MS Messages may be contained within a C2MS Message Envelope. This Envelope includes fields used for tracking, meta information about the user sending the message, encryption information about the message and a digital signature or message authentication code.

The purpose of the C2MS Message Envelope is to separate fields out from the message that are not part of C2, but related to message handling. In earlier versions of C2MS and its predecessors, these fields were part of the message itself. At this time, all the fields that exist in the C2MS Message related to these message-handling aspects are preserved within the C2MS Messages for backward compatibility. This means that many fields exist both in the C2MS Message and in the C2MS Message Envelope. This has the benefit of making the Envelope an entirely optional construct. However, in a future release, C2MS will make C2MS Message Envelope required for sending messages and may deprecate and/or remove overlapping fields from the C2MS Messages. With this in mind, it is advised to begin using C2MS Message Envelope at the earliest opportunity in order to aid future migration.

8.1.1 Message Security Support in Message Envelope

The Message Envelope includes fields that allow mission enterprises to utilize their own established and already-approved policies and procedures to handle data in a secure manner outside of C2MS and the transport layer implementation (PSM). In this way, a sender and receiver(s) are not dependent upon the third-party capabilities of the transport layer to protect their data. Additionally, this approach ensures that the message can be protected from access and/or modification by non-authorized entities throughout the entire transport of the message from the sender to each authorized recipient.

For example, a message producer may decide to encrypt a C2MS Message using mission policies and procedures and package the encrypted message into a Message Envelope along with optional information regarding how the message was encrypted (such as the possible inclusion of a SEC-ENCRYPT-KEY-ID). The message producer then sends the C2MS Message Envelope via the transport layer implementation (PSM) to one or more recipients. Any recipient of the Message Envelope can read the information in the provided fields to assist in performance of message decryption, if authorized by the same mission policies and procedures. In this example, C2MS and the transport layer do not encrypt or decrypt the message but are used to deliver the encrypted message from a sender to recipients.

These Message Envelope security fields fall into three areas: message encryption, message authentication, and user credentials. Each area provides a distinct security aspect of messages.

8.1.1.1 Message Encryption

The C2MS Message Envelope contains a C2MS Message. The contained Message may be in the clear or may be encrypted. As described above, the particular manner and method for encryption are independent of C2MS and assumed to be a matter for the C2MS Message sender and receiver. However, the C2MS Message Envelope contains a field called MESSAGE-SECURITY.SEC-ENCRYPT-KEY-ID that may be used to convey information about what encryption key is used to encrypt/decrypt the message. In this way, the sender can convey this information to the receiver. MESSAGE-SECURITY.SEC-ENCRYPT-KEY-ID is not interpreted by C2MS. It may, for example, contain the ID of a key, either symmetric or asymmetric that should be used to decrypt the message. This is entirely at the discretion of the sender/receiver.

C2MS provides the facility for C2 Systems to supply an encrypted C2MS Message but retain tracking information within the unencrypted C2MS Message Envelope, so that the message can be properly routed. With this approach, a sender can send a C2MS Message to a recipient and be assured that only that authorized recipient can decrypt the message; in this case, the C2MS Message remains encrypted throughout the entirety of message handling process.

Note that this encryption of the C2MS Message is apart from and not related to connection encryption (TLS) between the sending/receiving component and the transport layer. While TLS is good for protecting the connection, it does nothing to encrypt the message as it traverses a message bus or once it has been delivered to a recipient. Encryption of the C2MS Message prior to transport is a way to guarantee that only valid recipients can decrypt the message to view its contents.

8.1.1.2 Message Authentication

Because the Message Envelope is separate from the contained C2MS Message, the Envelope may be used to convey a digital signature or message authentication code (MAC) over the entirety of the enclosed C2MS Message. This information may be contained within the message authentication fields of the Message Envelope. The purpose of this is to provide proof of the message origin and assurance that the message has not been modified since it originated.

As with encryption, the particular manner and method for generating or evaluating these message authentication fields are left to the C2 System sender and receiver(s), though two common and expected forms are a Digital Signature (using asymmetric keys) and Message Authentication Code, or MAC (using symmetric keys).

C2MS itself does not examine or attempt to use the message authentication fields but conveys the information for the benefit and use of the sender/receiver to authenticate the C2MS Message.

Finally, as a note, a Digital Signature or MAC may be used on a C2MS Message whether or not that message has been encrypted. The two functions and purposes are separate from each other.

8.1.1.3 Sender Credentials Tracking in Message Envelope

The USER-NAME field from the C2MS Message Header has been brought into the C2MS Message Envelope for compatibility reasons. This field may be used to hold the "account name or owner of the account that started the component - reserved for use by implementation (PSM). API-generated tracking field." Note, though, that this is for informational purposes and should not be confused with providing user-based authentication or authorization.

Instead, the C2MS Message Envelope adds new fields: MESSAGE-SECURITY.SENDER-IDENTITY and MESSAGE-SECURITY.SENDER-ACCESS, which may be used for more secure user-based credentials. Note that SENDER, in this case could be human (user) or non-human (component). It could also refer to a group rather than an individual. This is all according to mission policy.

SENDER-IDENTITY is a Binary field that may be used to hold token-like data used by the mission enterprise policies and procedures for passing authentication credentials, usually the result from an authentication service, rather than the identity to be authenticated. This could be, for example, a SAML Token or an OpenID Connect ID Token.

Similarly, SENDER-ACCESS is a Binary field that may be used to hold token-like data used by the mission enterprise policies and procedures for passing authorization credentials. This could be, for example, an OAuth2 Access Token.

Each of these fields has a corresponding type indicator to convey within the enterprise the type of data contained in the field: SENDER-IDENTITY-TYPE and SENDER-ACCESS-TYPE. For example, these fields would convey if the corresponding field contains a SAML Token or OAuth2 Access Token.

Together, these fields allow the mission enterprise to utilize its own established policies and procedures to convey authentication and authorization information for a C2MS Message within the C2MS Message Envelope. Any individual mission enterprise may choose to use neither, either or both IDENTITY and ACCESS fields. These fields are not evaluated by C2MS. Note that it is often the case and may be required according to mission polices and procedures to sign and/or encrypt the SENDER-IDENTITY and SENDER-ACCESS fields. In practice these fields are often represented as String types, but because they may be encrypted and because C2MS does not proscribe any particular implementation for mission use of identity or authorization management, C2MS employs Binary types for these fields.

8.1.2 Message Envelope Subject

The C2MS Message Envelope does not define any particular Subject. It is assumed that the Message Subject is that of the contained C2MS Message itself and should be established by the sender at the time of creating the message and inserting it into the message envelope.

8.1.3 Message Envelope Contents

The C2MS Message Envelope contains fields as described in the figure and table below.

The following figure shows a UML object diagram of the of the Message Envelope with its required, optional, and tracking fields.

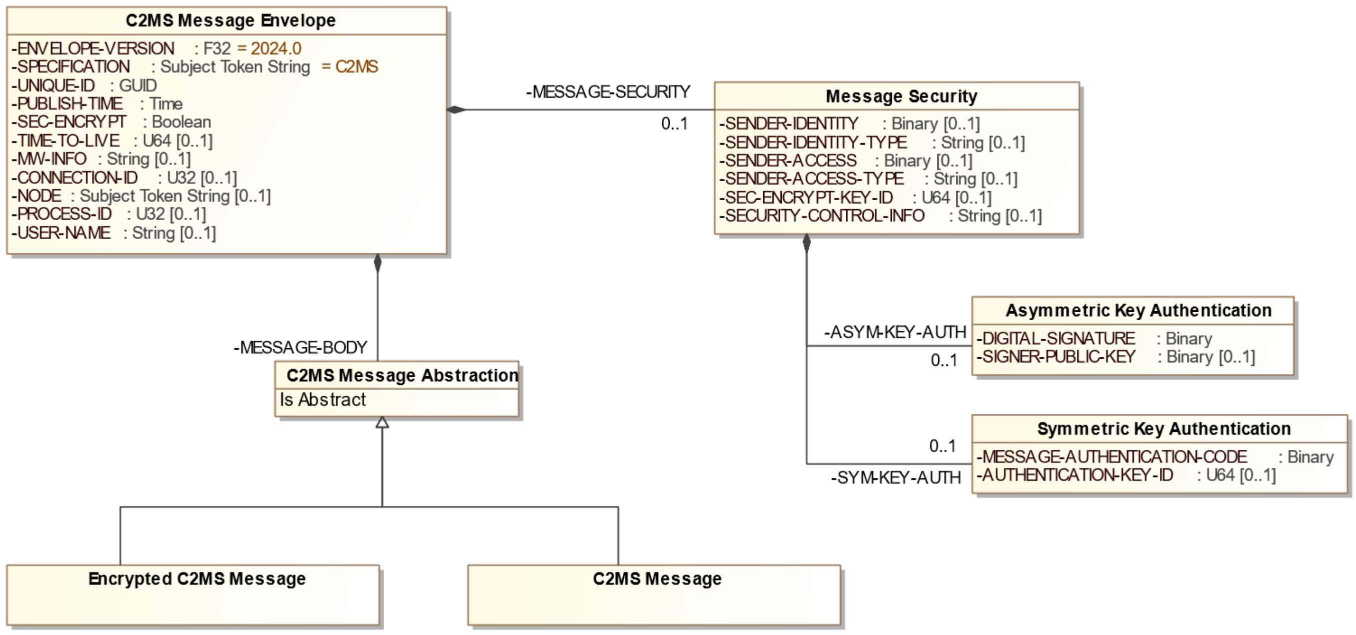

C2MS Message Envelope
-ENVELOPE-VERSION : F32 = 2024.0
-SPECIFICATION : Subject Token String = C2MS
-UNIQUE-ID : GUID
-PUBLISH-TIME : Time
-SEC-ENCRYPT : Boolean
-TIME-TO-LIVE : U64 [0..1]
-MW-INFO : String [0..1]
-CONNECTION-ID : U32 [0..1]
-NODE : Subject Token String [0..1]
-PROCESS-ID : U32 [0..1]
-USER-NAME : String [0..1]
-MESSAGE-SECURITY
0..1
Message Security
-SENDER-IDENTITY : Binary [0..1]
-SENDER-IDENTITY-TYPE : String [0..1]
-SENDER-ACCESS : Binary [0..1]
-SENDER-ACCESS-TYPE : String [0..1]
-SEC-ENCRYPT-KEY-ID : U64 [0..1]
-SECURITY-CONTROL-INFO : String [0..1]
Asymmetric Key Authentication
-DIGITAL-SIGNATURE : Binary
-SIGNER-PUBLIC-KEY : Binary [0..1]
-ASYM-KEY-AUTH
0..1
-MESSAGE-BODY
C2MS Message Abstraction
Is Abstract
0..1
-SYM-KEY-AUTH
Symmetric Key Authentication
-MESSAGE-AUTHENTICATION-CODE : Binary
-AUTHENTICATION-KEY-ID : U64 [0..1]
Encrypted C2MS Message
C2MS Message

Figure 8-1. Message Envelope Diagram

The following table describes additional field names, values, and notes for the Message Envelope.

Table 8-1. Message Envelope Additional Information

Field Name	Type	Presence	Value	Description
HEADER-VERSION	F32	R	2024	Version number for this envelope description
SPECIFICATION	Subject Token String	R	C2MS	Name of Specification Document used to define this header
UNIQUE-ID	GUID	O		Globally unique ID – reserved for use by implementation (PSM)
PUBLISH-TIME	Time	O		Time the message was published - reserved for use by implementation (PSM)
SEC-ENCRYPT	Boolean	R		Indicates whether the MESSAGE-BODY is encrypted
TIME-TO-LIVE	U64	O	In Seconds	Duration in seconds starting with the PUBLISH-TIME that it is valid for the message to be delivered. Note, however, that this is considered a suggestion. If not specified the transport layer implementation (PSM) should use its default duration and if a TIME-TO-LIVE is specified here, the implementation (PSM) may choose how to and whether to honor it based on policy. It is assumed that the polices of the transport layer implementation (PSM) will be understood by the message sender. Note that after a message has expired, the implementation (PSM) should not deliver it to any additional recipients and should scramble the security information (keys, key IDs, and signatures).
MW-INFO	String	O		Container for information on the underlying middleware - reserved for use by implementation (PSM)
CONNECTION-ID	U32	O		Unique ID for each connection per process. Reserved for use by implementation (PSM)
NODE	String	O		Actual device (host) generating the message. Reserved for use by implementation (PSM)
PROCESS-ID	U32	O		Application ID for onboard events or Process ID for ground events - reserved for use by implementation (PSM)
USER-NAME	String	O		Account name or owner of the account that started the component - reserved for use by implementation (PSM)
MESSAGE-SECURITY.SENDER-IDENTITY	Binary	O		May be used to hold token-like data used by the mission enterprise policies and procedures for passing authentication credentials, usually the result from an authentication service, rather than the identity to be authenticated. This could be, for example, a SAML Token or an OpenID Connect ID Token.
MESSAGE-SECURITY.SENDER-IDENTITY-TYPE	String	O		A discreet string defined by the mission enterprise to convey the type of SENDER-IDENTITY used, such as a defined string indicating that a SAML token is held in the SENDER-IDENTITY field.
MESSAGE-SECURITY.SENDER-ACCESS	Binary	O		May be used to hold token-like data used by the mission enterprise policies and procedures for passing authorization credentials. This could be, for example, an OAuth2 Access Token.
MESSAGE-SECURITY.SENDER-ACCESS-TYPE	String	O		A discreet string defined by the mission enterprise to convey the type of SENDER-ACCESS used, such as a defined string indicating that an OAuth2 Access Token is held in the SENDER-ACCESS field.

Field Name	Type	Presence	Value	Description
MESSAGE-SECURITY.SEC-ENCRYPT-KEY-ID	U64	O		An ID used to look up the encryption or decryption key from an external key storage service, such as a Key Management Service, as defined by the mission enterprise.
MESSAGE-SECURITY.SECURITY-CONTROL-INFO	String	O		The mission enterprise may define and pass string values as a way to communicate expected processing of security information. For example, how a signer's public key is to be found. These are to be non-executable discreet reference string values defined outside of C2MS.
MESSAGE-SECURITY.ASYM-KEY-AUTH.DIGITAL-SIGNATURE	Binary	O		The digital signature of the MESSAGE-BODY generated before the message envelope was sent.
MESSAGE-SECURITY.ASYM-KEY-AUTH.SIGNER-PUBLIC-KEY	Binary	O		The public key of the entity that digitally signed the MESSAGE-BODY. Note that this is not required to accompany a digital signature, but may, if desired under mission enterprise policies.
MESSAGE-SECURITY.SYM-KEY-AUTH.MESSAGE-AUTHENTICATION-CODE	Binary	O		The Message Authentication Code, or MAC (using symmetric keys).
MESSAGE-SECURITY.SYM-KEY-AUTH.AUTHENTICATION-KEY-ID	U64	O		An ID used to look up the symmetric key used in generating the MAC from an external key storage service, such as a Key Management Service, as defined by the mission enterprise.
MESSAGE-BODY	C2MS Message Abstraction	R		The contained C2MS Message. This may be an in-the-clear message (type C2MS Message) or an encrypted message (type Encrypted C2MS Message).

8.2 C2MS Message Header

The C2MS Message class is a super-class for all C2MS messages and any extensions. All C2MS Messages comprise a Message Header along with message-defined fields. Therefore, all fields in the Message Header appear in all C2MS messages. However, the values of the Message Header fields may vary between messages. The specifics of the Message Header field values for each message are included in its corresponding message section.

The following figure shows a UML object diagram of the of the Message Header with its required, optional, and tracking fields.

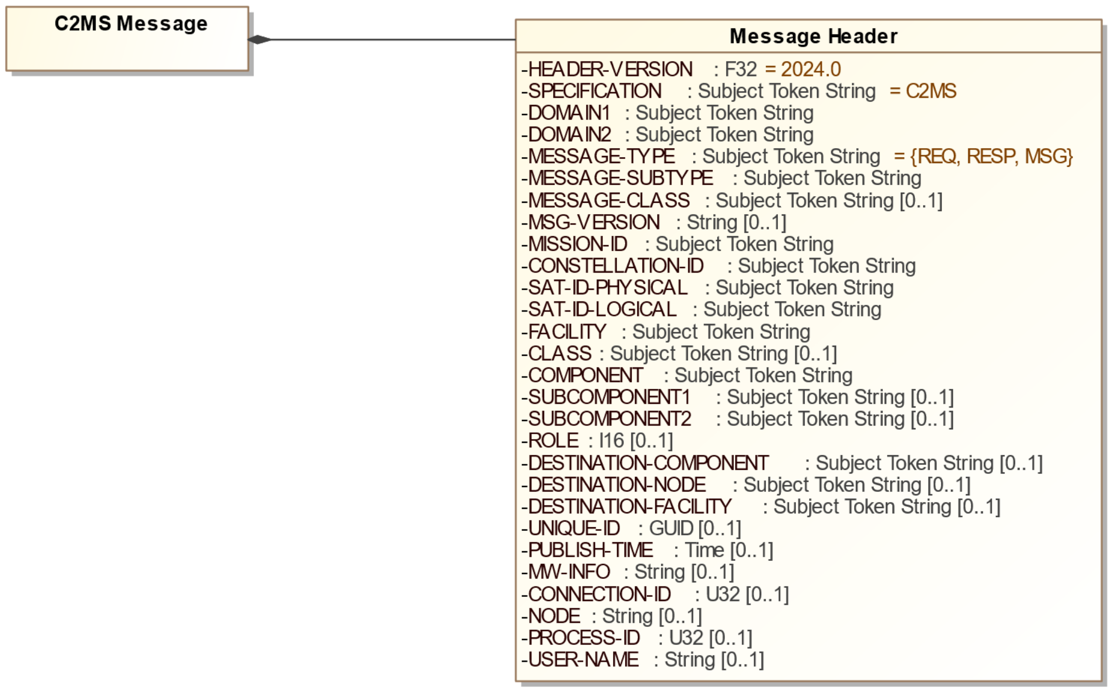

C2MS Message
Message Header
-HEADER-VERSION : F32 = 2024.0
-SPECIFICATION : Subject Token String = C2MS
-DOMAIN1 : Subject Token String
-DOMAIN2 : Subject Token String
-MESSAGE-TYPE : Subject Token String = {REQ, RESP, MSG}
-MESSAGE-SUBTYPE : Subject Token String
-MESSAGE-CLASS : Subject Token String [0..1]
-MSG-VERSION : String [0..1]
-MISSION-ID : Subject Token String
-CONSTELLATION-ID : Subject Token String
-SAT-ID-PHYSICAL : Subject Token String
-SAT-ID-LOGICAL : Subject Token String
-FACILITY : Subject Token String
-CLASS : Subject Token String [0..1]
-COMPONENT : Subject Token String
-SUBCOMPONENT1 : Subject Token String [0..1]
-SUBCOMPONENT2 : Subject Token String [0..1]
-ROLE : I16 [0..1]
-DESTINATION-COMPONENT : Subject Token String [0..1]
-DESTINATION-NODE : Subject Token String [0..1]
-DESTINATION-FACILITY : Subject Token String [0..1]
-UNIQUE-ID : GUID [0..1]
-PUBLISH-TIME : Time [0..1]
-MW-INFO : String [0..1]
-CONNECTION-ID : U32 [0..1]
-NODE : String [0..1]
-PROCESS-ID : U32 [0..1]
-USER-NAME : String [0..1]

Figure 8-2. Message Header Diagram

The following table describes additional field names, values, and notes for the Message Header.

Table 8-2. Message Header Additional Information

Field Name	Type	Presence	Value	Description
HEADER-VERSION	F32	R	2024	Version number for this message description
SPECIFICATION	Subject Token String	R	C2MS	Name of Specification Document used to define this header
DOMAIN1	Subject Token String	R		Additional field to allow for more filtering
DOMAIN2	Subject Token String	R		Additional field to allow for more filtering
MESSAGE-TYPE	Subject Token String	R	**Value** / **Description**: REQ – Request; RESP – Response; MSG – Message	Message type identifier: REQ, RESP, or MSG
MESSAGE-SUBTYPE	Subject Token String	R	See Table 6-6 Descriptions of the Message Elements of the C2MS Subject Name	Unique message subtype identifier, fixed for C2MS Standard Messages
MESSAGE-CLASS	Subject Token String	O		Generic field for missions to classify their message set to aid message disposition.
MSG-VERSION	String	O		Version information for the message

Field Name	Type	Presence	Value	Description
MISSION-ID	Subject Token String	R		Unique mission name, e.g., MOONMAP, SUNSCAN, TEMPTRACK, etc.
CONSTELLATION-ID	Subject Token String	R		Used for constellations or satellite groupings
SAT-ID-PHYSICAL	Subject Token String	R		An ID for the satellite that is fixed for its mission life
SAT-ID-LOGICAL	Subject Token String	R		An ID for a satellite or group of satellites that can change during its mission life (ex., a positional reference)
FACILITY	Subject Token String	R		A physical source (i.e. ACS Lab, CandDH String, etc.) generating the message, e.g., spacecraft, remote site, etc.
CLASS	Subject Token String	O		See Table A-1, Software Class and Subclass Categories.
COMPONENT	Subject Token String	R		Name of software application, ex. APP1, CLIENT2, TAC3
SUBCOMPONENT1	Subject Token String	O		First subsystem level within the component that produced the message
SUBCOMPONENT2	Subject Token String	O		Second subsystem level within the component that produced the message
ROLE	I16	O	**Value** – **Description** 1 – Primary, Master 2 – Secondary, Backup 3 – Tertiary … – Spare, …	Role the component is assigned in the configuration (Primary, Backup, Hot Backup, Secondary, Spare …). Roles are dependent on the operational concepts being employed in the configuration.
DESTINATION-COMPONENT	Subject Token String	O		Intended component recipient of the message (if any).
DESTINATION-NODE	Subject Token String	O		Intended node recipient of the message (if any).
DESTINATION-FACILITY	Subject Token String	O		Intended facility recipient of the message (if any).
UNIQUE-ID	GUID	O		**Tracking Field -** Globally unique ID – reserved for use by implementation (PSM)
PUBLISH-TIME	Time	O		**Tracking Field -** Time the message was published - reserved for use by implementation (PSM)
MW-INFO	String	O		**Tracking Field -** Container for information on the underlying middleware - reserved for use by implementation (PSM)
CONNECTION-ID	U32	O		**Tracking Field -** Unique ID for each connection per process. Reserved for use by implementation (PSM)
NODE	String	O		**Tracking Field -** Actual device (host) generating the message. Reserved for use by implementation (PSM)

Field Name	Type	Presence	Value	Description
PROCESS-ID	U32	O		**Tracking Field -** Application ID for onboard events or Process ID for ground events - reserved for use by implementation (PSM)
USER-NAME	String	O		**Tracking Field -** Account name or owner of the account that started the component - reserved for use by implementation (PSM)

8.2.1 Fields for Subject Names

SPECIFICATION, DOMAIN1, DOMAIN2, MISSION-ID, CONSTELLATION-ID, SAT-ID-PHYSICAL, SAT-ID-LOGICAL, MESSAGE-TYPE, MESSAGE-SUBTYPE

As discussed in Section 6.2.3 C2MS Subject Name Standard, the first set of elements of the C2MS subject are fixed. Please refer to this section for important information on the format of subject names and their content. All of the elements of the C2MS Subject Name can be made up from required fields in the C2MS Message Header. Recall that the definition of the subject follows the following format:

```
SPECIFICATION.DOMAIN1.DOMAIN2.MISSION.CONST.SAT.MSGTYPE.MSGSUBTYPE.ME1.ME2.ME
3…
```

The following table shows how fields from the C2MS Message Header can be directly used as elements of a C2MS Subject Name.

It is important to remember that a C2MS Subject Name is text that is in Subject Token String format, so if a field's value is extracted from the Message Header and used in the subject name, it must be in Subject Token String format.

Table 8-3. Mapping of Message Header Fields to the C2MS Subject Name

Subject Name Element	Possible Fields Used from the Message Header Field
Specification	SPECIFICATION
Domain1	DOMAIN1
Domain2	DOMAIN2
Mission	MISSION-ID
Const	CONSTELLATION-ID
SAT	SAT-ID-PHYSICAL or SAT-ID-LOGICAL
Type	MESSAGE-TYPE
Subtype	MESSAGE-SUBTYPE

8.2.2 Fields for Message Uniqueness

UNIQUE-ID, PUBLISH-TIME

The UNIQUE-ID is a globally unique ID reserved for use by the implementing software. It will make each message uniquely identifiable from all others. The implementing software likewise may supply the PUBLISH-TIME at the time of publication of the message. These fields are valuable for identification and tracking purposes.

8.2.3 Middleware Tracking Information

MW-INFO, CONNECTION-ID

These fields are reserved for use by the implementing software. The implementing software may fill in these Message Header fields prior to sending the message. They serve to identify the connection and/or path of the message with the underlying middleware.

8.2.4 Component Information – Location and ID

FACILITY, NODE, PROCESS-ID

These fields refer to physical locations, equipment, or identifiers. For example, facility might refer to a city, building, LAN, satellite control center, room, or whatever is used to uniquely identify the physical location. NODE will normally be used to identify a single piece of equipment - commonly a computer. It could also refer to a larger entity such as a satellite tracking station. Typically, the node is found within the facility. Another example usage is the computers (nodes) found on a LAN (facility).

PROCESS-ID is the unique ID of an executing task or process on a NODE and is usually assigned by the host operating system. The combination of these three fields serves to uniquely identify the executing software process within an enterprise. The implementing software will optionally supply the NODE and PROCESS-ID information on which the process is executing in the Message Header.

8.2.5 Component Information – Logical

CLASS, COMPONENT, SUBCOMPONENT1, SUBCOMPONENT2, USER-NAME, ROLE, DESTINATION-COMPONENT

These fields provide for source traceability of a message. A C2MS Class is a high-level category of functionality. A C2MS Subclass is defined as a subset genre of a Class. A Class is composed of one or more subclasses. A C2MS Component is defined as the name of the executing software application that fulfills, in part or in whole, an instance of a C2MS Class or Subclass. Class and Subclass are sometimes used interchangeably to refer to a type of system, whereas component always refers to an actual piece of software. The component generates C2MS messages and identifies in the Message Header the Class and/or Subclass to which it belongs. It has the option to further delineate the source of a message by using the SUBCOMPONENT1 and SUBCOMPONENT2 fields. The relationship between the C2MS Classes and Subclasses is shown in Table A-1, Product Categories.

The implementing software will optionally supply the USER-NAME field, which is the name or owner of the account that started the component. The ROLE of the component is optionally supplied to indicate what function the component is assigned to play in the overall concept of operations. Components will perform different functions depending on their roles and responsibilities. The DESTINATION-COMPONENT field is the intended recipient of the message and also appears in the subject name of many messages.

8.3 Control and Monitor Level Messages

8.3.1 Log Message

A Log Message is time-tagged text generated by an application to notify the operator that a ground system or satellite event has occurred. Log Messages can be as trivial as those giving user confirmations but also used to convey the severity of a situation.

An example of a trivial type of Log Message is one that notifies the user that a display page request is complete. Log Messages can also provide error information such as device failures or operator input errors.

Another type of Log Message is one that identifies a certain occurrence or event has happened; for example, Loss-of-Signal is detected for a satellite data stream.

If Log Messages are saved in an archive, they can provide a wealth of data as well as a chronological history of the ground system activities. This audit trail can be very useful in troubleshooting and reporting.

Table 8-4. Log Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Publish
Receiver	Expert Subclass, Alert Subclass, and Assessment Subclass that uses Log Messages
Receivers Intended Usage	Subscribe
What	Log action or event for display, archive, data mining, reporting, etc.
When	As needed

Examples

1. Any component needing to disseminate information should publish a Log Message.

2. A Message Logger application may subscribe to all messages to place in an archive.

3. A flight dynamics application may subscribe to Flight Dynamics Notification event type.

4. A display application may subscribe to Critical (Level 4) Severity messages.

Table 8-5. Log Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	mission	[constell ation]	[sat]	MSG	LOG	[COMPON ENT]	[Subclass: ARC, CFG, CMD, DIR, … TLM]	[Occur-rence: AOS, LOS, …CRITICAL , …WARNIN G]	[Severity: 1-Normal, INFO, 2-Warning, WARN, …, 4-Critical, FATAL]	[user, optional]	[ref ID, optional]
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	LOG	TLM3	TLM	CRITICAL	4	ws3	794
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	LOG	TLM3	CMD	CMDV	4	ws5	123
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	*	MSG	LOG	*	*	*	*	>	

8.3.1.1 Log Message Subject Names

Table 8-6. Properties of the *Miscellaneous Elements* for the Log Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of publisher	COMPONENT from header
ME2	Required	The subclass of the log message	SUBCLASS
ME3	Required	The occurrence type of the event	OCCURRENCE-TYPE
ME4	Required	The severity of the log message	SEVERITY
ME5	Optional	The user or work-position originating the log message	USER
ME6	Optional	A reference ID assigned to the log message	REFERENCE-ID

Examples for Publisher / Sender

App1, TLM2, and TLM3 each send a Log Message.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.LOG.APP1.TLM.WARNING.4

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.LOG.APP1.CMD.CMDV.4.WS3.794

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.LOG.TLM2.TLM.CRITICAL.4.DECOM1.456

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.LOG.TLM3.SCH.1.NORMAL
```

Examples for Subscriber / Receiver

```
C2MS.*.*.*.*.*.MSG.LOG. >

C2MS.*.*.MSSN.*.*.MSG.LOG. >

C2MS.*.*.*.*.SAT1.MSG.LOG.TLM2. >

C2MS.*.*.MSSN.*.*.MSG.LOG.*.*.*.4. >
```

Note that since some elements are optional, it is best to subscribe with the “>” character to ensure capturing all messages.

8.3.1.2 Log Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Log Message.

Table 8-7. Log Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	LOG

8.3.1.3 Log Message Contents

The following figure shows a UML object diagram of the Log Message with its required and optional fields.

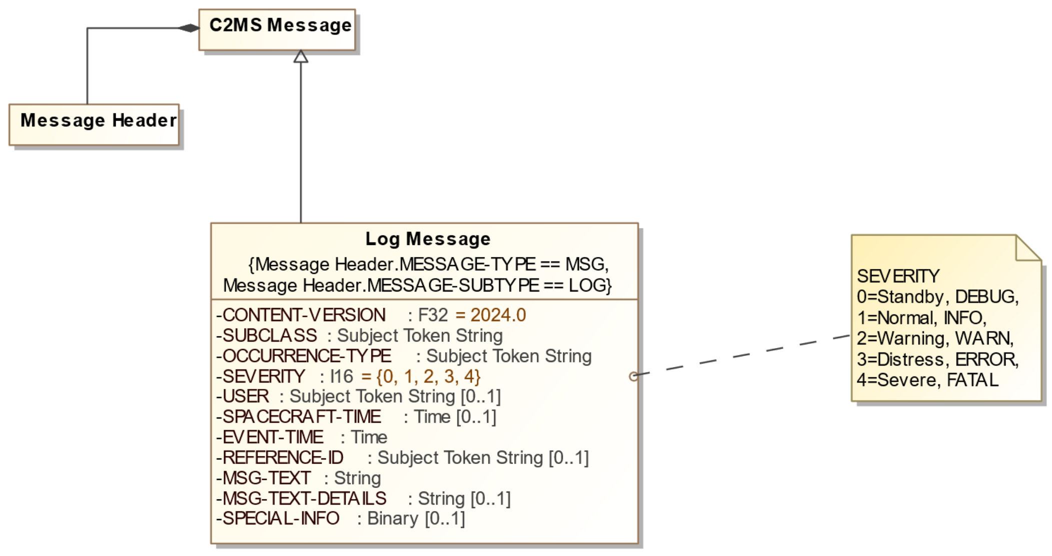

C2MS Message
Message Header
Log Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == LOG}
-CONTENT-VERSION : F32 = 2024.0
-SUBCLASS : Subject Token String
-OCCURRENCE-TYPE : Subject Token String
-SEVERITY : I16 = {0, 1, 2, 3, 4}
-USER : Subject Token String [0..1]
-SPACECRAFT-TIME : Time [0..1]
-EVENT-TIME : Time
-REFERENCE-ID : Subject Token String [0..1]
-MSG-TEXT : String
-MSG-TEXT-DETAILS : String [0..1]
-SPECIAL-INFO : Binary [0..1]
SEVERITY
0=Standby, DEBUG,
1=Normal, INFO,
2=Warning, WARN,
3=Distress, ERROR,
4=Severe, FATAL

Figure 8-3. Log Message Diagram

The following table describes additional field names, values, and notes for the Log Message.

Table 8-8. Log Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message
SUBCLASS	Subject Token String	R	See Table A-1 Software Class and Subclass Categories.		Subclass generating the log message (or applicable subsystem of which the log message belongs)
OCCURRENCE-TYPE	Subject Token String	R	See tables in Section 8.3.1.4 Log Message Occurrence Types		An occurrence type that categorizes the kind of activity or event that happened, triggering the log message
SEVERITY	I16	R	**Value**	**Description**	Indicates the severity of the Log Message. Scale traditionally applied to message based on requirements and characteristics of the component or ground system. The severity may be used to alert the operator in some way such as visual or audible notification.
			0	Standby, DEBUG	
			1	Normal, INFO	
			2	Warning, WARN	
			3	Distress, ERROR	
			4	Critical, FATAL	
USER	Subject Token String	O			Which user/work position/proc the message has to do with
SPACECRAFT-TIME	Time	O			Time event happened (may be earlier than actual posted time)
EVENT-TIME	Time	R			Time event happened (may be earlier than published time)
REFERENCE-ID	Subject Token String	O			A local index or map to a table of additional information
MSG-TEXT	String	R			Text for display (typically about 60 characters)

Field Name	Type	Presence	Value	Description
MSG-TEXT-DETAILS	String	O		One or more paragraphs that includes more detail. Suggested corrective action. Suggest specifying URL in this field
SPECIAL-INFO	Binary	O		application use

8.3.1.4 Log Message Occurrence Types

The following tables, starting with Table 8-9, list suggested Log Message occurrence types. The OCCURENCE-TYPE field is a required field. The tables contain lists of ground system occurrences that could be used to identify the event that triggered the generation of the Log Message.

The component is not required to put out a Log Message for each type of occurrence or state change. But if the Log Message is issued, the suggested format is strongly recommended. By implementing common Log Message occurrence types along with the recommended formats, the following goals can be achieved for Log Messages, developers, and mission operations:

Decipherable – Notifications through Log Messages should be readable, self-explanatory, and easily recognizable whether in a display or hardcopy report.

Discernable – Log Messages should be easy to parse or extract information from, particularly the MSG-TEXT field, or any field for that matter (such as the OCCURRENCE-TYPE). Putting form to the MSG-TEXT field for common or important Log Messages will make them understandable and the situation more apparent and thus, easier to act upon.

Deterministic – Once a Log Message has been recognized and information extracted from it (and others to provide context), making a decision and taking action becomes more perceptible. It also becomes easier to pre-define the actions and program them into clear, unambiguous rules.

A final benefit to standard looking Log Messages is that the combined C2MS compliant components provide more uniform look-and-feel to the users making it easier for multi-system training across mission operation centers and even enterprise centers.

This is not a complete list of occurrence types. Additional values can be added as necessary.

8.3.1.4.1 Pass Related Occurrence Types

Table 8-9. Pass-Related Occurrence Types

Value	Occurrence Type	Occurrence Description
PREPASS	Pre-Pass	Period of time prior to start of a pass. Allows for allocation, setup, and check out of resources, communication pathways, and general preparation. Could be ~5-10 minutes in length.
PASSSTART	Planned Start Time of Pass	Time the pass is scheduled to start. (Scheduled AOS time).
AOS	Acquisition of Signal	Actual time of the AOS when ground (antenna) receives the signal; or could be the time in mission operation center when first data is received.
LOS	Loss of Signal	Actual time the data drops out or ceases transmission.
PASSEND	Planned End Time of Pass	Time the pass is scheduled to end. (Scheduled LOS time).

Value	Occurrence Type	Occurrence Description
POSTPASS	Post-Pass	Period of time immediately after the LOS to wrap up the pass activities. Could include deallocation of resources, producing pass summary reports, initiation of offline data processing, and so on. Could last a few minutes or until next Pre-Pass.

For pass-related occurrences, it is recommended that the MSG-TEXT portion of the Log Message contain the Occurrence Type value and the EVENT-TIME value in the format of:

```
[OCCURRENCE-TYPE] Time: [EVENT-TIME]
```

Examples

```
PREPASS Time: 2014-74-16:18:15
PASSSTART Time: 2014-74-16:28:15
AOS Time: 2014-74-16:28:16
PASSEND Time: 2014-74-16:46:30
LOS Time: 2014-74-16:46:40
POSTPASS Time: 2014-74-16:47:20
```

8.3.1.4.2 Telemetry Limit Violation Occurrence Types

Table 8-10. Telemetry Limit Violation Occurrence Types

Value	Occurrence Type	Occurrence Description
DEPRECATED RED	Red Limit	Reports mnemonic entering or exiting red limit condition
DEPRECATED YEL	Yellow Limit	Reports mnemonic entering or exiting yellow limit condition
DEPRECATED NORM	Normal Range	Reports mnemonic returning to a normal (within range) condition.
NORMAL	Normal Range	Reports mnemonic returning to a Normal (within range) condition.
WARNING	Warning Limit	Reports mnemonic entering or exiting Warning limit condition
DISTRESS	Distress Limit	Reports mnemonic entering or exiting Distress limit condition
CRITICAL	Critical Limit	Reports mnemonic entering or exiting Critical limit condition

For telemetry limit violation notices, it is recommended that the MSG-TEXT portion of the Log message contain the following information.

[term for value] [tlm word #] [mnemonic] [violation description] "`the`" [limit description] "`Limit of`" [threshold value] "`with a value of`" [mnemonic value] [mnemonic units]

Examples

```
LRV #102 BATVOLT1 exceeded the Lower Warning limit of −12 with a value of –
13 volts
TLM #156 BATVOLT2 is above the Upper Distress limit of 15 with a value of
15.75 volts
CVT #156 BATVOLT3 is within the Normal limit of 16 with a value of 14.75
volts
```

Note:

`LRV` - Last Received (or recorded) Value

`TLM` – Telemetry
`CVT` – Current Value Table

8.3.1.4.3 Command Verification Occurrence Types

Table 8-11. Command Verification Occurrence Types

Value	Occurrence Type	Occurrence Description
XFRD	transferredToRange	The network that connects the ground system to the spacecraft has received the command. (Comes from something other than the spacecraft.)
SENT	sentFromRange	The command has been transmitted to the spacecraft by the network that connects the ground system to the spacecraft. (Verifier comes from something other than the spacecraft.)
RCVD	Received	The SpaceSystem has received the command.
ACPT	Accepted	The SpaceSystem has accepted the command.
QUED	Queued	The SpaceSystem has scheduled the command for execution.
EXEC	Executing	The command is being executed.
COMP	Complete	Command is considered complete.
FAIL	Failed	The command failed.
TIMEOUT	Timeout	Time expired for the command to complete. A specific instance of failed.

The values in the table above were taken from the master schema for the OMG Space Domain Task Force XML Telemetric and Command Exchange (XTCE) format. This document is found at https://www.omg.org/space/xtce.

No recommended format has been defined for this occurrence type.

8.3.1.4.4 Miscellaneous Occurrence Types

Table 8-12. Miscellaneous Occurrence Types

Value	Occurrence Type	Occurrence Description
CFG	Configuration Change	Reports that the physical or logical configuration of the equipment and/or software has changed.
DIR	Directive	The echo of a directive message issued by the operator, command procedure, command schedule, or automated process
FDN	Flight Dynamics Notification	Reports completion of flight dynamics process or product
OPER	Operator Information	Operator entered log message
ORB	Orbital Event	Reports calculated orbital event such as orbit number, ascending/descending node, eclipse state
PROD	Product Available and/or generated	Reports that a product has been generated and is available to access
SAT	Satellite	Indicates/reports activity occurred on the satellite
SYS	System/Software	Reports detected system/software error or unexpected condition

DIR

It is recommended to follow the table below when a Log Message is used to echo a Directive Request Message.

Table 8-13. Log Message to Echo a Directive Request Message

	Retrieve From Here	And Insert Into Here
Message	Directive Request Message	Log Message
Field	DIRECTIVE-STRING	MSG-TEXT

PROD

It is recommended to follow the table below when a Log Message is used to echo a Product Message.

Table 8-14. Product Message to Echo a Directive Request Message

	Retrieve From Here	And Insert Into Here
Message	Product Message	Log Message
Field	TIME-COMPLETED	EVENT-TIME
Field	PROD-NAME PROD-TYPE PROD-SUBTYPE PROD-DIST-METHOD URI	MSG-TEXT

The MSG-TEXT format of the Log Message would be in the following format:

> "Product Type/Subtype:" [PROD-TYPE] / [[PROD-SUBTYPE], "Created:" [TIME-COMPLETED], "Available By:" [PROD-DIST-METHOD] {- URI}

Examples

```
Product Type/Subtype: PAS / Contact Schedule, Prod ID: WhiteSands124,
Created: 2014-123-14:32:25, Available By: URI -
Facility.Node.Computer.Disk.directory.filename

Product Type/Subtype: PAS / Schedule, Prod ID: SCHED124, Created: 2014-
123-14:32:25, Available By: PROD REQ
```

8.4 Archive Message Retrieval Messages

Archive Message Retrieval Messages provide access to messages, excluding telemetry data, stored in a C2MS message archive. The Archive Message Retrieval Request is used to request messages from the message archive by either:

1. Providing a specific query-like statement to be used directly by the responder to access the underlying data storage mechanism. These statements could be in the form of a Structured Query Language (SQL) statement for a database, a grep statement to search for strings inside a file, or Perl script statement.

2. Providing a time range and pairs of message types/subtypes for a coarse description and gathering of messages.

The Archive Message Retrieval Response returns the status of the Request and the location of the output product. Although telemetry messages could be archived and retrieved, they are categorized and treated separately since their data requires specialized processing.

Any number of C2MS messages may be archived by any number of components. For example, one component may only archive C2MS Log messages. Another component may archive Log Messages and all Directive Request Messages. Also, components that archive messages do not necessarily have to provide a service to other components to extract those messages. The messages may be for that component's internal use only. How the C2MS messages are archived and organized is left up to the mission. Lastly, as inferred from above, the method of storing the messages is not defined. A database, flat text files, or any other means could be used.

8.4.1 Archive Message Retrieval Request

The Archive Message Retrieval Request is a service request issued when an application desires messages from a message archive. The component issues an Archive Message Retrieval Request to an archive provider component that has access to a message archive. The request specifies the time range and message types.

Table 8-15. Archive Message Retrieval Request Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Request
Receiver	Any C2MS compliant application that provides access to a message archive, e.g., GREAT
Receivers Intended Usage	Subscribe
What	C2MS messages
When	As needed

Examples

1. An Archive and Assessment component may request archived messages, such as Log Messages for limit violations, in order to generate a report.

2. A Planning and Scheduling component may request archived messages for the re-planning of a schedule, or may even request future event messages that have been archived.

3. An analyst may want all archived messages in a specific time window for further problem analysis.

The method for identifying, matching, and extracting messages from the archive is to specify a few key fields in the messages. This method will result in a coarse selection of messages for perusal. It is not meant to identify a specific value of a specific field of a specific message - though that is possible in some circumstances. Typically, a group of associated messages will be located, extracted, and returned.

The Archive Message Retrieval Request Message Header identifies the message as a C2MS Archive Message Retrieval Request. The message contents specify the parameters for extraction out of an archive. The parameters are:

Time range

- Type of time: spacecraft or a C2MS standard format
- Start and stop times of messages from which to retrieve

Messages to examine

- Type and Subtype pairing
- Name of field in the messages that contains the time to use for the time range

Fields in messages to pattern match

- Other fields in the message to extract the contents and pattern match

Other fields in the message content specify the output. They are:

Location of the one output file for the product

- Uniform Resource Identifier (URI) location to store the output file
- File name of the output file

File description

- Maximum size of the output file
- Format of the output file
- Version number of the format

Only one file is expected for the output product. The requestor of archived messages has the option of getting the resulting product file in either or both of the following ways:

- By reference within the response message
- Included within the response message

The C2MS standard messages that may be archived contain different time fields. Some are spacecraft time and some are in the C2MS standard time format. Also, times in messages can be the actual occurrence time of an event, the requested execution time, or the publish time of a message. These times are naturally named differently. Therefore, the Request Message must specify the name of the time field in the message to use for the boundaries of the time range.

A previous version of C2MS supported what was called "shorthand" queries in which the requestor could send in a REQ-STRING rather than enumerating a set of MSGs, specified by NUM-OF-MSGS, to formulate the query. This REQ-STRING contains "a database query, a script expression, Unix statement, or some other statement" to be executed within the responding service on behalf of the requestor. REQ-STRING and the "shorthand" method have been deprecated over security concerns.

What was previously termed the "longhand" method, enumerating a set of MSGs, specified by NUM-OF-MSGS, has been retained and is now the only non-deprecated method for performing this request.

Note that while REQ-STRING is retained in the C2MS Message definition with DEPRECATED designation, a service implementer may choose not to allow REQ-STRING to be used for "shorthand" queries due to those same security concerns.

8.4.1.1 Archive Message Retrieval Request Subjects

Table 8-16. Archive Message Retrieval Request Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	REQ	AMSG	[DESTINATION-COMPONENT]					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	AMSG	ARCHIVER					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	AMSG	ANALYZER					
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	*	REQ	AMSG	ANALYZER					

Table 8-17. Properties of the *Miscellaneous Elements* for Archive Message Retrieval Request

Miscellaneous Element	**Required / Optional**	**Description**	**Field in Msg, if applicable**
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2	Not used		
ME3	Not used		

Examples

Two components, ANALYZER and ARCHIVER, interact with the Archive Message Retrieval Request.

ANALYZER subject name to send the Archive Message Retrieval Request to ARCHIVER:

```
C2MS.FILL.FILL.FILL.FILL.FILL.REQ.AMSG.ARCHIVER
```

ARCHIVER subject name to receive its own Archive Message Retrieval Request:

```
C2MS.*.*.*.*.*.REQ.AMSG.ARCHIVER
```

8.4.1.2 Archive Message Retrieval Request Message Header

The abbreviated following table shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Archive Message Retrieval Request Message header.

Table 8-18. Archive Message Retrieval Request Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	AMSG

8.4.1.3 Archive Message Retrieval Request Contents

The following figure shows a UML object diagram of the Archive Message Retrieval Request with its required, optional, and dependent fields.

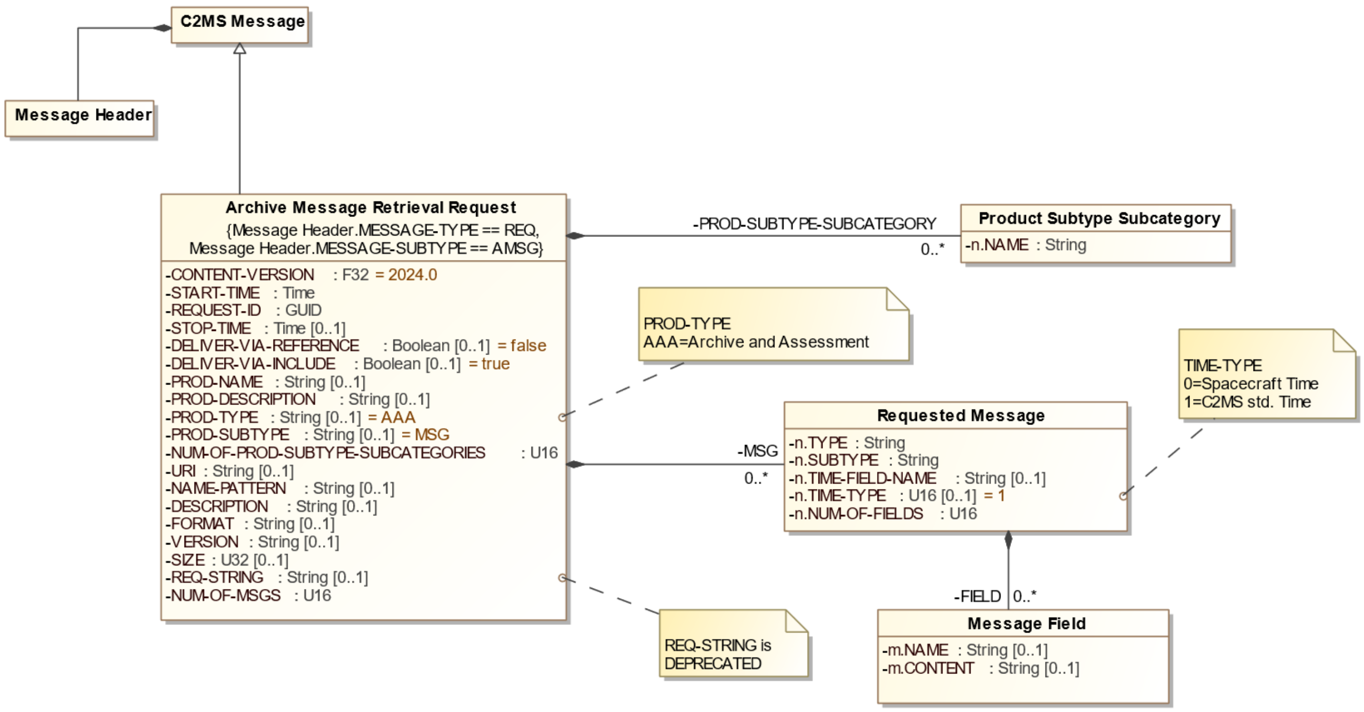

C2MS Message
Message Header
Archive Message Retrieval Request
{Message Header.MESSAGE-TYPE == REQ, Message Header.MESSAGE-SUBTYPE == AMSG}
-CONTENT-VERSION : F32 = 2024.0
-START-TIME : Time
-REQUEST-ID : GUID
-STOP-TIME : Time [0..1]
-DELIVER-VIA-REFERENCE : Boolean [0..1] = false
-DELIVER-VIA-INCLUDE : Boolean [0..1] = true
-PROD-NAME : String [0..1]
-PROD-DESCRIPTION : String [0..1]
-PROD-TYPE : String [0..1] = AAA
-PROD-SUBTYPE : String [0..1] = MSG
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-URI : String [0..1]
-NAME-PATTERN : String [0..1]
-DESCRIPTION : String [0..1]
-FORMAT : String [0..1]
-VERSION : String [0..1]
-SIZE : U32 [0..1]
-REQ-STRING : String [0..1]
-NUM-OF-MSGS : U16
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
PROD-TYPE
AAA=Archive and Assessment
TIME-TYPE
0=Spacecraft Time
1=C2MS std. Time
-MSG
0..*
Requested Message
-n.TYPE : String
-n.SUBTYPE : String
-n.TIME-FIELD-NAME : String [0..1]
-n.TIME-TYPE : U16 [0..1] = 1
-n.NUM-OF-FIELDS : U16
-FIELD
0..*
Message Field
-m.NAME : String [0..1]
-m.CONTENT : String [0..1]
REQ-STRING is DEPRECATED

Figure 8-4. Archive Message Retrieval Request Diagram

The following table describes additional field names, values, and notes for the Archive Message Retrieval Request.

Table 8-19. Archive Message Retrieval Request Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message
START-TIME	Time	R		Requested start time of the messages to be retrieved from the Message Archive
REQUEST-ID	GUID	R		ID to identify the request message
STOP-TIME	Time	O		Requested stop time of the messages to be retrieved from the Message Archive. Defaults to the end of the Message Archive.
DELIVER-VIA-REFERENCE	Boolean	O		Indicates if the data will be referenced by a URI in the single response message. Defaults to False.
DELIVER-VIA-INCLUDE	Boolean	O		Indicates if the data is to be included in the single response message. Defaults to True.
PROD-NAME	String	O		Name of the product being requested
PROD-DESCRIPTION	String	O		Description of the product in text or xml
PROD-TYPE	String	O	**Value** **Description**	

Field Name	Type	Presence	Value		Description
			AAA	Archive and Assessment	Product type and subtype being requested. (See Table A-2, Product Categories)
PROD-SUBTYPE	String	O	MSG		
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R			Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6*, etc. in the Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D			First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc. of the Product Message) - "n" starts at "1". This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
URI	String	O			Location where the requesting component is asking for the product file(s) to be stored. Could be a web address, directory, or folder specification
NAME-PATTERN	String	O			Describes the name of the output file
DESCRIPTION	String	O			Description of the file in text or xml
FORMAT	String	O			For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
VERSION	String	O			Identifies the version of the file
SIZE	U32	O	Kilobytes		Maximum size of the file acceptable to the requester. Size specified in KB.
DEPRECATED REQ-STRING	String	O			Note that is field has been deprecated and will be removed in a future release of C2MS. Deprecated description: Specific to the responder / provider of the requested information. The string will define a database query, a script expression, Unix statement, or some other statement for extracting the information from the provider's repository.
NUM-OF-MSGS	U16	R	1+		Indicates the number of different message type / subtype pairs requested from the Message Archive.
MSG.n.TYPE	String	D			Message Type/Subtype pairing to identify the message to be retrieved from the archive - "n" starts at "1". This field is required for each message specified by NUM-OF-MSGS.
MSG.n.SUBTYPE	String	D			
MSG.n.TIME-FIELD-NAME	String	O			Name of field in the message that contains the time to examine. Will default to PUBLISH-TIME in Message Header.
MSG.n.TIME-TYPE	U16	O	**Value**	**Description**	Indicates the format of the time to examine in the retrieved messages. Defaults to C2MS standard time format.
			0	Spacecraft Time	
			1	C2MS std. Time	
MSG.n.NUM-OF-FIELDS	U16	D			Number of message fields to examine and match for retrieval from the archive. This field is required for each message specified by NUM-OF-MSGS.
MSG.n.FIELD.m.NAME	String	O			Name of the message field to match for retrieval from the archive - both "n" and "m" start at "1".

Field Name	Type	Presence	Value	Description
MSG.n.FIELD.m.CONTENT	String	O		Contents of the message field used in matching the messages for retrieval from the archive

START-TIME and STOP-TIME

The requestor could specify the START and STOP times in the ways shown in the following table. At least one of the start and stop times must be absolute.

Table 8-20. Examples of Start and Stop Times

Start Time	Stop Time	Example	Description
Absolute	Absolute	START: 2014-123-14:30:00 STOP: 2014-123-14:30:10	Boundaries of the Start and Stop time have been exactly specified.
Absolute	Relative (duration)	START: 2014-123-14:30:00 STOP: +10:00	Time window has an absolute start time and extends for 10 minutes.
Relative (duration)	Absolute	START: -10:00 STOP: 2014-123-14:30:00	Time window will end at a specified stop time and start 10 minutes prior to that.
Relative	Relative	Not Applicable	Unless there has been some previously established baseline time upon which to offset the duration times (such as “now”), this combination is ambiguous.

8.4.2 Archive Message Retrieval Response

An archive message provider responds to an Archive Message Retrieval Request by sending an Archive Message Retrieval Response. The archive message provider must return the status of the action completed along with the location of the Archive Retrieval Message file product.

A series of Archive Message Retrieval Responses may be required. In this case, an initial acknowledgement response message is issued, followed by interim or interactive "working" response type messages to let the requesting application know that the request is still being processed, and finally by a completion response type message.

Please see Section 6.4 C2MS Messages: Their Characteristics and Interactions for a general discussion on these types of messages.

Table 8-21. Archive Message Retrieval Response Summary

Sender	Application that received the Archive Message Retrieval Request
Senders Intended Usage	Publish or Reply
Receiver	Application that issued the Archive Message Retrieval Request
Receivers Intended Usage	Subscribe
What	Provide success/failure response to the service that was requested and the knowledge to access the extracted messages
When	Upon receipt of the Archive Message Retrieval Request, on an interval for those services that are time-consuming, and on completion.

Examples

1. Acknowledge receipt of an Archive Message Retrieval Request.
2. Indicate the Archive Message Retrieval Request is still being processed.
3. Indicate the Archive Message Retrieval Request has successfully completed.

8.4.2.1 Archive Message Retrieval Response Subjects

Table 8-22. Archive Message Retrieval Response Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	AMSG	[DESTINATION-COMPONENT]	[Response Status: 1-acknowledge ment, …4-failed]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	AMSG	ARCVR	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	AMSG	ARCVR	3				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	*	RESP	AMSG	APP1	*				

Table 8-23. Properties of the *Miscellaneous Elements* for the Archive Message Retrieval Response

Miscellaneous Element	Required / Optional	Description	Field Origination in Msg, if applicable
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components, APP1 and ARCVR, interact with the Archive Message Retrieval Response.

1. ARCVR subject to send the Archive Message Retrieval Response to APP1:

   ```
   C2MS.FILL.FILL.FILL.FILL.FILL.RESP.AMSG.APP1.3
   ```
2. APP1 subject to receive its own Archive Message Retrieval Responses:

   ```
   C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.AMSG.APP1.*  or
   C2MS.*.*.*.*.*.RESP.AMSG.APP1.>
   ```

8.4.2.2 Archive Message Retrieval Response Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Archive Message Retrieval Response Message header.

Table 8-24. Archive Message Retrieval Response Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	AMSG

8.4.2.3 Archive Message Retrieval Response Contents

The following figure shows a UML object diagram of the Archive Message Retrieval Response with its required and optional fields.

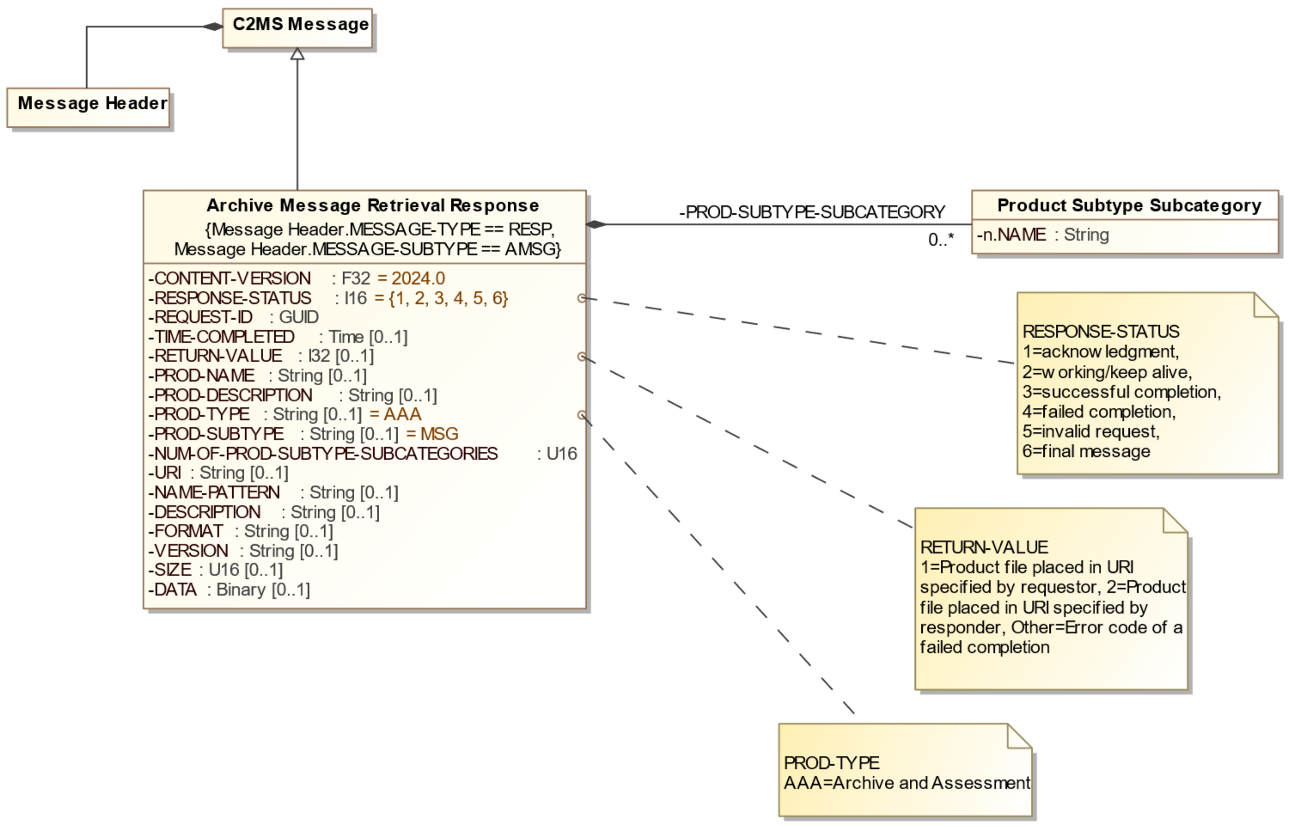

C2MS Message
Message Header
Archive Message Retrieval Response
{Message Header.MESSAGE-TYPE == RESP, Message Header.MESSAGE-SUBTYPE == AMSG}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1, 2, 3, 4, 5, 6}
-REQUEST-ID : GUID
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-PROD-NAME : String [0..1]
-PROD-DESCRIPTION : String [0..1]
-PROD-TYPE : String [0..1] = AAA
-PROD-SUBTYPE : String [0..1] = MSG
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-URI : String [0..1]
-NAME-PATTERN : String [0..1]
-DESCRIPTION : String [0..1]
-FORMAT : String [0..1]
-VERSION : String [0..1]
-SIZE : U16 [0..1]
-DATA : Binary [0..1]
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
RESPONSE-STATUS
1=acknowledgment,
2=working/keep alive,
3=successful completion,
4=failed completion,
5=invalid request,
6=final message
RETURN-VALUE
1=Product file placed in URI specified by requestor, 2=Product file placed in URI specified by responder, Other=Error code of a failed completion
PROD-TYPE
AAA=Archive and Assessment

Figure 8-5. Archive Message Retrieval Response Diagram

The following table describes additional field names, values, and notes for the Archive Message Retrieval Response.

Table 8-25. Archive Message Retrieval Response Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	Value	Description	Identifies the status of the Archive retrieval message Request being processed
			1	Acknowledgement	
			2	Working/keep alive	
			3	Successful completion	
			4	Failed completion	
			5	Invalid Request	
			6	Final Response	
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
TIME-COMPLETED	Time	O			Time application completed processing the request
RETURN-VALUE	I32	O	Value	Description	Return value or status based on the RESPONSE-STATUS. Used to indicate product URI as requestor or responder. Also,
			1	Product file placed in URI specified by requestor	

Field Name	Type	Presence	Value		Description
			2	Product file placed in URI specified by responder	can be used to provide function call status or error code in the case of failed completion
			Other	Error code of a failed completion	
PROD-NAME	String	O			Name of the product being returned
PROD-DESCRIPTION	String	O			Description of the product in text or xml
PROD-TYPE	String	O	**Value**	**Description**	Product type and subtype being requested. (See Table A-2 Product Categories).
			AAA	Archive and Analysis	
PROD-SUBTYPE	String	O	MSG		
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R	0+		Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6,* etc. in Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D			First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc. of the Product Message) - "n" starts at "1". This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
URI	String	O			URI specifying the location where the file product is stored
NAME-PATTERN	String	O			Describes the name of the file
DESCRIPTION	String	O			Description of the file contents in text or xml
FORMAT	String	O			For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
VERSION	String	O			Indicates the version of the file
SIZE	U16	O	KB		Actual size of the file
DATA	Binary	O			The file content

A product consisting of a single file is returned in the response message. The file contains the messages that satisfied the criteria specified in the Archive Message Retrieval Request Message.

Table 8-26. Meaning of Response Status and Return Value with Recommended Actions

User Specified the URI	RESPONSE-STATUS	RETURN-VALUE	URI	Action
N	Successful	2 (The only meaningful value)	Product was generated and placed in URI chosen by responder	Requestor should retrieve file at responder's URI location
Y	Successful	1	Product was generated and placed in URI specified by requestor	Requestor should retrieve file at the specified URI
Y	Successful	2	Product was generated but placed in alternate URI chosen by responder	Requestor should retrieve file at responder's URI location
Y or N	Failed	3	Product exceed maximum requested file size or the default maximum file size	

8.5 Directive Messages

Directives (which are instructions directing a component to action) may originate from a Graphical User Interface (GUI), command line, schedule, procedure, or virtually anywhere within a space-ground system.

The Directive Request Message is the mechanism to send a directive to a component.

The Directive Response Message is used to return an acknowledgement and status of the directive action to the originator.

Please see Section 6.3.7 C2MS Messages: Their Characteristics and Interactions for a general discussion about these types of messages.

8.5.1 Directive Request Message

A Directive Request Message is the means for one application to request a service from, or an action to be taken by, another application. Directives themselves can be input from a user through a GUI or command line, or as part of the internal logic of a component. They can also be grouped together with logic in a file as a procedure (or proc).

Directives can also be found in a command schedule organized by time for automatic execution. Typically, as automation increases, more and more Directive Request Messages are generated internally as certain data conditions are detected.

The Directive Request Message is primarily used for requests that are also to be visible to the operations staff and are directly related to the overall mission of the system. For example, a Directive Request would normally be used to page a Flight Operations Team member. Directive Messages will typically include a text string the receiver will parse to determine the function is being requested.

Table 8-27. Directive Request Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Request a service or function
Receiver	Application providing a service, or an application collecting directives for audit trail purposes
Receivers Intended Usage	Subscribe
What	Action or service request initiated by user, software, procedure, command schedule, etc.
When	Upon detection of data condition, upon scheduled time, upon entry

Examples

1. An operator input issuing a satellite command.
2. An operator input requesting a display page.
3. A request to start, pause, stop, or resume a schedule.
4. A request to initiate pre-pass setup.
5. Any request issued from a procedure.

8.5.1.1 Directive Request Subjects

Table 8-28. Directive Request Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	REQ	DIR	[DESTINATION-COMPONENT]	[DIRECTIVE-KEYWORD]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	DIR	APP1	[DO_ABC]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	DIR	APP2	[DO-XYZ]				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	REQ	*	APP4	[DO_ABC]				

Table 8-29. Properties of the *Miscellaneous Elements* for the Directive Request Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2	Optional	A keyword the receiving process could use for filtering	DIRECTIVE-KEYWORD
ME3 …	Not used		

Examples

Two components, APP1 and APP4, interact with the Directive Request Message.

APP1 subject to send the Directive Request to APP4:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.DIR.APP4
```

APP4 subject to receive its own Directive Request Messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.DIR.APP4.>
```

APP4 subject to receive any APP4 Request Message:

```
C2MS.*.*.*.*.*.REQ.*.APP4.>
```

APP4 subject to receive a specific Directive Request Message:

```
C2MS.*.*.*.*.*.REQ.DIR.APP4.DO_ABC
```

8.5.1.2 Directive Request Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Directive Request Message header.

Table 8-30. Directive Request Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	DIR

8.5.1.3 Directive Request Contents

The following figure shows a UML object diagram of the Directive Request Message with its required and optional fields.

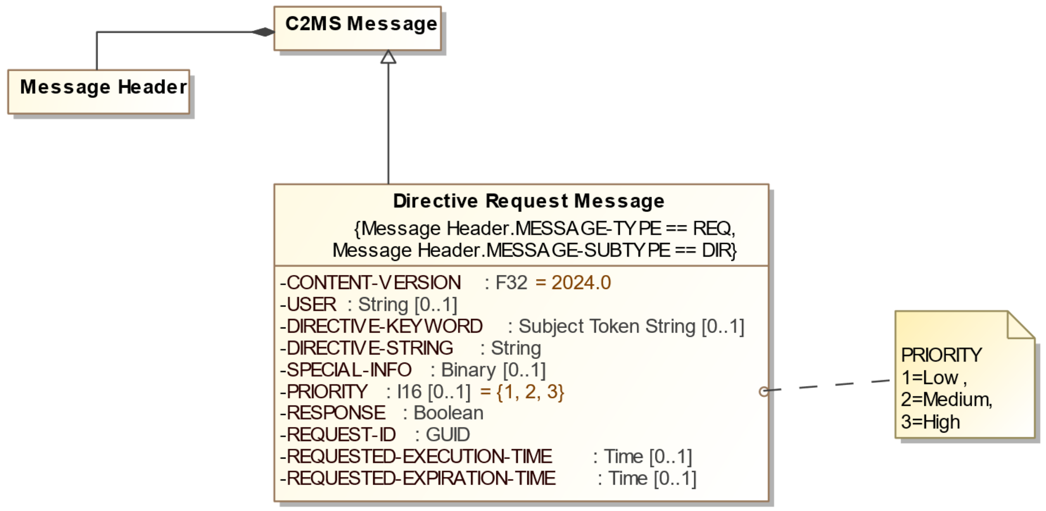

C2MS Message
Message Header
Directive Request Message
{Message Header.MESSAGE-TYPE == REQ,
Message Header.MESSAGE-SUBTYPE == DIR}
-CONTENT-VERSION : F32 = 2024.0
-USER : String [0..1]
-DIRECTIVE-KEYWORD : Subject Token String [0..1]
-DIRECTIVE-STRING : String
-SPECIAL-INFO : Binary [0..1]
-PRIORITY : I16 [0..1] = {1, 2, 3}
-RESPONSE : Boolean
-REQUEST-ID : GUID
-REQUESTED-EXECUTION-TIME : Time [0..1]
-REQUESTED-EXPIRATION-TIME : Time [0..1]
PRIORITY
1=Low ,
2=Medium,
3=High

Figure 8-6. Directive Request Diagram

The following table describes additional field names, values, and notes for the Directive Request Message.

Table 8-31. Directive Request Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
USER	String	O		Which user/work position/proc/schedule the message is coming from
DIRECTIVE-KEYWORD	Subject Token String	O		Keyword extracted from the directive string. Useful for routing/processing
DIRECTIVE-STRING	String	R		Full directive string that includes the keyword
SPECIAL-INFO	Binary	O		For application use
PRIORITY	I16	O	**Value** / **Description**: 1 Low; 2 Medium; 3 High	Indicates processing priority, if applicable
RESPONSE	Boolean	R		Indicates if a response is required.
REQUEST-ID	GUID	R		ID to identify the request message
REQUESTED-EXECUTION-TIME	Time	O		Absolute or relative time can apply.

Field Name	Type	Presence	Value	Description
REQUESTED-EXPIRATION-TIME	Time	O		Absolute or relative time can apply.

For an explanation on how the REQUESTED-EXECUTION-TIME and REQUESTED-EXPIRATION-TIME could operate, see Table 8-20 Examples of Start and Stop Times.

8.5.2 Directive Response Message

A Directive Response Message is sent by an application in response to a Directive Request Message. The Directive Response Message will provide acknowledgment of the Directive Request Message and a status of the action completed. A series of Directive Response Messages may be required in the case where the processing of the action is not immediate.

An example of this would be an archive retrieval, a plot, or an orbit determination calculation. In this event, an interim or interactive "working" type message would be issued to let the requesting application know that the action is still being processed.

Please see Section 6.3.7 C2MS Messages: Their Characteristics and Interactions for a general discussion on these types of messages.

Table 8-32. Directive Response Message Summary

Sender	Application that received the Directive Request Message corresponding to the Directive Response Message
Senders Intended Usage	Reply
Receiver	Application that issued the Directive Request Message or an application collecting Directive Response Messages for audit trail purposes
Receivers Intended Usage	Subscribe
What	Provide success/failure response to the service that was requested
When	Upon receipt of Directive Request Message or at intervals for those services that are time-consuming

Examples

1. Acknowledge receipt of a directive.

2. Indicate the directive is still being processed.

3. Indicate the directive has successfully completed.

8.5.2.1 Directive Response Subjects

Table 8-33. Directive Response Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	RESP	DIR	[DESTINATION-COMPONENT]	[Status]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	DIR	APP1	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	DIR	APP4	4				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	RESP	DIR	APP4	*				

Table 8-34. Properties of the *Miscellaneous Elements* for the Directive Response Message

Miscellaneous Element	Required / Optional	Description	Field Origination in Msg, if applicable
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components, APP4 and APP1, interact with the Directive Response message.

APP1 subject to send the Directive Response to APP4:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.DIR.APP4.3
```

APP4 subscribes to receive its own Directive Response Messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.DIR.APP4.*   or

C2MS.*.*.*.*.*.RESP.DIR.APP4.>
```

8.5.2.2 Directive Response Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Directive Response Message header.

Table 8-35. Directive Response Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	DIR

8.5.2.3 Directive Response Contents

The following figure shows a UML object diagram of the Directive Response Message, with its required and optional fields.

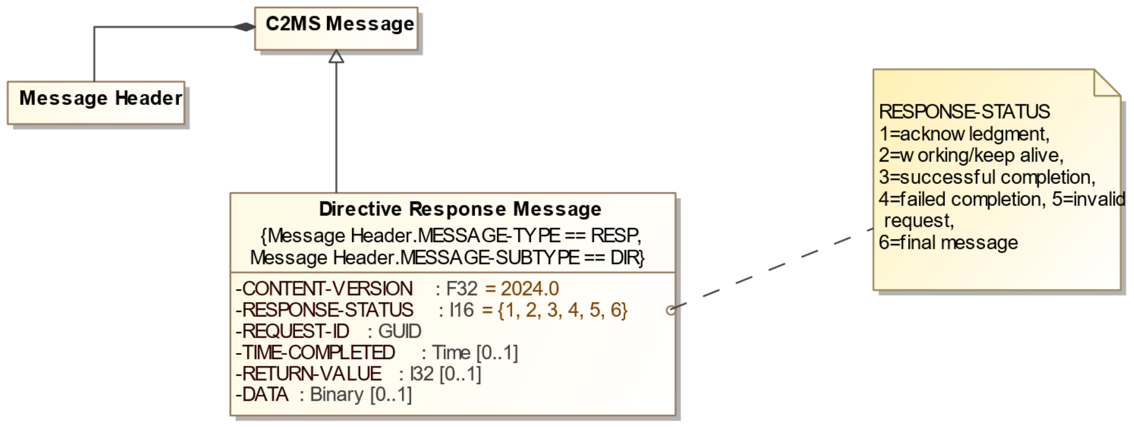

C2MS Message
Message Header
Directive Response Message
{Message Header.MESSAGE-TYPE == RESP,
Message Header.MESSAGE-SUBTYPE == DIR}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1, 2, 3, 4, 5, 6}
-REQUEST-ID : GUID
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-DATA : Binary [0..1]
RESPONSE-STATUS
1=acknow ledgment,
2=w orking/keep alive,
3=successful completion,
4=failed completion, 5=invalid request,
6=final message

Figure 8-7. Directive Response Diagram

The following table describes additional field names, values, and notes for the Directive Response Message.

Table 8-36. Directive Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the Directive being processed
			1	Acknowledgement	
			2	Working/keep alive	
			3	Successful completion	
			4	Failed completion	
			5	Invalid Request	
			6	Final Message	
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
TIME-COMPLETED	Time	O			Time application completed processing the directive
RETURN-VALUE	I32	O			Return value or status based on the RESPONSE-STATUS. Useful to provide function call status or error code in the case of failed completion
DATA	Binary	O			Additional data that may be desired along with the completion status

8.6 Component-to-Component Transfer (C2CX) Messages

The Component-to-Component Transfer (C2CX) Messages provide the ability to transfer control and status information between components. The messages are used for sending status, control, setup, initialization, heartbeat, security, a handshake, etc. from one component to another (one-to-one) or to multiple components (one-to-many).

The C2CX messages are typically unidirectional. If components need back and forth interaction or confirmation, it is recommended that the Directive Request and Response messages be used. Also, if it is desirable to know the progress or status of a C2CX message, the receiver of the message can issue a Log message at noteworthy points within its processing cycle. This means the C2CX messages act at a layer below general system knowledge that is accomplished through other messages such as the Log message and Directive messages. The C2CX messages seek to provide a communication mechanism just under the radar of the general system, but also easily viewable, as necessary.

8.6.1 Configuration Status Message

The **CFG – Configuration Status** C2CX message is used by software components to report their configuration information. (The **DEV – Device** C2CX message is used to report the status of devices.) Note that the Message Header already contains information about the component:

- Support of mission and satellites
- Name and location (facility and node)
- Class of capabilities provided

The additional information to be passed or reported is non-standard. The components reporting the configuration information and the components monitoring the configuration will need to establish:

- What configuration information is to be reported (and name the fields)
- When it is to be reported (upon change or periodically)
- What format to report the configuration information

A monitoring agent would collect Heartbeat messages and Configuration Status messages and possibly the Device messages to maintain a picture of what software is operating where, in what capacity, in what state, and with what physical and/or logical associations. The collected information can be:

- Presented in a graphical colored display depicting the operating environment and the logical associations of the components.
- Used to determine what pre-determined actions to take in the event of a failure or degraded operations.

As an example, a front-end telemetry and command processor would publish a CFG message whenever it is first run and thereafter when it associates (or disassociates) itself with another cooperating component. In addition to the information already provided in the Message Header, it would also report the following configuration information:

- The role of the reporting component (PRIMARY, BACKUP)
- Number of associations

  Name of component or port associated with, such as:

  - Telemetry and command processor (decommutation and command verification).
  - External telemetry and command link / port.
  - Planner and Scheduler - to direct the setup and operation of a pass.
  - Flight dynamics component - to exchange downlink or other information.

Node of the associated component, if known

Role of the associated component (PRIMARY, BACKUP)

If the monitoring agent is also a configuration manager, it can establish the present operating configuration and also prepare a contingent configuration. In the event of a component failure or processor failure, the configuration manager will know if it has the required and sufficient number of components to sustain operations. If not, it can also determine if it has the required and necessary numbers of components should a failover or restart procedure be invoked, and if so, automatically initiate that procedure.

The minimum and sometimes maximum configuration information a component can report is its own role. A single component with no associations would normally report its role as PRIMARY. Some configuration information may already be known and available if the components used a pre-registration or registration mechanism to disclose such information as:

- Nodes where they can execute.
- If they are standalone or redundant, and if redundant, on what nodes could the redundant component operate.

Furthermore, the Configuration Status Message may be used to report group associations. That is, to what group does this component belong, and is it a member and/or a manager of the group. Some groups of components may operate in a peer only association where other groups may require a group manager for organization, control, and direction.

Groups can be used for a number of purposes. These include, but are not limited to:

Message Exchange - groups can be formed to pass messages in a number of relationships and locations that include:

- Peer-to-Peer
- Client-Server
- Manager-Member
- Local and Distributed

Configuration Management - equipment can be logically associated to form groups (or suites or strings) that must operate together, failover together, have a minimum configuration (quorum), be addressed as a group, or other operating constraints or configurations.

Group formation can be pre-defined or dynamic. If dynamic, then additional group functions may be required, such as Create / Disband and Join / Leave.

Groups can also be hierarchical, as shown in the following table.

Table 8-37. Group Hierarchical Associations

Grouping Level	Space	Ground
Software	Software Application	Software Application
Hardware / Equipment	Processor	Computer
	Bus	Bus/LAN/WAN/Web
	Satellite	Facility/Center
	Constellation	Enterprise
Business	Mission	

The above discussion and illustrations provide a sampling of the ways groups may be employed within the messaging framework.

Table 8-38. Configuration Status Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Publish
Receiver	Any C2MS compliant application
Receivers Intended Usage	Subscribe
What	Status and Control type information
When	As needed and depends upon the type of information being transferred

Example

1. A component needs information about another component's software configuration information from a logical or relational perspective.

8.6.1.1 Configuration Status Message Subjects

Table 8-39. Configuration Status Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	MSG	CFG	[COMPONENT]	[DESTINATION-COMPONENT]	[DESTINATION-NODE]	[DESTINATION-FACILITY]	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	CFG	APP1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	CFG	CFGMGR	AGENT3			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	MSG	CFG	*	>			

Table 8-40. Properties of the Miscellaneous Elements for the Configuration Status Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	Component name of destination	DESTINATION-COMPONENT from header
ME3 *ME4*	Optional	Component destination: The two *miscellaneous elements* may be used to direct the message to a specific destination, as necessary, in Subject Token String format. *ME3* = DESTINATION-NODE *ME4* = DESTINATION-FACILITY	DESTINATION-NODE DESTINATION-FACILITY

Configuration status messages may be published for general consumption or they may be targeted to a central collector component. The second element, *ME2* (component of recipient), is used if necessary.

Examples

Publishing / Sending Configuration Status messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CFG.APP1 or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CFG.APP1.COMPONENT or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CFG.APP1.COMPONENT.DESTINATION-
NODE.DESTINATION-FACILITY
```

Subscribe to / receive a Configuration Status message:

```
C2MS.*.*.*.*.*.MSG.CFG.MYAGENT.CFGMGR.> or

C2MS.*.*.*.*.*.MSG.CFG.CFGMGR.>
```

8.6.1.2 Configuration Status Message Header

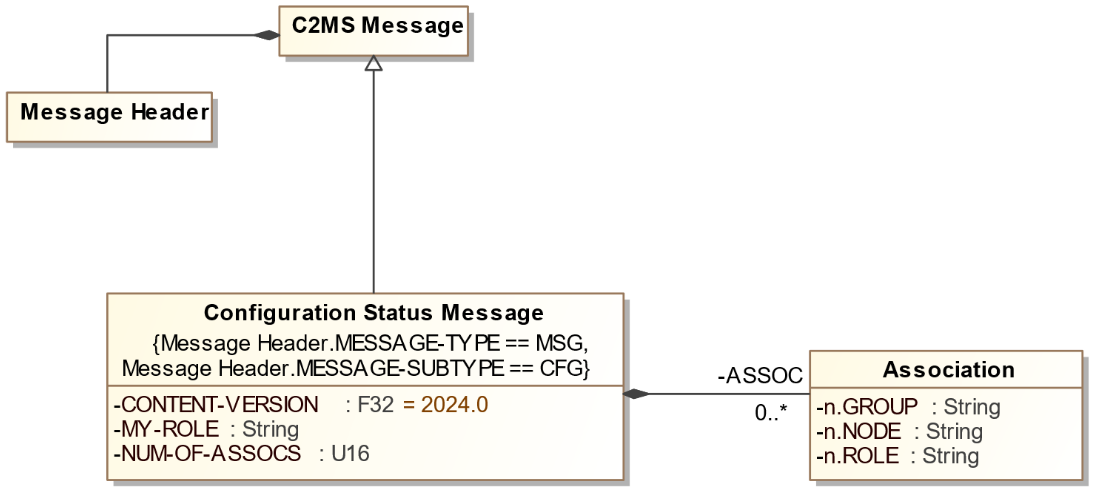
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Configuration Status Message header.

Table 8-41. Configuration Status Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	CFG

8.6.1.3 Configuration Status Message Contents

The following figure shows a UML object diagram of the Configuration Status Message with its required and optional fields.



C2MS Message
Message Header
Configuration Status Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == CFG}
-CONTENT-VERSION : F32 = 2024.0
-MY-ROLE : String
-NUM-OF-ASSOCS : U16
-ASSOC
0..*
Association
-n.GROUP : String
-n.NODE : String
-n.ROLE : String

Figure 8-8. Configuration Status Message Diagram

The following table describes additional field names, values, and notes for the Configuration Status Message.

Table 8-42. Configuration Status Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
MY-ROLE	String	R		Role the reporting component has in the configuration. E.g. PRIMARY, BACKUP, AGENT, SERVER, MEMBER, MGR, …
NUM-OF-ASSOCS	U16	R		The number of associations to be reported.
ASSOC.n.GROUP	String	O		Name of component or group associated with - “n” starts at “1”.
ASSOC.n.NODE	String	O		Location of associated component or group
ASSOC.n.ROLE	String	O		Role the associated component has, if known. E.g. PRIMARY, BACKUP, AGENT, SERVER, MEMBER, MGR, …

8.6.2 Control Message

The Control messages are those that are typically used to request “under the hood” functions that, while essential for operations, are not of general interest.

The **Control** message is used to start, restart, reinitialize, or otherwise control another component. As an example, components may determine that they will not proceed with their processing until they have received a Control message with a CNTL-STRING of “INIT” or “START”. Other components may require additional information in CNTL-STRING to begin processing. Still other components that have been performing their processing may allow themselves to be re-directed in their processing. Upon the reception of a Control message, a component will re-direct itself according to the supplied string of parameters.

For instance, a component could request another component to change its rate of publishing a Heartbeat message. The requesting component would send a Control message with a known decipherable command string in the CNTL-STRING field; for example: “SET HB 15”.

Missions may want to have different processing modes or signals when the course of events changes what standard actions are to follow. For example, a system wide indicator may be sent using the CNTL messages with a value for the CNTL-STRING to signify that a pass has begun and the processing mode is ‘PASS’. Or, the processing mode is ‘PRE-PASS’, ‘POST-PASS’, ‘LIGHTS-OUT’, ‘LIGHTS-ON’. ‘AUTONOMOUS’, ‘SIMULATION’, ‘LAUNCH’, ‘ECLIPSE-PERIOD’, ‘MANEUVER’, ‘SAFE-MODE’, or any such state that could affect some components and result in conditional processing or decision making.

In these examples, a monitoring agent, criteria action agent, decision making component, script controller, or processing manager would monitor the events for state changes and then issue the CNTL message for all or a subset of the components.

A further example could involve distributed simulations. A key factor in these simulations is to know the simulated time. A CNTL message can be defined to set, distribute, or synchronize components to a simulated time. The CNTL message might be used to set the time or advance the simulated time by a delta time. This includes training, development, integration, and pre-launch / operations simulations. If necessary, a separate C2CX message may be developed with a called SETTIME with associated parameters.

Finally, a simple application could be a PING function. “PING” placed in the CNTL-STRING field would simply require the receiver to publish the same type of message but with “PING-ACK” in the CNTL-STRING field. Alternately, separate C2CX messages could be defined for the PING and PING-ACK functionality.

Table 8-43. Control Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Publish
Receiver	Any C2MS compliant application
Receivers Intended Usage	Subscribe
What	Status and Control type information
When	As needed and depends upon the type of information being transferred

Example:

1. A component can request another component to action with the CNTL message.

8.6.2.1 Control Message Subjects

Table 8-44. Control Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	CNTL	[COMPONE NT]	[DESTINATI ON-COMPONE NT]	[DESTINATION-NODE]	[DESTINAT ION-FACILITY]	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	CNTL	APP1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	CNTL	CFGMGR	AGENT3			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	MSG	CNTL	*	>			

Table 8-45. Properties of the *Miscellaneous Elements* for the Control Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	Component name of destination	DESTINATION-COMPONENT from header
ME3 *ME4*	Optional	Component destination: The two *miscellaneous elements* may be used to direct the message to a specific destination, as necessary, in Subject Token String format. *ME3* = DESTINATION-NODE *ME4* = DESTINATION-FACILITY	DESTINATION-NODE DESTINATION-FACILITY
ME5	Optional	A keyword the receiving process could use for filtering	"CNTL-KEYWORD" from msg content

Control messages may be published for general consumption or they may be targeted to a central collector component. The second element, *ME2* (component of recipient), is used if necessary.

Examples

Publishing / Sending Control messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CNTL.APP1 or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CNTL.APP1.COMPONENT or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CNTL.APP1.COMPONENT.DESTINATION-
NODE.DESTINATION-FACILITY or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CNTL.APP1.COMPONENT.DESTINATION-
NODE.DESTINATION-FACILITY.KEYWORD
```

Subscribe to / receive a Control message:

```
C2MS.*.*.*.*.*.MSG.CNTL.MYAGENT.CFGMGR.> or

C2MS.*.*.*.*.*.MSG.CNTL.CFGMGR.>
```

8.6.2.2 Control Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Control Message header.

Table 8-46. Control Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	CNTL

8.6.2.3 Control Message Contents

The following figure shows a UML object diagram of the Control Message with its required and optional fields.

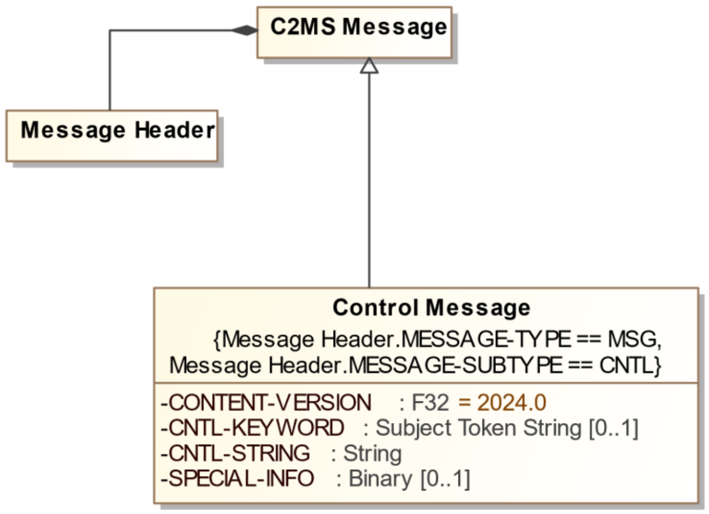

C2MS Message
Message Header
Control Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == CNTL}
-CONTENT-VERSION : F32 = 2024.0
-CNTL-KEYWORD : Subject Token String [0..1]
-CNTL-STRING : String
-SPECIAL-INFO : Binary [0..1]

Figure 8-9. Control Message Diagram

The following table describes additional field names, values, and notes for the Control Message.

Table 8-47. Control Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
CNTL-KEYWORD	Subject Token String	O		Keyword extracted from the CNTL-STRING. Useful for routing/processing.
CNTL-STRING	String	R		Parameters to guide the component on further processing. E.g., INIT, Stop, Shutdown, Restart, Do X, Y, and Z.
SPECIAL-INFO	Binary	O		For application use. Any additional information can be provided here.

8.6.3 Device Message

The **Device** C2CX message is used to report the status of devices, physical or virtual. (The Heartbeat message and the Configuration Status message are used to report status on software components.) The Device message would typically be used to report the status of devices that would not be capable of reporting themselves. For example, a software component may interact with a specialized device or merely have access to the status of devices operating in the same environment. A designated software component would gather the status of the device(s) and publish the information with the Device C2CX message. This message is not intended to communicate with the device.

Thus, in conjunction with the Configuration Status and Heartbeat messages, a full story on the configuration can be gathered for reporting and subsequent actions when a system-wide re-configuration is implemented.

Of course, this message does not need to be restricted to physical devices. Virtual devices may be constructed and reported on as well. A physical device may be logically partitioned, or a logical device may be spread over a number of physical devices. Or, a virtual (or pseudo) device could be constructed or defined with no relation to any physical device. For example, a set of parameters, somehow related, could be grouped as a “device” and reported on for display and monitoring. A single reporting agent could be responsible for a virtual device and report on it. Or, a number of agents could report on separate parameters and the collector of the Device messages could effectively construct a virtual device from the disparate information. A set of key or critical parameters could be constructed and reported on using this method.

A hypothetical example for a communications data path could consist of a ground antenna, a ground station processor/controller, a data link, and a front-end processor. Together these devices could constitute a virtual data link device whose individual device statuses are collected (and logically ANDed together) to provide a GO/NOGO or rollup Normal, Warning, Distress or Critical status on the data link.

Table 8-48. Device Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Publish
Receiver	Any C2MS compliant application
Receivers Intended Usage	Subscribe
What	Status and Control type information
When	As needed and depends upon the type of information being transferred

Example

1. A component reports its status, physical or virtual, or any collection of data.

8.6.3.1 Device Message Subjects

Table 8-49. Device Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	DEV	[COMPONE NT]	[DESTINATI ON-COMPONE NT]	[DESTINATION-NODE]	[DESTINATI ON-FACILITY]	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	DEV	APP1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	DEV	CFGMGR	AGENT3			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	MSG	DEV	*	>			

Table 8-50. Properties of the *Miscellaneous Elements* for the Device Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	Component name of destination	DESTINATION-COMPONENT from header
ME3 *ME4*	Optional	Component destination: The two *miscellaneous elements* may be used to direct the message to a specific destination, as necessary, in Subject Token String format. *ME3* = DESTINATION-NODE *ME4* = DESTINATION-FACILITY	DESTINATION-NODE DESTINATION-FACILITY

Device messages may be published for general consumption or they may be targeted to a central collector component. The second element, *ME2* (component of recipient), is used if necessary.

Examples

Publishing / Sending Device messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.DEV.APP1 or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.DEV.APP1.COMPONENT or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.DEV.APP1.COMPONENT.DESTINATION-
NODE.DESTINATION-FACILITY
```

Subscribe to / receive a Device message:

```
C2MS.*.*.*.*.*.MSG.DEV.MYAGENT.CFGMGR.> or

C2MS.*.*.*.*.*.MSG.DEV.CFGMGR.>
```

8.6.3.2 Device Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Device Message header.

Table 8-51. Device Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	DEV

8.6.3.3 Device Message Contents

The following figure shows a UML object diagram of the Device Message with its required and optional fields.

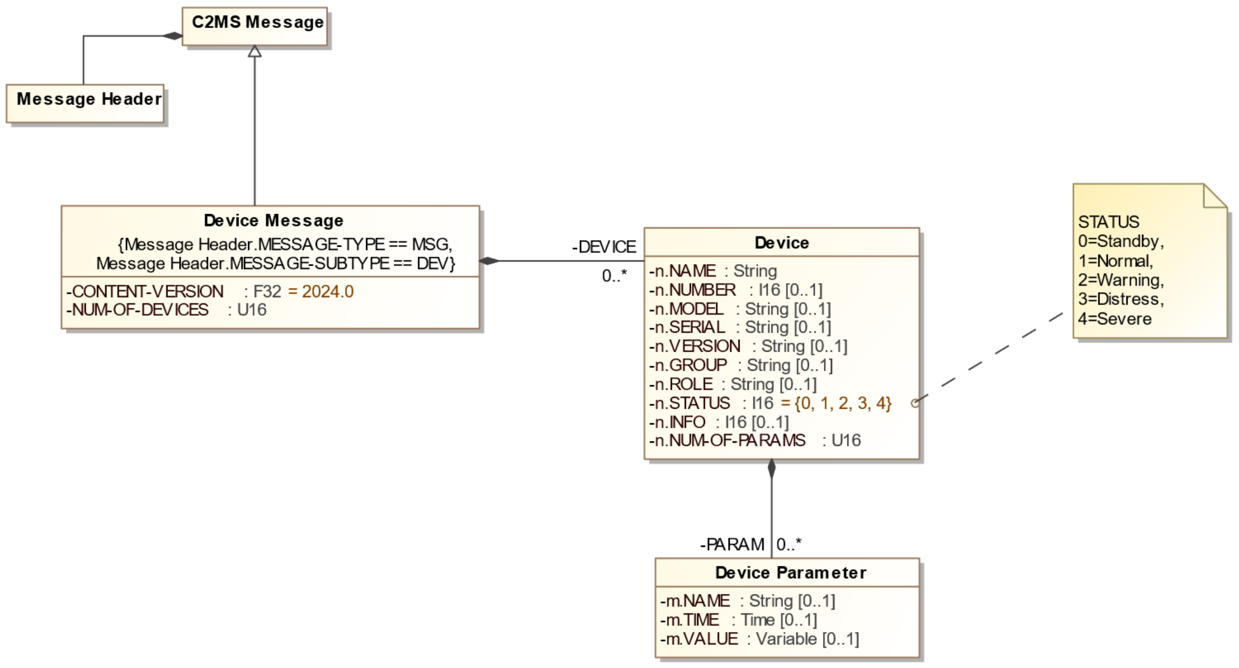


Figure 8-10. Device Message Diagram

The following table describes additional field names, values, and notes for the Device Message.

Table 8-52. Device Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
NUM-OF-DEVICES	U16	R	1+		Number of devices being reported in this message
DEVICE.n.NAME	String	R			Name of the device - "n" starts at "1". This field is required for each device specified by NUM-OF-DEVICES.
DEVICE.n.NUMBER	I16	O			Number assigned to the device to distinguish it from identical devices.
DEVICE.n.MODEL	String	O			Model number of the device
DEVICE.n.SERIAL	String	O			Serial number of the device
DEVICE.n.VERSION	String	O			Version of the firmware operating within the device
DEVICE.n.GROUP	String	O			Name of group with which device is associated
DEVICE.n.ROLE	String	O			Role of the device, if known. E.g. PRIMARY, BACKUP, AGENT, SERVER, MEMBER, MGR, …
DEVICE.n.STATUS	I16	D	**Value**	**Description**	Condition of the device being reported. The criteria for selecting the DEVICE.n.STATUS
			0	Standby	

Field Name	Type	Presence	Value		Description
			1	Normal	description is left to the reporting component. This field is required for each device specified by NUM-OF-DEVICES.
			2	Warning	
			3	Distress	
			4	Critical	
DEVICE.n.INFO	I16	O			An additional status code that can be supplied that is specific to that device
DEVICE.n.NUM-OF-PARAMS	U16	D			Number of additional parameters being reported that are associated with the device. This field is required for each device specified by NUM-OF-DEVICES.
DEVICE.n.PARAM.m.NAME	String	O			Name of the additional parameter
DEVICE.n.PARAM.m.TIME	Time	O			Time of parameter sampling
DEVICE.n.PARAM.m.VALUE	Variable	O			Value of the named parameter being reported

8.6.4 Heartbeat Message

The Heartbeat message is used to notify other components that the sending / publishing component is alive or active. Other components monitoring the Heartbeat messages can determine what action to take, if any, when a Heartbeat message fails to appear as scheduled, or if the COMPONENT-STATUS is not normal/green. If the component does not publish the heartbeat at the default rate, it can supply the publishing rate (PUB-RATE field) and a counter (COUNTER field) for a monitor to calculate when a heartbeat is expected or might be missing or late. Each component or system or mission can determine its own preferred heartbeat rate.

Not Using the COMPONENT-STATUS Field

If the COMPONENT-STATUS field is not to be used, then if the component is running but not 100%, then a component should cease publishing its Heartbeat message. The termination of the Heartbeat message will then make auto/re-configuration options possible. Which is to say, if the component is either 100% or 0%, or those are the only two states the component can report, then using the COMPONENT-STATUS field is not necessary, as long as the component can cease publishing the Heartbeat message in circumstances when it knows it is not 100%.

Using the COMPONENT-STATUS Field

A component may typically only supply the COMPONENT-STATUS of 1 – Normal/Green. However, when the status of the component is less than normal / green (100%), say yellow (75%), indicating a less than optimal operating state, it may also supply a status code in the COMPONENT-INFO field. This code would only have context within that component. A component may also issue a Log Message in conjunction with a change in COMPONENT-STATUS. The component would include the COMPONENT-INFO value in the subsequent Log Message so the Heartbeat message and the Log Message could be cross-referenced. Components can self-determine what constitutes a Normal, Warning, Distress, or Critical state of processing. A component that ceases to send a heartbeat message will be presumed to be absent and in a Critical condition. If applicable, a component will then be susceptible to a pre-determined recovery action, including failover and restart. A monitoring agent can use the COMPONENT-STATUS value to color code a display of the component's status.

Table 8-53. Heartbeat Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Publish
Receiver	Any C2MS compliant application

Receivers Intended Usage	Subscribe
What	Status and Control type information
When	As needed and depends upon the type of information being transferred

Example

1. All active components publish a heartbeat (aka keep-alive); a monitoring component checks on the ongoing presence of the components and detects their absence.

8.6.4.1 Heartbeat Message Subjects

Table 8-54. Heartbeat Message Subject Naming

	Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	MSG	HB	[COMPONENT]	[DESTINATION-COMPONENT]	[DESTINATION-NODE]	[DESTINATION-FACILITY]	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	HB	APP1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	HB	CFGMGR	AGENT3			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	MSG	HB	*	>			

Table 8-55. Properties of the *Miscellaneous Elements* for the Heartbeat Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	Component name of destination	DESTINATION-COMPONENT from header
ME3 *ME4*	Optional	Component destination: The two *miscellaneous elements* may be used to direct the message to a specific destination, as necessary, in Subject Token String format. *ME3* = DESTINATION-NODE *ME4* = DESTINATION-FACILITY	DESTINATION-NODE DESTINATION-FACILITY

Heartbeat messages may be published for general consumption or they may be targeted to a central collector component. The second element, *ME2* (component of recipient), is used if necessary.

Examples

Publishing / Sending Heartbeat messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.HB.APP1 or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.HB.APP1.COMPONENT or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.HB.APP1.COMPONENT.DESTINATION-
NODE.DESTINATION-FACILITY
```

Subscribe to / receive a Heartbeat message:

```
C2MS.*.*.*.*.*.MSG.HB.MYAGENT.CFGMGR.> or

C2MS.*.*.*.*.*.MSG.HB.CFGMGR.>
```

8.6.4.2 Heartbeat Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Heartbeat Message header.

Table 8-56. Heartbeat Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	HB

8.6.4.3 Heartbeat Message Contents

The following figure shows a UML object diagram of the Heartbeat Message with its required, optional, and tracking fields.

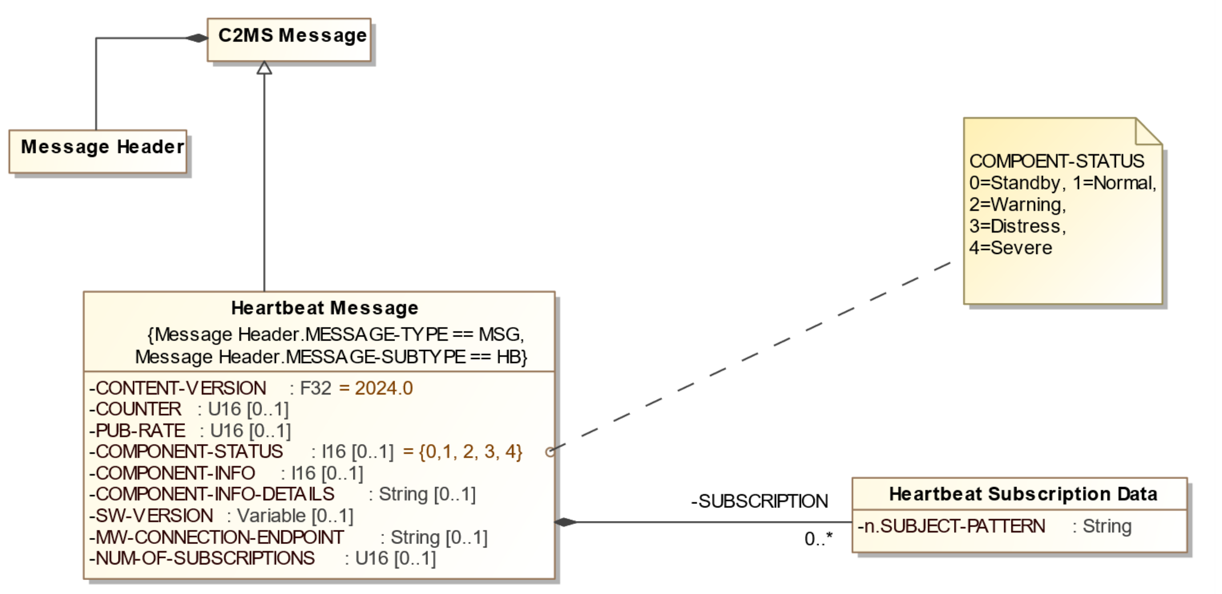

C2MS Message
Message Header
COMPOENT-STATUS
0=Standby, 1=Normal,
2=Warning,
3=Distress,
4=Severe
Heartbeat Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == HB}
-CONTENT-VERSION : F32 = 2024.0
-COUNTER : U16 [0..1]
-PUB-RATE : U16 [0..1]
-COMPONENT-STATUS : I16 [0..1] = {0,1, 2, 3, 4}
-COMPONENT-INFO : I16 [0..1]
-COMPONENT-INFO-DETAILS : String [0..1]
-SW-VERSION : Variable [0..1]
-MW-CONNECTION-ENDPOINT : String [0..1]
-NUM-OF-SUBSCRIPTIONS : U16 [0..1]
-SUBSCRIPTION
0..*
Heartbeat Subscription Data
-n.SUBJECT-PATTERN : String

Figure 8-11. Heartbeat Message Diagram

The following table describes additional field names, values, and notes for the Heartbeat Message.

Table 8-57. Heartbeat Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
COUNTER*	U16	O	1+		Indicates the number of times that the C2CX heartbeat message has been published, including this message.
PUB-RATE	U16	O			Indicates the rate, in number of seconds, which the C2CX heartbeat message is being published by the component. A rate of zero or less indicates that this C2CX message is not repeatedly published by the component. The default publishing rate of the C2CX heartbeat message is 30 seconds.
COMPONENT-STATUS	I16	O	**Value**	**Description**	Indicates the condition of the component being monitored, typically itself, although it may be a proxy for a remote component. The component may choose the condition level based on its own criteria.
			0	Standby	
			1	Normal	
			2	Warning	
			3	Distress	
			4	Critical	
COMPONENT-INFO	I16	O			An additional status code the component can supply that is specific to that component.
COMPONENT-INFO-DETAILS	String	O			Allows a component to detail its status in a verbose message

Field Name	Type	Presence	Value	Description
SW-VERSION	Variable	O		Version number identifier of the reporting component. Component must ascertain the data type before accessing the value (e.g. with a function call).
MW-CONNECTION-ENDPOINT	String	O		**Tracking field -** Broker(s) to which the client application is currently connected - reserved for use by implementation (PSM)
NUM-OF-SUBSCRIPTIONS	U16	O		**Tracking field -** The number of active subscriptions set up across all connections held by the running application - reserved for use by implementation (PSM)
SUBSCRIPTION.n.SUBJECT-PATTERN	String	D		**Tracking field -** The nth subscription subject pattern held by the running application - "n" starts at "1"- reserved for use by implementation (PSM) . This field is required for each subscription when NUM-OF-SUBSCRIPTIONS > 0.

* Note: At a rate of two messages per minute, this counter will overflow after 22 days.

8.6.5 Resource Message

The C2CX **Resource** message is used to publish computer performance data. Resource data is organized per CPU, disk, and network port. It is intended that the data be a snapshot of the resources at the time of collection and not a cumulative summary. The snapshot of the resources can be paired with the time of publication of the message to produce a data point. After the collection of a number of data points, a trend /plot of the resources can be established.

All resource information has been marked as optional so that a component may provide any or all of the resource information as necessary. For example, an agent collecting and publishing data may determine to publish the CPU resources at a difference rate than the disk resources. Resource messages for CPUs might be published every 60 seconds, while disk Resource messages might be published every 300 seconds.

If a component is to be controlled or directed as to the frequency of resource publishing, it could use the CNTL message with a CNTL-STRING of "`SET RSRC CPU 60`" to set the CPU resources publication rate at 60 seconds, or disk resources at 300 seconds with "`SET RSRC DISK 300`".

Table 8-58. Resource Message Summary

Sender	Any C2MS compliant application
Senders Intended Usage	Publish
Receiver	Any C2MS compliant application
Receivers Intended Usage	Subscribe
What	Status and Control type information
When	As needed and depends upon the type of information being transferred

Example

1. This message is used to report a snapshot of computer performance data (CPU, memory, disk, and network usage).

8.6.5.1 Resource Message Subjects

Table 8-59. Resource Message Subject Naming

	Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	RSRC	[COMPONEN T]	[DESTINATION-COMPONENT]	[DESTINATI ON-NODE]	[DESTINA TION-FACILITY]	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	RSRC	APP1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	RSRC	CFGMGR	AGENT3			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	MSG	RSRC	*	>			

Table 8-60. Properties of the *Miscellaneous Elements* for the Resource Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	Component name of destination	DESTINATION-COMPONENT from header
ME3 *ME4*	Optional	Component destination: The two *miscellaneous elements* may be used to direct the message to a specific destination, as necessary, in Subject Token String format. *ME2* = destination component or group *ME3* = DESTINATION-NODE *ME4* = DESTINATION-FACILITY	DESTINATION-NODE DESTINATION-FACILITY

Resource messages may be published for general consumption or they may be targeted to a central collector component. The second element, *ME2* (component of recipient), is used if necessary.

Examples

Publishing / Sending Resource messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.RSRC.APP1 or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.RSRC.APP1.COMPONENT or
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.DEV.APP1.COMPONENT.DESTINATION-
NODE.DESTINATION-FACILITY
```

Subscribe to / receive a Resource message:

```
C2MS.*.*.*.*.*.MSG.RSRC.MYAGENT.CFGMGR.> or

C2MS.*.*.*.*.*.MSG.RSRC.CFGMGR.>
```

8.6.5.2 Resource Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Resource Message header.

Table 8-61. Resource Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	RSRC

8.6.5.3 Resource Message Contents

The following figure shows a UML object diagram of the Resource Message with its required and optional fields.

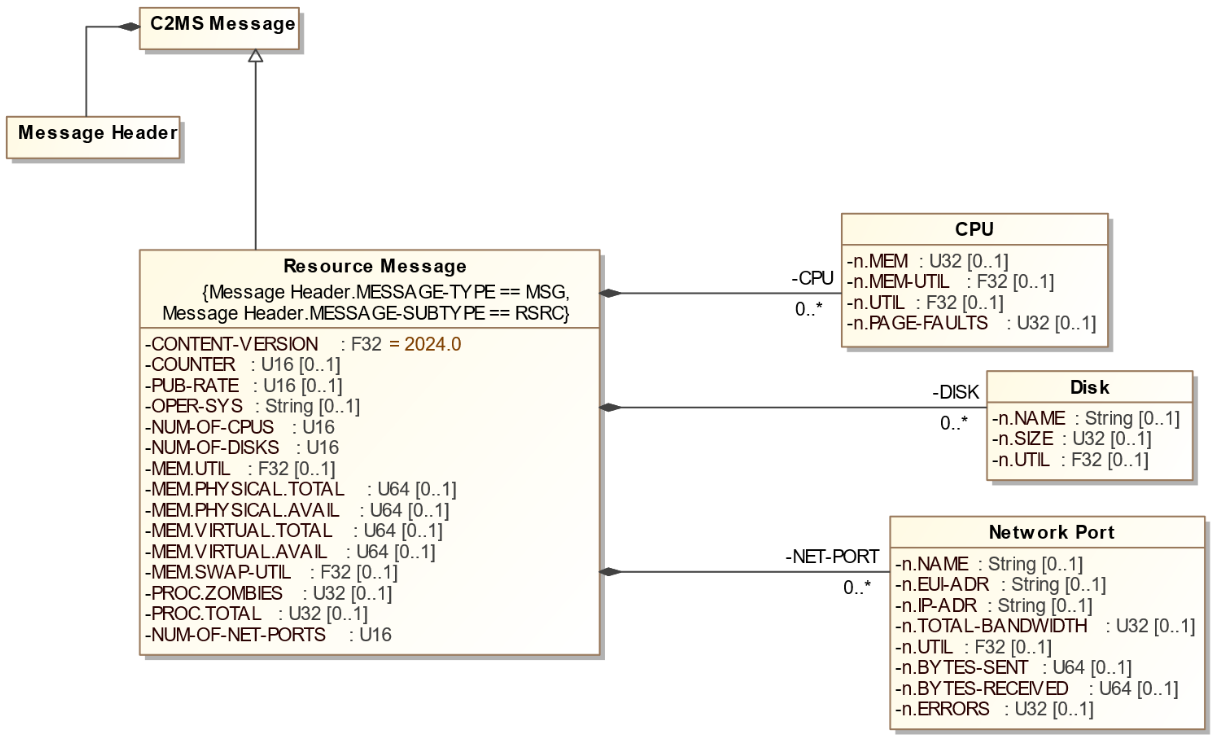

C2MS Message
Message Header
Resource Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == RSRC}
-CONTENT-VERSION : F32 = 2024.0
-COUNTER : U16 [0..1]
-PUB-RATE : U16 [0..1]
-OPER-SYS : String [0..1]
-NUM-OF-CPUS : U16
-NUM-OF-DISKS : U16
-MEM.UTIL : F32 [0..1]
-MEM.PHYSICAL.TOTAL : U64 [0..1]
-MEM.PHYSICAL.AVAIL : U64 [0..1]
-MEM.VIRTUAL.TOTAL : U64 [0..1]
-MEM.VIRTUAL.AVAIL : U64 [0..1]
-MEM.SWAP-UTIL : F32 [0..1]
-PROC.ZOMBIES : U32 [0..1]
-PROC.TOTAL : U32 [0..1]
-NUM-OF-NET-PORTS : U16
-CPU
0..*
CPU
-n.MEM : U32 [0..1]
-n.MEM-UTIL : F32 [0..1]
-n.UTIL : F32 [0..1]
-n.PAGE-FAULTS : U32 [0..1]
-DISK
0..*
Disk
-n.NAME : String [0..1]
-n.SIZE : U32 [0..1]
-n.UTIL : F32 [0..1]
-NET-PORT
0..*
Network Port
-n.NAME : String [0..1]
-n.EUI-ADR : String [0..1]
-n.IP-ADR : String [0..1]
-n.TOTAL-BANDWIDTH : U32 [0..1]
-n.UTIL : F32 [0..1]
-n.BYTES-SENT : U64 [0..1]
-n.BYTES-RECEIVED : U64 [0..1]
-n.ERRORS : U32 [0..1]

Figure 8-12. Resource Message Diagram

The following table describes additional field names, values, and notes for the Resource Message.

Table 8-62. Resource Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
COUNTER*	U16	O	1+	Indicates the number of times that the C2CX Resource message has been published, including this message.
PUB-RATE	U16	O	Seconds	Rate the data is being collected and published. The default publishing rate is 30 seconds. A rate of zero indicates this message is not being repeatedly published.
OPER-SYS	String	O		Operating system component is using
NUM-OF-CPUS	U16	R	1+	Number of CPUs being monitored
CPU.n.MEM	U32	O	In megabytes	Amount of memory for this CPU - “n” starts at “1”.
CPU.n.MEM-UTIL	F32	O	0-100	Memory utilization. Percentage of memory utilized.
CPU.n.UTIL	F32	O	0-100	CPU utilization. Percentage of CPU utilized
CPU.n.PAGE-FAULTS	U32	O	1+	Number of page faults
NUM-OF-DISKS			0+	Number of disks being monitored

Field Name	Type	Presence	Value	Description
DISK.n.NAME	String	O		Name of the disk - “n” starts at “1”.
DISK.n.SIZE	U32	O	In megabytes	Absolute size of the disk
DISK.n.UTIL	F32	O	0-100	Disk space utilization. Percentage of Disk space utilized.
MEM.UTIL	F32	O	0-100	Percent of main memory utilized
MEM.PHYSICAL.TOTAL	U64	O	1+	Total amount of physical memory present, in bytes
MEM.PHYSICAL.AVAIL	U64	O	0+	Total amount of physical memory available, in bytes
MEM.VIRTUAL.TOTAL	U64	O	1+	Total amount of virtual memory present, in bytes
MEM.VIRTUAL.AVAIL	U64	O	0+	Total amount of virtual memory available, in bytes
MEM.SWAP-UTIL	F32	O	0-100	Percent of swap space used
PROC.ZOMBIES	U32	O	0+	Number of zombie processes
PROC.TOTAL	U32	O	0+	Number of total processes
NUM-OF-NET-PORTS	U16	R	1+	Number of network ports
NET-PORT.n.NAME	String	O		Name of the network port - “n” starts at “1”.
NET-PORT.n.EUI-ADR	String	O	Format of: 01-23-45-67-89-ab or 01:23:45:67:89:ab	Media Access Control (MAC) or Extended Unique Identifier (EUI) physical address. MAC-48, EUI-48, or EUI-64 format.
NET-PORT.n.IP-ADR	String	O	208.77.188.166 or 2001:0db8:85a3:08d3:1319:8a2e:0370:7334	Internet Protocol (IP) logical address. IPv4 (32-bit) or IPv6 (128-bit) format.
NET-PORT.n.TOTAL-BANDWIDTH	U32	O	0+	Bandwidth of the port in Kbps
NET-PORT.n.UTIL	F32	O	0-100	Percentage of Network port utilization
NET-PORT.n.BYTES-SENT	U64	O	0+	Number of bytes sent over the port
NET-PORT.n.BYTES-RECEIVED	U64	O	0+	Number of bytes received over the port
NET-PORT.n.ERRORS	U32	O	0+	Number of errors encountered on the port
* Note: At a rate of 2 messages per minute, this counter will overflow after 22 days.				

8.7 Real-time Telemetry Data Messages

Telemetry Messages are data packages that contain spacecraft health and safety data. In most ground systems, a Telemetry Message is packaged by the spacecraft and sent to the ground station. The ground station performs air-to-ground quality checking, adds ground station information, and routes the data to the ground system. Archive retrieval systems, simulators, and data generators can also provide Telemetry Messages in replay, simulation, and test modes.

Additionally, the data can be published “as is” (Raw), or after a degree or level of processing (Processed). The latter could involve a number of data scrubbing techniques plus conversion from binary values to engineering units (EU). The following table lists the messages that have been defined to transport the various kinds of telemetry data.

Table 8-63. Telemetry Messages

Telemetry Message	Data Form (Level)	Data Format
Telemetry CCSDS Packet	Raw	CCSDS Packet

Telemetry Message	Data Form (Level)	Data Format
DEPRECATED Telemetry CCSDS Frame	Raw	CCSDS Frame
Telemetry CCSDS CADU Frame	Raw	CCSDS Channel Access Data Unit (CADU) Frame
Telemetry CCSDS Transfer Frame	Raw	CCSDS Transfer Frame
Telemetry TDM Frame	Raw	TDM Frame
Processed Telemetry Data	Processed (Converted)	Data samples for one frame organized by mnemonic

The CCSDS CADU Frame, CCSDS Transfer Frame, and CCSDS Packet are industry standard formats. Time Division Multiplexing (TDM) is the method and format for sending multiple digital signals along a single telecommunications transmission. Specific decommutation instructions for the frames and packets are documented in other resources.

Note: Additional telemetry message contents may be added as necessary.

8.7.1 Telemetry CCSDS Packet Message

The Telemetry CCSDS Packet Message is used for transferring CCSDS telemetry packets. The Telemetry Message Contents consists of the raw CCSDS telemetry packet, the time of the packet and the quality of the data.

Table 8-64. Telemetry CCSDS Packet Message Summary

Sender	A C2MS compliant application such as a ground station, simulator, archive component, or front-end processor
Senders Intended Usage	Publish
Receiver	Telemetry Decommutation System, Archive System, Trending System, Expert System
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data to be decommutated and/ or archived
When	As needed but usually dependent on data rate and/or replay rate

Example

- Spacecraft health and safety data sent from ground station to ground system

8.7.1.1 Telemetry CCSDS Packet Message Subjects

Table 8-65. Telemetry CCSDS Packet Message Subject Naming

	Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	MSG	TLMPKT	[COMPONENT]	Stream-mode	Virtual Channel ID	APID	Collection Point	Spacecraft ID (SCID)
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMPKT	SATSIM	SIM	2	1	GEP1	4
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMPKT	TFEP	RT	1	2	*	5
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	TLMPKT	*	RT	*	*	*	6

Table 8-66. Properties of the *Miscellaneous Elements* for the Telemetry CCSDS Packet Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field in Msg, if applicable**
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Required	Virtual Channel ID	VCID from msg content; or from header portion of CCSDS frame
ME4	Required	APID – identifies a particular subsystem on the spacecraft	From header portion of data stream
ME5	Optional	Point on ground system where data was captured	COLLECTION-POINT
ME6	Optional	Spacecraft ID	SCID from msg content or from header portion of CCSDS packet

Example for Publisher / Sender of Telemetry Messages

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMPKT.SATSIM.SIM.2.1
```

Example for Subscriber / Receiver of Telemetry Messages

```
C2MS.*.*.MSSN.*.*.MSG.TLMPKT.TFEP.RT.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMPKT.TFEP.RT.2.1
```

8.7.1.2 Telemetry CCSDS Packet Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for Telemetry CCSDS Packet Message header.

Table 8-67. Telemetry CCSDS Packet Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	TLMPKT

8.7.1.3 Telemetry CCSDS Packet Message Contents

The following figure shows a UML object diagram of the Telemetry CCSDS Packet Message with its required and optional fields.

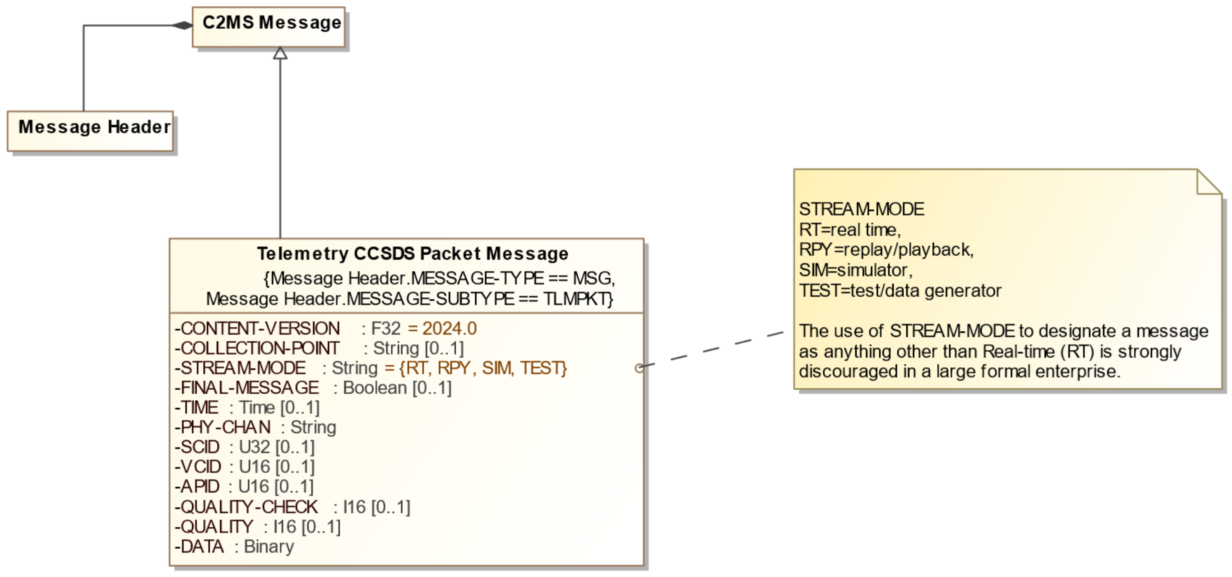

C2MS Message
Message Header
Telemetry CCSDS Packet Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == TLMPKT}
-CONTENT-VERSION : F32 = 2024.0
-COLLECTION-POINT : String [0..1]
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-FINAL-MESSAGE : Boolean [0..1]
-TIME : Time [0..1]
-PHY-CHAN : String
-SCID : U32 [0..1]
-VCID : U16 [0..1]
-APID : U16 [0..1]
-QUALITY-CHECK : I16 [0..1]
-QUALITY : I16 [0..1]
-DATA : Binary
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-13. Telemetry CCSDS Packet Message Diagram

The following table describes additional field names, values, and notes for the Telemetry CCSDS Packet Message.

Table 8-68. Telemetry CCSDS Packet Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
STREAM-MODE	String	R	**Value**	**Description**	Identifies the mode of the stream of telemetry as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
TIME	Time	O			Time of packet, usually ground receipt time
PHY-CHAN	String	R			Physical channel on which data is received
SCID	U32	O			CCSDS Spacecraft Identifier (SCID). This field originates from the CCSDS Transfer Frame and is not part of the CCSDS Packet.

Field Name	Type	Presence	Value		Description
VCID	U16	O			CCSDS Virtual Channel Identifier (VCID). This field originates from the CCSDS Transfer Frame and is not part of the CCSDS Packet.
APID	U16	O			CCSDS Application Process Identifier (APID)
QUALITY-CHECK	I16	O	**Value**	**Description**	Indicates quality checking was performed for reason indicated.
			Bit 0	Partial Packet	
QUALITY	I16	O	**Value**	**Description**	Indicates quality state if checking was performed
			Bit 0	Partial Packet	
DATA	Binary	R			Raw telemetry data

8.7.2 Telemetry CCSDS Frame Message (DEPRECATED)

The Telemetry CCSDS Frame Message has been deprecated and will be removed in a future release of C2MS. The new CCSDS Frame Messages, Telemetry CCSDS CADU Frame Message and Telemetry CCSDS Transfer Frame Message, should be used instead.

The Telemetry CCSDS Frame Message is used to transfer CCSDS telemetry frames. The Telemetry Message Contents consists of the raw CCSDS telemetry frame, the time of the frame, quality checking performed, and the resulting quality of the data.

A frame with a Frame Sync pattern at the front that includes Reed-Solomon check symbols is called a Coded Virtual Channel Data Unit (CVCDU) in the CCSDS documentation.

Table 8-69. Telemetry CCSDS Frame Message Summary

Sender	A C2MS compliant application such as a ground station, simulator, archive component, or front-end processor
Senders Intended Usage	Publish
Receiver	Telemetry Decommutation System, Archive System, Trending System, Expert System
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data to be decommutated and/ or archived
When	As needed but usually dependent on data rate and/or replay rate

Example

- Spacecraft health and safety data sent from ground station to ground system

8.7.2.1 Telemetry CCSDS Frame Message Subjects

Table 8-70. Telemetry CCSDS Frame Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	TLMFRAME	[COMPONENT]	Stream-mode	Virtual Channel ID	APID	Collection Point
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMFRAME	SATSIM	SIM	2	1	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMFRAME	TFEP	RT	1	2	
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	TLMFRAME	*	RT	*	*	

Table 8-71. Properties of the *Miscellaneous Elements* for the Telemetry CCSDS Frame Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Required	Virtual Channel ID	VCID from msg content; or from header portion of CCSDS frame
ME4	Optional	APID – identifies a particular subsystem on the spacecraft	From header portion of data stream
ME5	Optional	Point on ground system where data was captured	COLLECTION-POINT

Example for Publisher / Sender of Telemetry Messages

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMFRAME.SATSIM.SIM.2.1
```

Example for Subscriber / Receiver of Telemetry Messages

```
C2MS.*.*.MSSN.*.*.MSG.TLMFRAME.TFEP.RT.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMFRAME.TFEP.RT.2.1
```

8.7.2.2 Telemetry CCSDS Frame Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for Telemetry CCSDS Frame Message header.

Table 8-72. Telemetry CCSDS Frame Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	TLMFRAME

8.7.2.3 Telemetry CCSDS Frame Message Contents

The following figure shows a UML object diagram of the Telemetry CCSDS Frame Message with its required and optional fields.

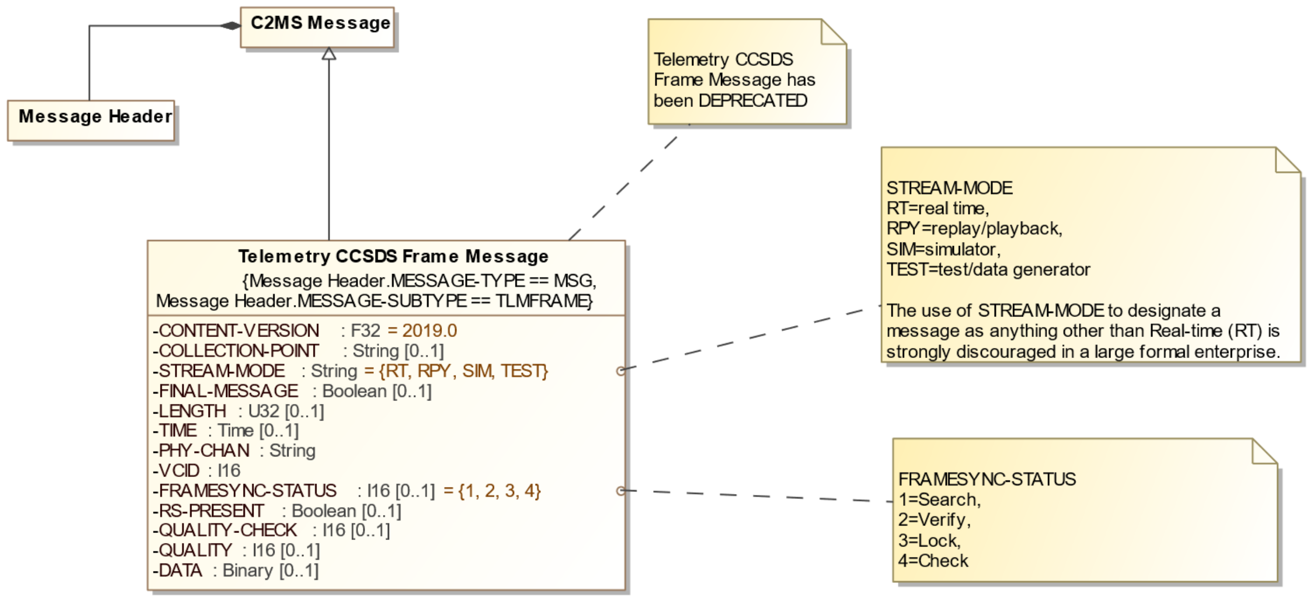

C2MS Message
Message Header
Telemetry CCSDS Frame Message has been DEPRECATED
Telemetry CCSDS Frame Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == TLMFRAME}
-CONTENT-VERSION : F32 = 2019.0
-COLLECTION-POINT : String [0..1]
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-FINAL-MESSAGE : Boolean [0..1]
-LENGTH : U32 [0..1]
-TIME : Time [0..1]
-PHY-CHAN : String
-VCID : I16
-FRAMESYNC-STATUS : I16 [0..1] = {1, 2, 3, 4}
-RS-PRESENT : Boolean [0..1]
-QUALITY-CHECK : I16 [0..1]
-QUALITY : I16 [0..1]
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.
FRAMESYNC-STATUS
1=Search,
2=Verify,
3=Lock,
4=Check

Figure 8-14. Telemetry CCSDS Frame Message Diagram

The following table describes additional field names, values, and notes for the Telemetry CCSDS Frame Message.

Table 8-73. Telemetry CCSDS Frame Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2019		Version number for this message content description
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
STREAM-MODE	String	R	**Value**	**Description**	Identifies the kind or source of the stream of telemetry as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
LENGTH	U32	O	Bytes		Length of frame
TIME	Time	O			Time of frame, usually ground receipt time
PHY-CHAN	String	R			Physical channel on which data is received

Field Name	Type	Presence	Value		Description
VCID	I16	R			CCSDS Virtual Channel Identifier (VCID)
FRAMESYNC-STATUS	I16	O	**Value**	**Description**	State of frame synchronization from equipment when frame is ingested
			1	Search	
			2	Verify	
			3	Lock	
			4	Check	
RS-PRESENT	Boolean	O			Indicates if the Reed-Solomon codes are present in the data.
QUALITY-CHECK	I16	O	**Value**	**Description**	Indicates quality checking performed, if applicable. If the bit is set the particular quality check was performed.
			Bit 0	CRC Quality Check	
			Bit 1	Reed-Solomon Quality Check	
			Bit 2	Turbo Code Quality Check	
QUALITY	I16	O	**Value**	**Description**	Indicates quality state if checking is performed. If the bit is set the particular quality check failed.
			Bit 0	CRC Quality State	
			Bit 1	Reed-Solomon Quality State	
			Bit 2	Turbo Code Quality State	
DATA	Binary	O			Raw telemetry data

8.7.3 Telemetry CCSDS CADU Frame Message

The Telemetry CCSDS CADU Frame Message is used to transport CCSDS Channel Access Data Units (CADU).

The CCSDS CADU Message consists of varying information depending on how the data source generated the CADU. In some cases the CADU will consist of only the Attached Sync Marker (ASM) and CCSDS Transfer Frame. In other cases the CADU will consist of the ASM, CCSDS Transfer Frame, and Forward Error Correction (FEC) parity data known as codeblocks or codewords, e.g. Reed-Solomon codeblocks.

The intent of this CADU message is not only to transport the data, but also to carry critical metadata describing both the contents of the CADU as well as the results of various CADU processing functions. Content metadata includes timestamp, physical channel from which the CADU originated, ASM presence and length, Inverted Data, and FEC Parity Length. Processing metadata includes Frame Synchronization Status, Derandomization, ASM Bit Errors detected, and FEC Decode results.

In a satellite ground system, the CADU Frame is typically produced by a Frame Synchronizer and FEC Decoder, which is then transferred to a CCSDS Data Link Processor that translates and produces CCSDS Transfer Frames from the CADUs.

Table 8-74. Telemetry CCSDS CADU Frame Message Summary

Sender	A C2MS compliant application such as a ground station, simulator, archive component, or front-end processor
Senders Intended Usage	Publish
Receiver	Telemetry Decommutation System, Archive System, Trending System, Expert System
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data to be decommutated and/ or archived
When	As needed but usually dependent on data rate and/or replay rate

Example

- Spacecraft health and safety data sent from ground station to ground system

8.7.3.1 Telemetry CCSDS CADU Frame Message Subjects

Table 8-75. Telemetry CCSDS CADU Frame Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*		
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	TLMCCSDS-CADUFRAME	[COMPONENT]	Stream-mode	Collection Point
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMCCSDS-CADUFRAME	SATSIM	SIM	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMCCSDS-CADUFRAME	TFEP	RT	
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	TLMCCSDS-CADUFRAME	*	RT	

Table 8-76. Properties of the *Miscellaneous Elements* for the Telemetry CCSDS CADU Frame Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field in Msg, if applicable**
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Optional	Point on ground system where data was captured	COLLECTION-POINT

Example for Publisher / Sender of Telemetry Messages

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMCCSDS-CADUFRAME.SATSIM.SIM
```

Example for Subscriber / Receiver of Telemetry Messages

```
C2MS.*.*.MSSN.*.*.MSG.TLMCCSDS-CADUFRAME.TFEP.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMCCSDS-CADUFRAME.TFEP.RT
```

8.7.3.2 Telemetry CCSDS CADU Frame Message Header

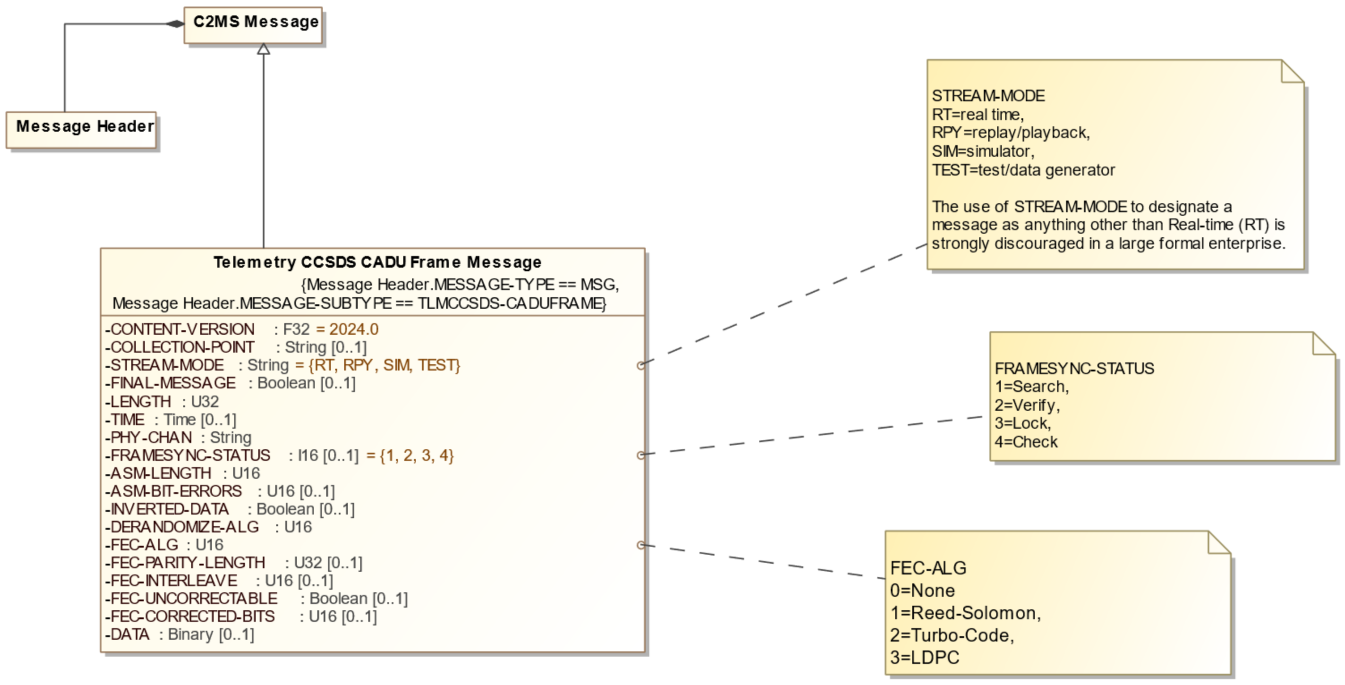
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for Telemetry CCSDS CADU Frame Message header.

Table 8-77. Telemetry CCSDS CADU Frame Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	TLMCCSDS-CADUFRAME

8.7.3.3 Telemetry CCSDS CADU Frame Message Contents

The following figure shows a UML object diagram of the Telemetry CCSDS CADU Frame Message with its required and optional fields.



C2MS Message
Message Header
Telemetry CCSDS CADU Frame Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == TLMCCSDS-CADUFRAME}
-CONTENT-VERSION : F32 = 2024.0
-COLLECTION-POINT : String [0..1]
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-FINAL-MESSAGE : Boolean [0..1]
-LENGTH : U32
-TIME : Time [0..1]
-PHY-CHAN : String
-FRAMESYNC-STATUS : I16 [0..1] = {1, 2, 3, 4}
-ASM-LENGTH : U16
-ASM-BIT-ERRORS : U16 [0..1]
-INVERTED-DATA : Boolean [0..1]
-DERANDOMIZE-ALG : U16
-FEC-ALG : U16
-FEC-PARITY-LENGTH : U32 [0..1]
-FEC-INTERLEAVE : U16 [0..1]
-FEC-UNCORRECTABLE : Boolean [0..1]
-FEC-CORRECTED-BITS : U16 [0..1]
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.
FRAMESYNC-STATUS
1=Search,
2=Verify,
3=Lock,
4=Check
FEC-ALG
0=None
1=Reed-Solomon,
2=Turbo-Code,
3=LDPC

Figure 8-15. Telemetry CCSDS CADU Frame Message Diagram

The following table describes additional field names, values, and notes for the Telemetry CCSDS CADU Frame Message.

Table 8-78. Telemetry CCSDS CADU Frame Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
STREAM-MODE	String	R	**Value**	**Description**	Identifies the kind or source of the stream of telemetry as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.

Field Name	Type	Presence	Value		Description
LENGTH	U32	R	Bits		Length of frame in bits, including the Attached Sync Marker (ASM) and any FEC Parity/Check-Symbols
TIME	Time	O			Time of frame, usually ground receipt time
PHY-CHAN	String	R			Physical channel on which data is received
FRAMESYNC-STATUS	I16	O	**Value**	**Description**	State of frame synchronization from equipment when frame is ingested
			1	Search	
			2	Verify	
			3	Lock	
			4	Check	
ASM-LENGTH	U16	R	Bits		Length of the Attached Sync Marker (ASM) in bits. A length of zero indicates the ASM was removed, e.g. by the FrameSync
ASM-BIT-ERRORS	U16	O			Number of bit errors detected in the ASM
INVERTED-DATA	Boolean	O			True if data inversion was required to achieve lock on the frame
DERANDOMIZE-ALG	U16	R	**Value**	**Description**	The algorithm used to derandomize/de-scramble the frame data
			0	None	
			1	CCSDS-131.0-B	
FEC-ALG	U16	R	**Value**	**Description**	The Forward Error Correction (FEC) algorithm used to decode/error-correct the frame data
			0	None	
			1	Reed-Solomon	
			2	Turbo-Code	
			3	LDPC	
FEC-PARITY-LENGTH	U32	O			Length of the FEC Parity/Check-Symbols in bits. Zero indicates the data was never FEC-encoded (when FEC-ALG=None), or the FEC-ALG decoded the data and removed the check-symbols
FEC-INTERLEAVE	U16	O			The block decoder's Interleave Depth used by FEC-ALG
FEC-UNCORRECTABLE	Boolean	O	**Value**	**Description**	Indicates the result of error correction processing for this CADU (if no errors were identified, this is set to FALSE)
			False	Any identified errors were corrected	
			True	Errors found that could not be corrected	
FEC-CORRECTED-BITS	U16	O			The number of error bits corrected by FEC-ALG
DATA	Binary	O			CADU telemetry containing the user data followed by the FEC Parity/Check-Symbols when FEC-PARITY-LENGTH is non-zero

8.7.4 Telemetry CCSDS Transfer Frame Message

The Telemetry CCSDS Transfer Frame Message is used to transport CCSDS Transfer Frames (TF). A TF is typically extracted from a CCSDS CADU Frame, then parsed and processed per one or more “Services” as described in the various CCSDS Data Link Protocol specifications.

The CCSDS TF Message consists of varying information depending on how the data source generated the TF. In some cases the TF will consist of only the CCSDS TF Primary Header and User Data. In other cases the TF will consist of the CCSDS TF Primary Header, Header Error Control Field (HECF), Insert Zone, User Data, Operational Control Field (OCF), and CRC known as Frame Error Control Field (FECF).

The intent of this TF message is not only to transfer the user data, but also to carry critical metadata describing both the contents of the TF as well as the results of various TF processing functions. Content metadata includes timestamp, physical channel from which the TF originated, TF Version, Spacecraft ID, Virtual Channel ID, Insert Zone Presence and Length, and FECF/HECF/OCF Presence and usage. Processing metadata includes Sequence Errors, Parse Errors, and HECF Decode results.

Table 8-79. Telemetry CCSDS Transfer Frame Message Summary

Sender	A C2MS compliant application such as a ground station, simulator, archive component, or front-end processor
Senders Intended Usage	Publish
Receiver	Telemetry Decommutation System, Archive System, Trending System, Expert System
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data to be decommutated and/ or archived
When	As needed but usually dependent on data rate and/or replay rate

Example

- Spacecraft health and safety data sent from ground station to ground system

8.7.4.1 Telemetry CCSDS Transfer Frame Message Subjects

Table 8-80. Telemetry CCSDS Transfer Frame Message Subject Naming

	Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	TLMCCSDS-TRANSFERFRAME	[COMPON ENT]	Stream-mode	Spacecraft ID (SCID)	Virtual Channel ID (VCID)	Collection Point
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMCCSDS-TRANSFERFRAME	SATSIM	SIM	4	2	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMCCSDS-TRANSFERFRAME	TFEP	RT	5	1	
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	TLMCCSDS-TRANSFERFRAME	*	RT	6	*	

Table 8-81. Properties of the *Miscellaneous Elements* for the Telemetry CCSDS Transfer Frame Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Required	SCID (Spacecraft ID)	SCID from msg content or from header portion of CCSDS transfer frame
ME4	Required	VCID (Virtual Channel ID)	VCID from msg content or from header portion of CCSDS transfer frame
ME5	Optional	Point on ground system where data was captured	COLLECTION-POINT

Example for Publisher / Sender of Telemetry Messages

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMCCSDS-TRANSFERFRAME.SATSIM.SIM.2.1
```

Example for Subscriber / Receiver of Telemetry Messages

```
C2MS.*.*.MSSN.*.*.MSG.TLMCCSDS-TRANSFERFRAME.TFEP.RT.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMCCSDS-TRANSFERFRAME.TFEP.RT.2.1
```

8.7.4.2 Telemetry CCSDS Transfer Frame Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for Telemetry CCSDS Transfer Frame Message header.

Table 8-82. Telemetry CCSDS Transfer Frame Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	TLMCCSDS-TRANSFERFRAME

8.7.4.3 Telemetry CCSDS Transfer Frame Message Contents

The following figure shows a UML object diagram of the Telemetry CCSDS Transfer Frame Message with its required and optional fields.

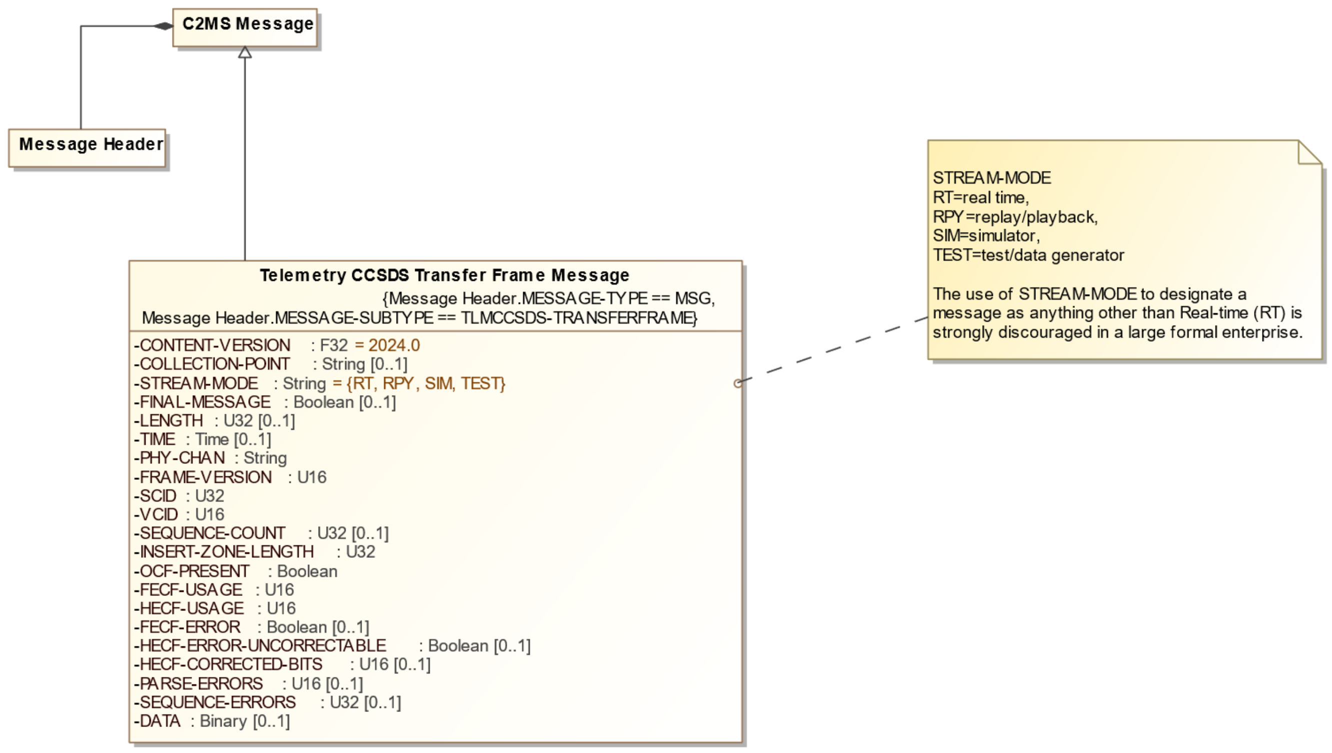

C2MS Message
Message Header
Telemetry CCSDS Transfer Frame Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == TLMCCSDS-TRANSFERFRAME}
-CONTENT-VERSION : F32 = 2024.0
-COLLECTION-POINT : String [0..1]
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-FINAL-MESSAGE : Boolean [0..1]
-LENGTH : U32 [0..1]
-TIME : Time [0..1]
-PHY-CHAN : String
-FRAME-VERSION : U16
-SCID : U32
-VCID : U16
-SEQUENCE-COUNT : U32 [0..1]
-INSERT-ZONE-LENGTH : U32
-OCF-PRESENT : Boolean
-FECF-USAGE : U16
-HECF-USAGE : U16
-FECF-ERROR : Boolean [0..1]
-HECF-ERROR-UNCORRECTABLE : Boolean [0..1]
-HECF-CORRECTED-BITS : U16 [0..1]
-PARSE-ERRORS : U16 [0..1]
-SEQUENCE-ERRORS : U32 [0..1]
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-16. Telemetry CCSDS Transfer Frame Message Diagram

The following table describes additional field names, values, and notes for the Telemetry CCSDS Transfer Frame Message.

Table 8-83. Telemetry CCSDS Transfer Frame Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
STREAM-MODE	String	R	**Value**	**Description**	Identifies the kind or source of the stream of telemetry as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.

Field Name	Type	Presence	Value	Description
LENGTH	U32	R	Bytes	Length of the transfer frame in bytes
TIME	Time	O		Time of frame, usually ground receipt time
PHY-CHAN	String	R		Physical channel on which data is received
FRAME-VERSION	U16	R		The CCSDS Transfer Frame Version of this frame
SCID	U32	R		CCSDS Spacecraft Identifier (SCID)
VCID	U16	R		CCSDS Virtual Channel Identifier (VCID)
SEQUENCE-COUNT	U32	O		Current value of the CCSDS Virtual Channel Frame Count
INSERT-ZONE-LENGTH	U32	R		The length in Bytes of the CCSDS AOS Insert Zone of this frame
OCF-PRESENT	Boolean	R		Indicates whether the CCSDS Operational Control Field (OCF) is present in this frame
FECF-USAGE	U16	R	**Value** / **Description**: 0 = Not Present; 1 = Is Present, Not Used; 2 = Is Present, Is Used	Indicates presence and usage of the CCSDS Frame Error Control Field (i.e. CRC). “Is Present Not Used” means the FECF field exists in the DATA, but the CRC was not performed
HECF-USAGE	U16	R	**Value** / **Description**: 0 = Not Present; 1 = Is Present, Not Used; 2 = Is Present, Is Used	Indicates presence and usage of the CCSDS Header Error Control Field (i.e. Reed-Solomon 10,6). “Is Present Not Used” means the HECF field exists in the DATA, but the Reed-Solomon (10,6) Decoding was not performed
FECF-ERROR	Boolean	O	**Value** / **Description**: False = FECF/CRC detected no error(s) in frame; True = FECF/CRC detected error(s) in frame	Indicates whether FECF/CRC detected any errors in this frame; these are uncorrectable
HECF-ERROR-UNCORRECTABLE	Boolean	O		True indicates that the HECF/Reed-Solomon(10,6) detected an uncorrectable error in this frame
HECF-CORRECTED-BITS	U16	O		The number of header error bits corrected by the HECF/Reed-Solomon(10,6) algorithm
PARSE-ERRORS	U16	O		The combined number of parse errors relative to this frame, includes: Uncorrectable FEC Error received from the FrameSync, FECF/CRC Error, Uncorrectable HECF/Reed-Solomon(10,6) error, Invalid Transfer Frame Version received, Invalid Length.
SEQUENCE-ERRORS	U32	O		Indicates the number of Virtual Channel Frame Count sequence errors relative to this frame, e.g. frame gaps/missed frames
DATA	Binary	O		CCSDS Transfer Frame telemetry containing the Transfer Frame Primary Header, the optional InsertZone (for AOS), the Transfer Frame Data Field, and the

Field Name	Type	Presence	Value	Description
				optional Transfer Frame Trailer (with optional OCF and FECF fields)

8.7.5 Telemetry TDM Frame Message

The Telemetry Time-Division Multiplexing (TDM) Frame Message is used to transfer TDM frames. The Telemetry Message Contents simply consists of the raw TDM frame, length of the frame, and the time of the frame.

Table 8-84. Telemetry TDM Frame Message Summary

Sender	A C2MS compliant application such as a ground station, simulator, archive component, or front-end processor
Senders Intended Usage	Publish
Receiver	Telemetry Decommutation System, Archive System, Trending System, Expert System
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data to be decommutated and/ or archived
When	As needed but usually dependent on data rate and/or replay rate

Example

- Spacecraft health and safety data sent from ground station to ground system

8.7.5.1 Telemetry TDM Frame Message Subjects

Table 8-85. Telemetry TDM Frame Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	TLMTDM	[COMPONENT]	Stream-mode	Not Used	Not Used	Collection Point
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMTDM	SATSIM	SIM	FILL	FILL	GEP1
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMTDM	TFEP	RT	FILL	FILL	GEP2
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	TLMTDM	*	RT			

Table 8-86. Properties of the *Miscellaneous Elements* for the Telemetry TDM Frame Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Not used		
ME4	Not used		
ME5	Optional	Point on ground system where data was captured	COLLECTION-POINT

Example for Publisher / Sender of Telemetry Messages

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMTDM.SATSIM.SIM.FILL.FILL.GEP1
```

Example for Subscriber / Receiver of Telemetry Messages

```
C2MS.*.*.MSSN.*.*.MSG.TLMTDM.TFEP.RT.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMTDM.TFEP.RT.FILL.FILL.GEP2
```

8.7.5.2 Telemetry TDM Frame Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for Telemetry TDM Frame Message header.

Table 8-87. Telemetry TDM Frame Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	TLMTDM

8.7.5.3 Telemetry TDM Frame Message Contents

The following figure shows a UML object diagram of the Telemetry TDM Frame Message with its required and optional fields.

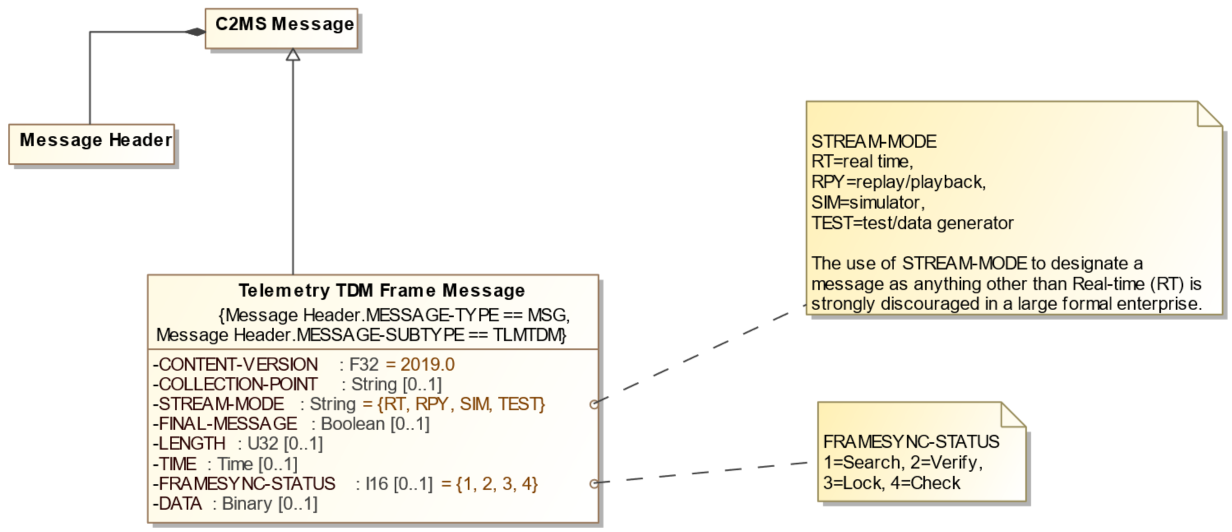

C2MS Message
Message Header
Telemetry TDM Frame Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == TLMTDM}
-CONTENT-VERSION : F32 = 2019.0
-COLLECTION-POINT : String [0..1]
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-FINAL-MESSAGE : Boolean [0..1]
-LENGTH : U32 [0..1]
-TIME : Time [0..1]
-FRAMESYNC-STATUS : I16 [0..1] = {1, 2, 3, 4}
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.
FRAMESYNC-STATUS
1=Search, 2=Verify,
3=Lock, 4=Check

Figure 8-17. Telemetry TDM Frame Message Diagram

The following table describes additional field names, values, and notes for the Telemetry TDM Frame Message.

Table 8-88. Telemetry TDM Frame Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2019		Version number for this message content description
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
STREAM-MODE	String	R	**Value**	**Description**	Identifies the kind or source of the stream of telemetry as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
LENGTH	U32	O	Bytes		Length of frame in bytes
TIME	Time	O			Time of frame, usually ground receipt time
FRAMESYNC-STATUS	I16	O	**Value**	**Description**	State of frame synchronization from equipment when frame is ingested
			1	Search	
			2	Verify	
			3	Lock	

Field Name	Type	Presence	Value		Description
			4	Check	
DATA	Binary	O			Raw telemetry data

8.7.6 Telemetry Processed Frame Message

The Telemetry Processed Frame Message is a hybrid between the unprocessed (raw) Telemetry Message and the Mnemonic Value Data Message. It contains both raw and converted data for a frame of telemetry data that is organized in the message by mnemonic. Thus, it is frame-based as are the telemetry messages, but mnemonic-organized as are the Mnemonic Value Data Messages. It serves to provide all the telemetry data in a raw and processed format. Therefore, many consumers could be provided with a substantial amount of data without needing to specifically request a custom selected mnemonic data set.

When this message is published is to be determined by the mission or data provider. It could be published “alongside” or in accordance with the raw telemetry data messages or by itself. Also, it could be published automatically or only by request, as a replay.

Table 8-89. Telemetry Processed Frame Message Summary

Sender	A C2MS compliant application such as a ground station, simulator, archive component, or front-end processor
Senders Intended Usage	Publish
Receiver	Telemetry Decommutation System, Archive System, Trending System, Expert System
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data to be decommutated and/ or archived
When	As needed but usually dependent on data rate and/or replay rate

Example

- Spacecraft health and safety data sent from ground station to ground system

8.7.6.1 Telemetry Processed Frame Message Subjects

Table 8-90. Telemetry Processed Frame Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*				
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	TLMPROC	[COMPONENT]	Stream-mode	Not Used	Not Used	Collection Point
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMPROC	SATSIM	SIM	FILL	FILL	GEP1
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	TLMPROC	TFEP	RT	FILL	FILL	GEP2
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	TLMPROC	*	RT			

Table 8-91. Properties of the *Miscellaneous Elements* for the Telemetry Processed Frame Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field in Msg, if applicable**
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Not Used		
ME4	Not Used		
ME5	Optional	Point on ground system where data was captured	COLLECTION-POINT

Example for Publisher / Sender of Telemetry Messages

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMPROC.SATSIM.SIM.FILL.FILL.GEP1
```

Example for Subscriber / Receiver of Telemetry Messages

```
C2MS.*.*.MSSN.*.*.MSG.TLMPROC.TFEP.RT.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.TLMPROC.TFEP.RT.FILL.FILL.GEP2
```

8.7.6.2 Telemetry Processed Frame Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for Telemetry Processed Frame Message header.

Table 8-92. Telemetry Processed Frame Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	TLMPROC

8.7.6.3 Telemetry Processed Frame Message Contents

The following figure shows a UML object diagram of the Telemetry Processed Frame Message with its required, optional, and dependent fields.

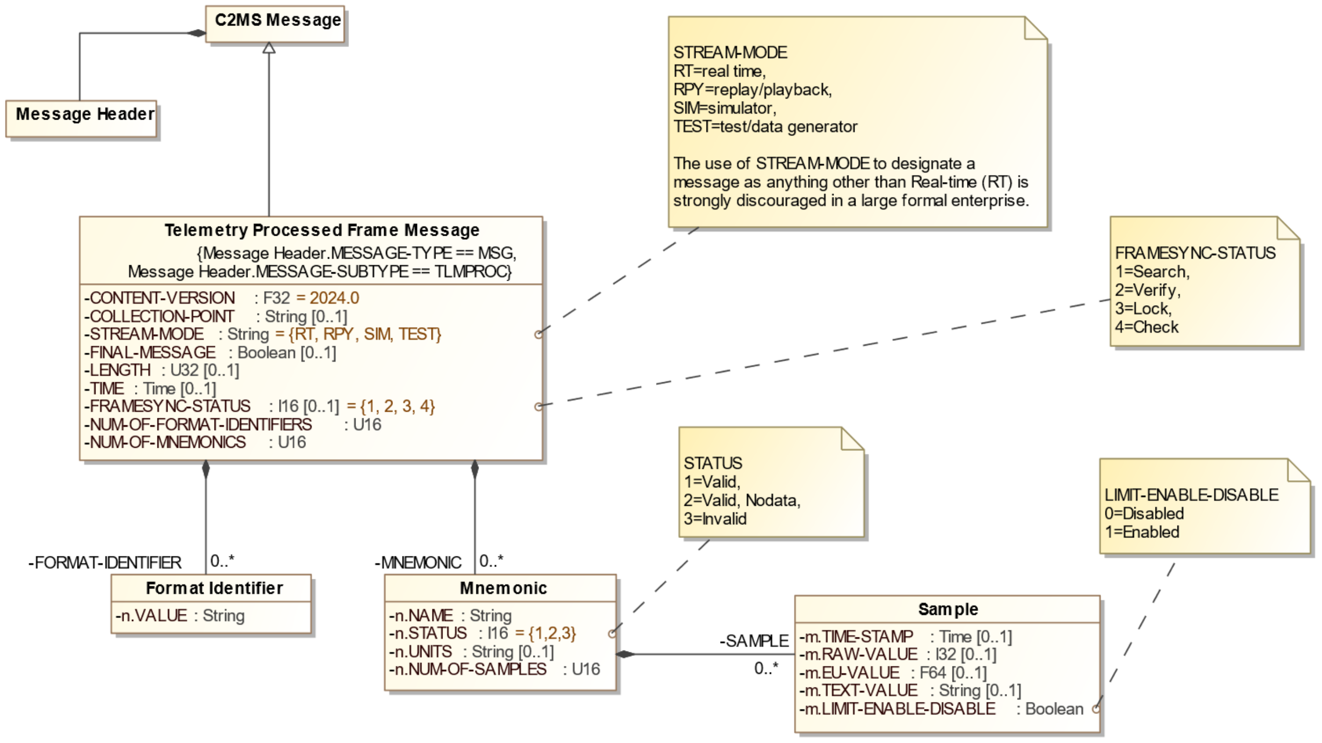

C2MS Message
Message Header
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.
FRAMESYNC-STATUS
1=Search,
2=Verify,
3=Lock,
4=Check
Telemetry Processed Frame Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == TLMPROC}
-CONTENT-VERSION : F32 = 2024.0
-COLLECTION-POINT : String [0..1]
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-FINAL-MESSAGE : Boolean [0..1]
-LENGTH : U32 [0..1]
-TIME : Time [0..1]
-FRAMESYNC-STATUS : I16 [0..1] = {1, 2, 3, 4}
-NUM-OF-FORMAT-IDENTIFIERS : U16
-NUM-OF-MNEMONICS : U16
STATUS
1=Valid,
2=Valid, Nodata,
3=Invalid
LIMIT-ENABLE-DISABLE
0=Disabled
1=Enabled
-FORMAT-IDENTIFIER
0..*
-MNEMONIC
0..*
Format Identifier
-n.VALUE : String
Mnemonic
-n.NAME : String
-n.STATUS : I16 = {1,2,3}
-n.UNITS : String [0..1]
-n.NUM-OF-SAMPLES : U16
-SAMPLE
0..*
Sample
-m.TIME-STAMP : Time [0..1]
-m.RAW-VALUE : I32 [0..1]
-m.EU-VALUE : F64 [0..1]
-m.TEXT-VALUE : String [0..1]
-m.LIMIT-ENABLE-DISABLE : Boolean

Figure 8-18. Telemetry Processed Frame Message Diagram

The following table describes additional field names, values, and notes for the Telemetry Processed Frame Message Additional Information

Table 8-93. Telemetry Processed Frame Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
STREAM-MODE	String	R	**Value**	**Description**	Identifies the kind or source of the stream of telemetry as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
LENGTH	U32	O			Length of frame in bytes

Field Name	Type	Presence	Value		Description
TIME	Time	O			Time of frame, usually ground receipt time
FRAMESYNC-STATUS	I16	O	**Value**	**Description**	State of frame synchronization from equipment when frame is ingested
			1	Search	
			2	Verify	
			3	Lock	
			4	Check	
NUM-OF-FORMAT-IDENTIFERS	U16	R	0+		Number of fields used to identify the frames (e.g. TDM major/minor frames would have a value of 2). Zero is only permissible for vehicles with one telemetry format with a single type of frame.
FORMAT-IDENTIFIER.n.VALUE	String	D			Value of the nth field used to identify the telemetry - "n" starts at "1". If the message is used with XTCE, this is the nth comparison in a comparison list in the restriction criteria in an XTCE container. This field is required for each format identifier when NUM-OF-FORMAT-IDENTIFIERS > 0 .
NUM-OF-MNEMONICS	U16	R	1+		Total number of mnemonics in this message
MNEMONIC.n.NAME	String	D			Name of the 'nth' mnemonic - "n" starts at "1". This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.STATUS	I16	D	**Value**	**Description**	Status of the 'nth' mnemonic: valid mnemonic, or valid mnemonic with no data, or invalid mnemonic. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
			1	Valid	
			2	Valid, Nodata	
			3	Invalid	
MNEMONIC.n.UNITS	String	O			Units associated with the raw value converted to engineering units for the 'nth' mnemonic
MNEMONIC.n.NUM-OF-SAMPLES	U16	D			Number of data samples for the 'nth' mnemonic. This value should equal the number of times the mnemonic appears in the telemetry frame (e.g. will be greater than 1 for super-commutated telemetry points. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.SAMPLE.m.TIME-STAMP	Time	O			Time stamp for the 'nth' data sample of the 'nth' mnemonic - both "n" and "m" start at "1".
MNEMONIC.n.SAMPLE.m.RAW-VALUE	I32	O			Raw value for the 'nth' data sample of the 'nth' mnemonic
MNEMONIC.n.SAMPLE.m.EU-VALUE	F64	O			Raw value converted to Engineering Units if engineering units conversion is present for the 'nth' data sample of the 'nth' mnemonic
MNEMONIC.n.SAMPLE.m.TEXT-VALUE	String	O			Raw value converted to a text string if text conversion is present for the 'nth' data sample of the 'nth' mnemonic
MNEMONIC.n.SAMPLE.m.LIMIT-ENABLE-DISABLE	Boolean	D	**Value**	**Description**	Indicates the limit checking state for the 'nth' data sample of the 'nth' mnemonic. This field is required for each mnemonic with samples, in other words, when NUM-OF-MNEMONICS > 0 and NUM-OF-SAMPLES > 0 for a given mnemonic.
			0	Disabled	
			1	Enabled	

8.8 Replay Telemetry Data Messages

The Replay (Request and Response) Telemetry Data Messages are for historical, archived data. However, these messages can also be used to request real-time data or even future data. Because not all downlinked data is automatically forwarded (published) in real-time, these messages also provide the means to request the publication of a current or future data stream in real-time.

The Replay Telemetry Data Messages provide access to streams of raw telemetry data (packets, frames, etc.) and also processed (converted) data. This data can be real-time, replayed from a data archive, from a simulator or data generator, or for a future flow of data. A request for future data occurs before the satellite has downlinked the data, and ensures that once the data is collected on the ground it will be forwarded in real-time.

The Replay Telemetry Data Messages are an example of the Request-Response-Message Triad. That is, the Replay Telemetry Data Request Message is followed by the Replay Telemetry Data Response Message followed by a series or stream of Telemetry Data messages.

A common use of the Replay Telemetry Data Messages is when a decommutation component needs to process raw archived telemetry data. The decommutation component builds a Replay Telemetry Data Request, specifying the source of telemetry data, range of data, and the playback speed.

The component sends the request to a Telemetry Archive component. The Telemetry Archive component processes the request by locating the requested telemetry data from the archive and returning the status of the request in a Replay Telemetry Data Response Message. The Telemetry Archive component will then retrieve the telemetry data from the archive and publish it in Replay Telemetry Data Messages at the requested speed. The requesting component that has subscribed to the Telemetry Data Messages receives and processes the requested telemetry data. The decommutation component may in turn provide processed telemetry messages or archived mnemonic data value messages.

As stated above, requests for real-time data can be for current streaming data or for a future data stream, one not yet received at a ground station. Requests for real-time data present some new ways of thinking in light of present-day technology.

Though not applicable in the past, some present-day recording devices can pause and resume real-time data streams, and even step through them. Traditionally, these features were only available or associated with playback data streams, but more sophisticated recording devices have merged these playback capabilities with real-time data streams. This has led to some new terms and concepts such as a “paused real-time data stream”, a “resumed real-time data stream”, and a “real-time data stream playing at half speed”.

Therefore, depending on the sophistication of the data provider and recording mechanism, some features traditionally associated with a playback data stream may be available with a real-time data stream. However, if the data provider does not provide such features, it must return an appropriate status to the requestor in the Response message.

8.8.1 Replay Telemetry Data Request Message

A Replay Telemetry Data Request Message is a service request that is issued to a telemetry data provider by an application to receive telemetry data. The request could be for archived or real-time data.

For an archived data request, every field in the Replay Telemetry Data Request Message will be valid. That is, any field can be used. However, since some fields are mutually exclusive, not all fields will be used. The Replay Telemetry Response Message returns the status of the request. Please see Section 6.3.7 C2MS Messages: Their Characteristics and Interactions for a general discussion on these types of messages.

For all requests, the mission and satellite identification is found in the message header and associated subject name. Other data selection parameters include:

- Data stream characteristics (flow control, speed, data type)
- Broad data selection parameters (time window, orbit no., or data file name)
- Refined data selection parameters (data format and data specific)

For a real-time data stream request, the following fields of the Request message are not valid:

- PLAYBACK-RATIO*
- DATA-RATE*
- File name fields

* However, as mentioned in the previous section, a more sophisticated data provider may be able to stream real-time data at rates slower than real-time.

Table 8-94. Replay Telemetry Data Request Message Summary

Sender	Any C2MS compliant application requesting archived telemetry data
Senders Intended Usage	Request
Receiver	Any C2MS compliant application with access to a telemetry archive
Receivers Intended Usage	Subscribe
What	Telemetry data as Telemetry Messages
When	As needed

Examples

1. Archived telemetry data replayed to a Telemetry and Command (T&C) component.

2. “Register” to receive a future real-time telemetry data stream.

8.8.1.1 Replay Telemetry Data Request Message Subjects

Table 8-95. Replay Telemetry Data Request Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	REQ	RTLM	[DESTINATION-COMPONENT]					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	RTLM	TLM2					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	RTLM	TLM3					
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	REQ	RTLM	TLM3					

Table 8-96. Properties of the *Miscellaneous Elements* for the Replay Telemetry Data Request Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field in Msg, if applicable**
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2	Not used		
ME3	Not used		

Examples

Two components, ARCHIVER and TLM3, interact with the Replay Telemetry Request Message.

TLM3 sends a message with the following subject to request a Replay of Telemetry.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.RTLM.ARCHIVER
```

ARCHIVER subscribes to receive the Replay Telemetry Request Message.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.RTLM.ARCHIVER or

C2MS.*.*.*.*.*.REQ.RTLM.ARCHIVER
```

8.8.1.2 Replay Telemetry Data Request Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Replay Telemetry Data Request Message header.

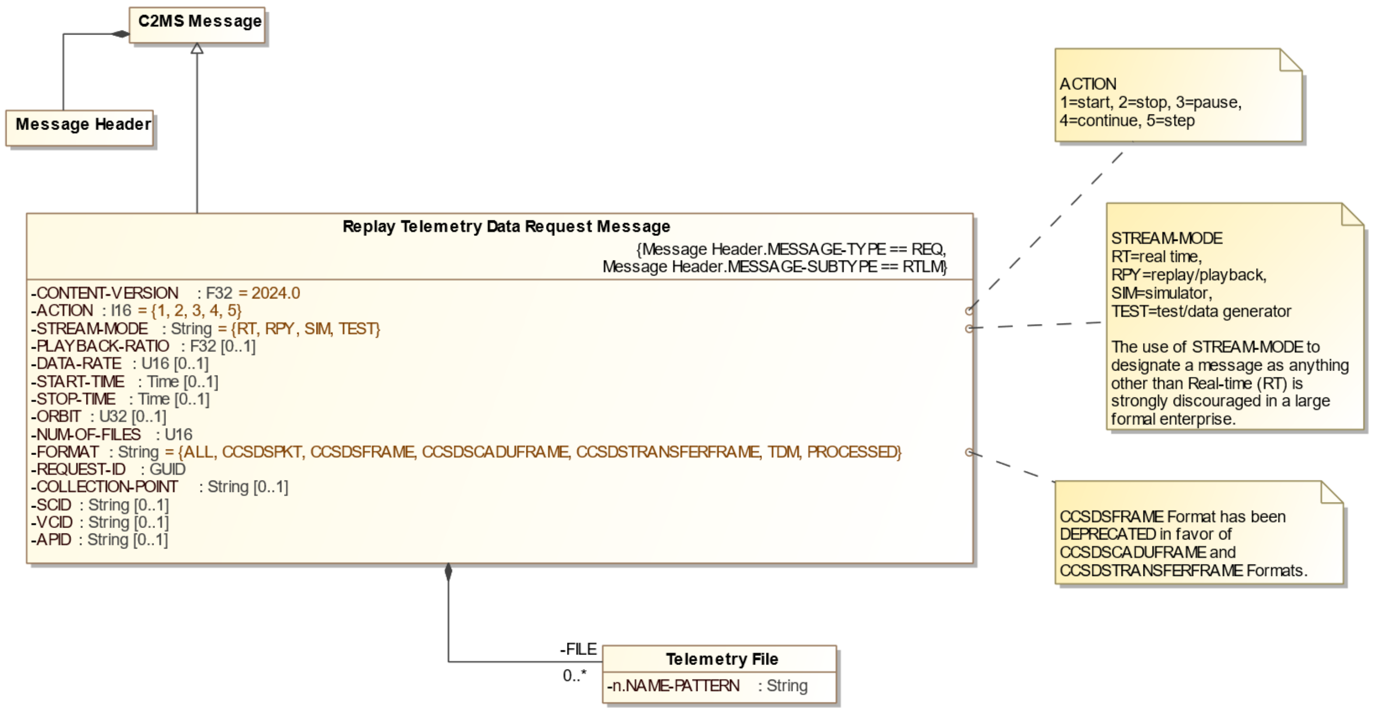
Table 8-97. Replay Telemetry Data Request Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	RTLM

8.8.1.3 Replay Telemetry Data Request Message Contents

The Replay Telemetry Data Request is used for real-time and archived data.

The following figure shows a UML object diagram of the Replay Telemetry Data Request Message with its required and optional fields.



C2MS Message
Message Header
Replay Telemetry Data Request Message
{Message Header.MESSAGE-TYPE == REQ, Message Header.MESSAGE-SUBTYPE == RTLM}
-CONTENT-VERSION : F32 = 2024.0
-ACTION : I16 = {1, 2, 3, 4, 5}
-STREAM-MODE : String = {RT, RPY, SIM, TEST}
-PLAYBACK-RATIO : F32 [0..1]
-DATA-RATE : U16 [0..1]
-START-TIME : Time [0..1]
-STOP-TIME : Time [0..1]
-ORBIT : U32 [0..1]
-NUM-OF-FILES : U16
-FORMAT : String = {ALL, CCSDSPKT, CCSDSFRAME, CCSDSCADUFRAME, CCSDSTRANSFERFRAME, TDM, PROCESSED}
-REQUEST-ID : GUID
-COLLECTION-POINT : String [0..1]
-SCID : String [0..1]
-VCID : String [0..1]
-APID : String [0..1]
ACTION 1=start, 2=stop, 3=pause, 4=continue, 5=step
STREAM-MODE RT=real time, RPY=replay/playback, SIM=simulator, TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.
CCSDSFRAME Format has been DEPRECATED in favor of CCSDSCADUFRAME and CCSDSTRANSFERFRAME Formats.
-FILE
0..*
Telemetry File
-n.NAME-PATTERN : String

Figure 8-19. Replay Telemetry Data Request Message Diagram

The following table describes additional field names, values, and notes for the Replay Telemetry Data Request.

Table 8-98. Replay Telemetry Data Request Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
ACTION	I16	R	**Value**	**Description**	Identifies the desired action to perform on the telemetry data stream.
			1	Start	
			2	Stop	
			3	Pause	
			4	Continue	
			5	Step	
STREAM-MODE	String	R	**Value**	**Description**	Identifies the type of data to be published. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real Time	
			RPY	Replay / Playback	
			SIM	Simulator	
			TEST	Test / Data Generator	

Field Name	Type	Presence	Value		Description
PLAYBACK-RATIO	F32	O	> 0 and < 1 is slower than real-time rate = 1 is equal to the real-time rate > 1 is faster than real-time rate		Speed of playback as a ratio of playback rate to real-time rate. This is the default method; default = 1.
DATA-RATE	U16	O	> 0		Data rate in Kilobits per second
START-TIME	Time	O			Time of first telemetry data. Defaults to the start of the archive
STOP-TIME	Time	O			Time of last telemetry data. Defaults to the end of the archive
ORBIT	U32	O	0+		Orbit or revolution number of the vehicle (past or future).
NUM-OF-FILES	U16	R	1+		Number of Telemetry files to replay
FILE.n.NAME-PATTERN	String	D			Name of the Telemetry file to replay - "n" starts at "1". This field is required for each file when NUM-OF-FILES > 0.
FORMAT	String	R	**Value**	**Description**	Telemetry Message types to playback. Note: A provider may not be capable of providing all types; e.g., only raw or only processed data. CCSDSFRAME has been deprecated. Use CCSDSCADUFRAME and CCSDSTRANSFERFRAME formats instead.
			ALL	All message types	
			CCSDSPKT	CCSDS packets	
			DEPRECATED CCSDSFRAME	CCSDS frames	
			CCSDSCADUFRAME	CCSDS CADU frames	
			CCSDSTRANSFERFRAME	CCSDS transfer frames	
			TDM	TDM frames	
			PROCESSED	Converted TLM	
REQUEST-ID	GUID	R			ID to identify the request message. If the REQUEST-ID is used here for the first time, ACTION must be "Start". However, specifying the same REQUEST-ID from a previous request is used to perform a subsequent ACTION against an established telemetry replay stream.
COLLECTION-POINT	String	O			Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
SCID	String	O			CCSDS Spacecraft Identifier (SCIDs): comma delimited SCIDs with '-' for SCID ranges. Example: 1,2,6-9,10
VCID	String	O			CCSDS Virtual Channel Identifier (VCIDs): comma delimited VCIDs with '-' for VCID ranges. Example: 1,2,6-9,10
APID	String	O			CCSDS Application Process Identifier (APIDs): comma delimited APIDs with '-' for APID ranges. Example: 1,2,6-9,10. The APID field is only applicable when requested FORMAT is CCSDSPKT or

Field Name	Type	Presence	Value	Description
				PROCESSED. APID does not apply to CADUs, Transfer Frames, or TDM.

For an explanation on how the START-TIME and STOP-TIME fields could be used, see Table 8-20 Examples of Start and Stop Times.

Field Name Usage

ACTION

This field controls the flow of the data from the provider. Once a telemetry data stream has begun, the requestor may Pause, Continue, Step though (frame-by-frame or Packet-by-packet), or Stop. If a data provider does not provide all the options of the data flow, it should respond with the appropriate status in a Replay Telemetry Data Response Message.

REQUEST-ID

This field may be used to submit an ACTION related to a previous request designated by the same REQUEST-ID. For example, a new REQUEST-ID is used to start a telemetry stream and a later request with the same REQUEST-ID can be used to pause, continue or stop the stream based on the specified ACTION.

The first time a REQUEST-ID value is used in a request, the ACTION must be "Start". It is invalid for any subsequent requests using that same REQUEST-ID to specify an ACTION value of "Start".

Data Stream Speed

The requestor has two methods of specifying the replay speed of the selected telemetry data in the fields PLAYBACK-RATIO and DATA-RATE. Either method can be used, but not both.

If both methods are specified in the request, the PLAYBACK-RATIO should default. DATA-RATE is the rate the data will be replayed in kilobits per second. PLAYBACK-RATIO is a ratio of the playback speed to the real-time speed.

If the rate of the replay is to be the same as the real-time rate, the ratio would be REPLAY: REAL-TIME or 1. If the replay speed is to be twice as fast as the real-time rate, the ratio would be 2. If the replay speed is to be one-tenth the speed of the real-time rate, the ratio would be 0.1. Thus, the PLAYBACK-RATIO must be greater than 0.

No upper bound is placed on the PLAYBACK-RATIO, however, the responder/publisher of the data may self-impose their own replay limit.

COLLECTION-POINT

Some satellites may be in contact with more than one ground station or receiver at a time, with each simultaneously collecting the downlinked data stream. If a requestor desires a data stream from a specific collection point, this field can be used to specify that site.

Broad Data Selection Parameters

The requestor of a telemetry data replay (or playback) can select the limits or range of the data to be replayed by specifying a time window or orbit number, or by naming specific files of data.

Any method can be used to limit the data, but not more than one. If Start and Stop times are present in the request message, this method will take precedence over any other provided information. Specifying specific data files will take precedence over orbit number.

Refined Data Selection Parameters (data format and data specific)

Requestors are required to specify the data format (FORMAT). More specific data selection can be accomplished with the CCSDS Virtual Channel Identifier (VCID) and CCSDS Application Process Identifier (APID) fields.

8.8.2 Replay Telemetry Data Response Message

A Replay Telemetry Data Response Message is sent by a telemetry data stream provider in response to a Replay Telemetry Data Request Message. The primary purpose of the Replay Telemetry Data Response Message is to provide acknowledgment of the Replay Telemetry Data Request Message and a status of the action completed.

Multiple Replay Telemetry Data Response Messages may be used by the requestor if the processing and completion of the request is lengthy. In this event, an interim or interactive "working" type message would be issued to let the original application know that the action is still being processed.

Table 8-99. Replay Telemetry Data Response Message Summary

Sender	Application that received the Replay Telemetry Request Message
Senders Intended Usage	Reply
Receiver	Application that issued the Replay Telemetry Request Message and an application collecting Messages for audit trail purposes
Receivers Intended Usage	Subscribe
What	Provide success/failure response to the service that was requested
When	Upon receipt of Replay Telemetry Request Message or on an interval for those services that are time-consuming

Example

- Previously recorded spacecraft health and safety data retrieved from an archive and sent to a Telemetry and Command System.

8.8.2.1 Replay Telemetry Data Response Message Subjects

Table 8-100. Replay Telemetry Data Response Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	RTLM	[DESTINATION -COMPONENT]	[Response-Status: ack, working, success, failure]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	RTLM	TLM2	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	RTLM	TLM3	3				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	RESP	RTLM	TLM3	*				

Table 8-101. Properties of the *Miscellaneous Elements* for the Replay Telemetry Data Response Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field Origination in Msg, if applicable**
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components, ARCHIVER and TLM3 interact with the Replay Telemetry Response Message.

ARCHIVER sends a Replay Telemetry Response Message back to the requestor.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.RTLM.TLM3.1
```

The original requestor subscribes to the Replay Telemetry Response Message.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.RTLM.TLM3.>
```

8.8.2.2 Replay Telemetry Data Response Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Replay Telemetry Data Response Message Header.

Table 8-102. Replay Telemetry Data Response Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	RTLM

8.8.2.3 Replay Telemetry Data Response Message Contents

The following figure shows a UML object diagram of the Replay Telemetry Data Response Message with its required and optional fields.

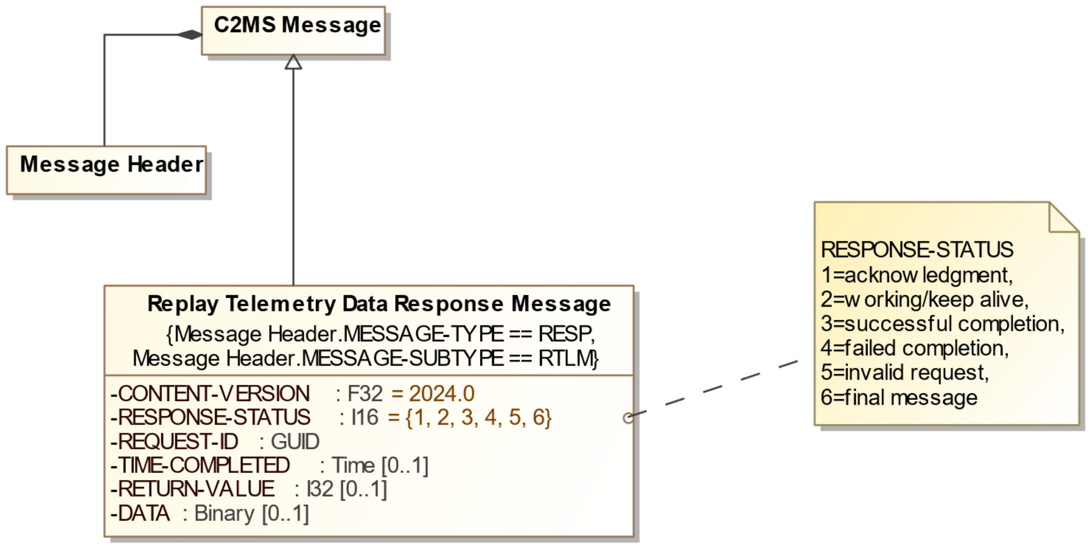

C2MS Message
Message Header
Replay Telemetry Data Response Message
{Message Header.MESSAGE-TYPE == RESP,
Message Header.MESSAGE-SUBTYPE == RTLM}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1, 2, 3, 4, 5, 6}
-REQUEST-ID : GUID
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-DATA : Binary [0..1]
RESPONSE-STATUS
1=acknowledgment,
2=working/keep alive,
3=successful completion,
4=failed completion,
5=invalid request,
6=final message

Figure 8-20. Replay Telemetry Data Response Message Diagram

The following table describes additional field names, values, and notes for the Replay Telemetry Data Response Message.

Table 8-103. Replay Telemetry Data Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the request being processed
			1	Acknowledgement	
			2	Working/Keep Alive	
			3	Successful completion	
			4	Failed completion	
			5	Invalid Request	
			6	Final Message	
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
TIME-COMPLETED	Time	O			Time application completed processing the request
RETURN-VALUE	I32	O			Return value or status based on the RESPONSE-STATUS. Useful to provide function call status or error code in the case of failed completion
DATA	Binary	O			Additional data that may be desired along with the completion status

8.9 Real-time Mnemonic Value Messages

The Mnemonic Value Messages provide the mechanism for requesting and sending mnemonic data that has been processed from a data stream.

A requesting component, such as a trending system, may request a set of mnemonics from a data stream. The request may be for a single set of values or for a continual stream of values based upon a sampling rate or upon change. The responding component, such as a Telemetry and Command system, extracts the set of requested mnemonics from the data stream and sends them to the requesting component.

The Mnemonic Value Messages remove the burden from the requesting component of having to process (decommutate) the data and therefore having to know the specifics of the telemetry database. The three specific Real-time Mnemonic Value messages are:

- Mnemonic Value Request Message
- Mnemonic Value Response Message
- Mnemonic Value Data Message

8.9.1 Mnemonic Value Request Message

The Mnemonic Value Request Message is used when an application is to receive real-time mnemonic data from another application.

A common use of the Mnemonic Value Request Message is when a Flight Dynamics application is to producing a flight dynamics product, such as ephemeris, orbit, and attitude products. The Flight Dynamics application builds a request of the mnemonic values to be collected and sends this request to the telemetry subsystem. The telemetry subsystem would package the decommutated values and flags for all the requested mnemonics into a Mnemonic Value Data Message and publish it for use.

The Mnemonic Value Request Message is used to subscribe to real-time mnemonics, while the Mnemonic Value Response and Mnemonic Value Data Message are used to publish mnemonics.

The Mnemonic Value Request Message is used to request current mnemonic values and optionally a subscription to ongoing mnemonic updates as a stream. This messages has the following specific uses:

- Oneshot - the mnemonic values are returned in the response message.
- Start - the mnemonic values are returned in the response message and a subscription is established resulting in a series of mnemonic value messages that flow until stopped or expired.
- Stop - the mnemonic values are returned in the response message and the subscription to mnemonic values established via Start is ended.

Table 8-104. Mnemonic Value Request Message Summary

Sender	A C2MS compliant application such as a GUI Subsystem, Command Verification process, Expert Subsystem, FDS Process
Senders Intended Usage	Request
Receiver	Any mnemonic processor such as a Telemetry Decommutation process or data archiver
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data and configuration data values
When	As needed

Example

1. Command verification process requests telemetry value to check if spacecraft command executed as expected.

2. Flight Dynamics process requests telemetry values used in the generation of Flight Dynamics products.

The following figure shows a UML sequence diagram for the different Mnemonic Value Messages exchanged between the requestor and data provider.

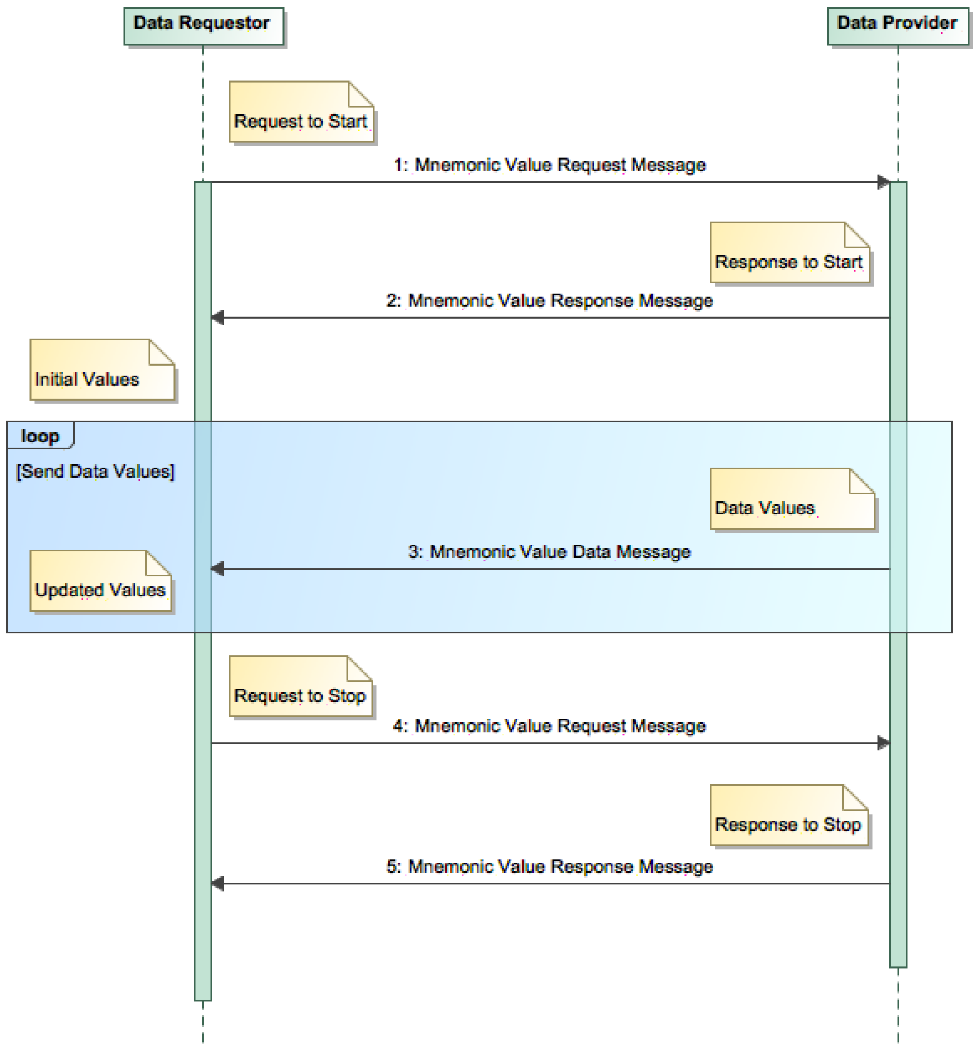

Data Requestor
Data Provider
Request to Start
1: Mnemonic Value Request Message
Response to Start
2: Mnemonic Value Response Message
Initial Values
loop
[Send Data Values]
Data Values
3: Mnemonic Value Data Message
Updated Values
Request to Stop
4: Mnemonic Value Request Message
Response to Stop
5: Mnemonic Value Response Message

Figure 8-21. Mnemonic Value Message Sequence Diagram

The Mnemonic Value Request Message consists of a Message Header and the Message Body Content. The Message Header identifies the message as a C2MS Mnemonic Value Request Message. The message contents specify the number of mnemonics being requested, the type of request, the data sampling criteria, the rate the messages should be published, and the mnemonic names.

It is recommended that the data requestor and data provider use a point-to-point method for data exchange rather than a publish/subscribe approach.

The data requestor (client) would issue a Request to send the Mnemonic Value Request Message (with a Request-Type of Start or Oneshot) and the data provider (server/publisher) would issue a Reply to send the Mnemonic Value Response Message back to the requestor. If the Request-Type was to “Start”, the data provider/publisher/server would publish the Mnemonic Value Data Messages until it was determined they were no longer needed.

To conclude the scenario, the data requestor would then again issue a Request to send the Mnemonic Value Request Message to the data provider with a Request-Type of “Stop”, and the data provider would use the Reply to send the final message in the sequence, the Mnemonic Value Response Message.

Please see Section 6.4 C2MS Messages: Their Characteristics and Interactions for a general discussion on these types of messages.

8.9.1.1 Mnemonic Value Request Message Subjects

Table 8-105. Mnemonic Value Request Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	REQ	MVAL	[DESTINATION-COMPONENT]					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	MVAL	TLM3					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	MVAL	TLM2					
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	REQ	MVAL	TLM3					

Table 8-106. Properties of the *Miscellaneous Elements* for the Mnemonic Value Request Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2	Not used		

Examples

Two components, APP5 and TLM3 interact with the Mnemonic Value Request Message.

TLM3 subscribes to receive the Mnemonic Value Request Message.

`C2MS.*.*.*.*.*.REQ.MVAL.TLM3`

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.*.TLM3` (TLM3 will receive any REQ msg)

APP5 (Data Requestor/Subscriber/Client) sends a request to TLM3, the (Data Provider/Publisher/Server).

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.MVAL.TLM3`

8.9.1.2 Mnemonic Value Request Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Mnemonic Value Request Message.

Table 8-107. Mnemonic Value Request Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	MVAL

8.9.1.3 Mnemonic Value Request Message Contents

The following figure shows a UML object diagram of the Mnemonic Value Request Message with its required, optional, and dependent fields.

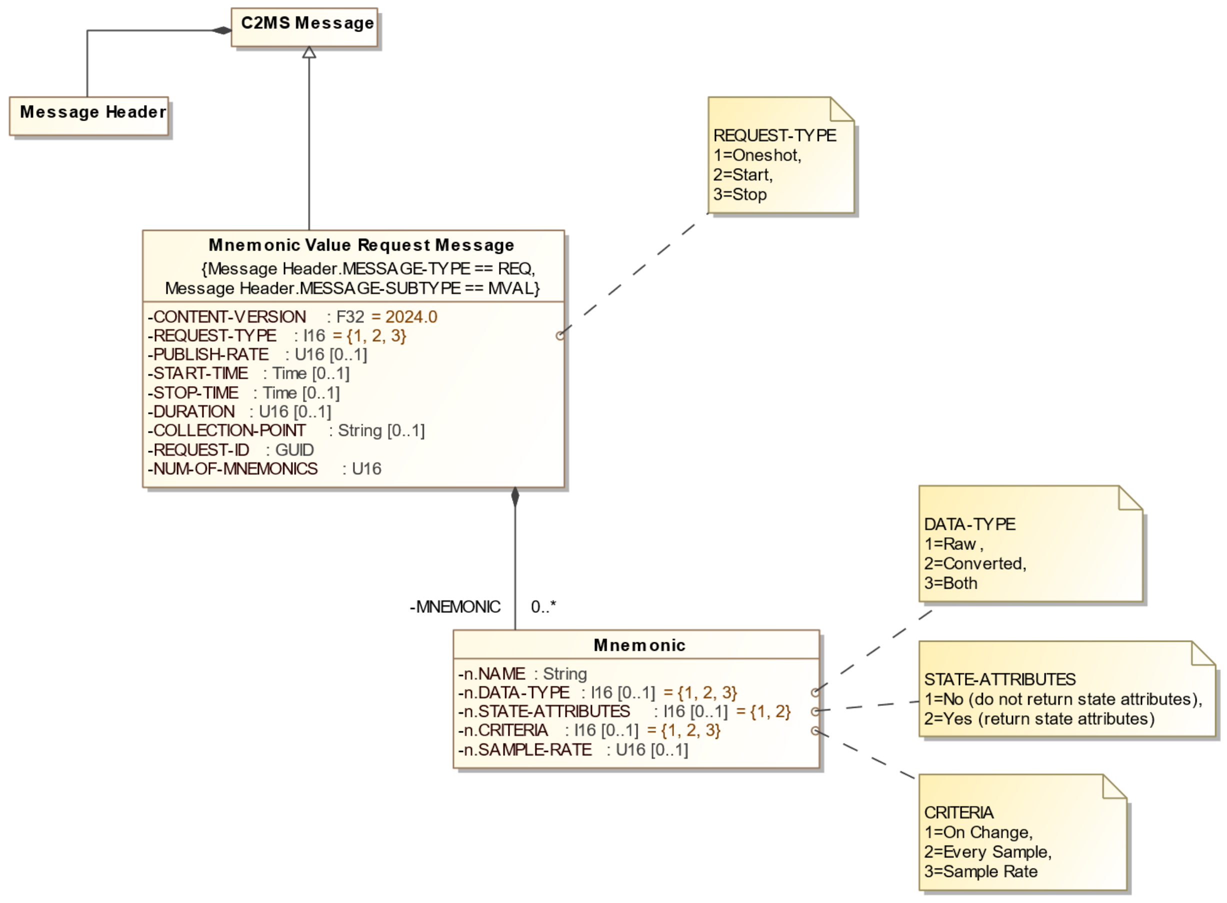

C2MS Message
Message Header
Mnemonic Value Request Message
{Message Header.MESSAGE-TYPE == REQ,
Message Header.MESSAGE-SUBTYPE == MVAL}
-CONTENT-VERSION : F32 = 2024.0
-REQUEST-TYPE : I16 = {1, 2, 3}
-PUBLISH-RATE : U16 [0..1]
-START-TIME : Time [0..1]
-STOP-TIME : Time [0..1]
-DURATION : U16 [0..1]
-COLLECTION-POINT : String [0..1]
-REQUEST-ID : GUID
-NUM-OF-MNEMONICS : U16
REQUEST-TYPE
1=Oneshot,
2=Start,
3=Stop
-MNEMONIC
0..*
Mnemonic
-n.NAME : String
-n.DATA-TYPE : I16 [0..1] = {1, 2, 3}
-n.STATE-ATTRIBUTES : I16 [0..1] = {1, 2}
-n.CRITERIA : I16 [0..1] = {1, 2, 3}
-n.SAMPLE-RATE : U16 [0..1]
DATA-TYPE
1=Raw ,
2=Converted,
3=Both
STATE-ATTRIBUTES
1=No (do not return state attributes),
2=Yes (return state attributes)
CRITERIA
1=On Change,
2=Every Sample,
3=Sample Rate

Figure 8-22. Mnemonic Value Request Message Diagram

The following table describes additional field names, values, and notes for the Mnemonic Value Request Message.

Table 8-108. Mnemonic Value Request Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
REQUEST-TYPE	I16	R	**Value**	**Description**	Identifies the type of mnemonic value request message
			1	Oneshot	
			2	Start	
			3	Stop	
PUBLISH-RATE	U16	O	0+		Identifies the rate, in number of seconds, which the Mnemonic Value Data messages are published. Zero means the server should publish the data as fast as possible. Default rate = 5 seconds.
START-TIME	Time	O			Requested start time of the mnemonic values.
STOP-TIME	Time	O			Requested stop time of the mnemonic values.

Field Name	Type	Presence	Value	Description
DURATION	U16	O	1+	Length of time from “now”, in seconds, for the request to be active, after which the data messages will automatically cease.
COLLECTION-POINT	String	O		Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
REQUEST-ID	GUID	R		ID to identify the request message. If the REQUEST-ID is used here for the first time, REQUEST-TYPE must be "Start" or "Oneshot". However, specifying the same REQUEST-ID from a previous request is used to perform a subsequent "Stop" the previously-established Mnemonic Value subscription.
NUM-OF-MNEMONICS	U16	R	1+ for Oneshot and Start, 0 for Stop	Total number of mnemonics being requested.
MNEMONIC.n.NAME	String	D		Name of the mnemonic - “n” starts at “1”. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.DATA-TYPE	I16	O	**Value** **Description** 1 Raw 2 Converted 3 Both	Indicates the data type to be returned, either the raw value, or the converted value (Engineering Units or Text converted), or both. Defaults to both.
MNEMONIC.n.STATE-ATTRIBUTES	I16	O	**Value** **Description** 1 No 2 Yes	Indicates if State Attributes (flags, limits, static flag, and data quality) are to be returned. Defaults to No.
MNEMONIC.n.CRITERIA	I16	O	**Value** **Description** 1 Change (value, flags, status) 2 Every Sample 3 Sample Rate	Identification of how the data is to be gathered for the mnemonic. Includes either upon change of data (value, flags or status), or every sample, or at a specified sampling rate. The default Criteria is “Change” only data.
MNEMONIC.n.SAMPLE-RATE	U16	D	1+	If CRITERIA is specified as “Sample Rate”, this field will specify the data sampling rate in milliseconds for the mnemonic. SAMPLE-RATE does not affect PUBLISH-RATE but may cause multiple samples to be included each time the data is published.

Oneshot Request

A REQUEST-TYPE of “Oneshot” will result in one set of mnemonic data being returned - the most recent value. No further updates will occur. For a “Oneshot” request the following fields or options **are not** meaningful:

- PUBLISH-RATE
- DURATION
- MNEMONIC.n.CRITERIA
- MNEMONIC.n.SAMPLE-RATE

Start Request

A REQUEST-TYPE of “Start” will result in an initial set of mnemonic data being returned - the most recent value - in the Mnemonic Value Response Message followed by data streaming in a series of Mnemonic Value Data Messages.

Stop Request

A REQUEST-TYPE of “Stop” is used to end a subscription for streaming mnemonic value in the form of Mnemonic Value Data Messages. It will result in a final set of mnemonic data being returned - the most recent value - in the Mnemonic Value Response Message. Note that it is invalid to request a "Stop" where there was no earlier corresponding "Start" request.

REQUEST-ID

This field may be used to submit a "Stop" related to a previous request designated by the same REQUEST-ID. For example, a new REQUEST-ID is used to start a Mnemonic Value subscription and a later request with the same REQUEST-ID can be used to stop the subscription. Issuing a stop in this way stops the flow of all data associated with the original start request, regardless of any information in other fields in the stop request. Once stopped, the flow may not be restarted using the same REQUEST-ID.

It is invalid to reuse the same REQUEST-ID from an earlier request to issue a new request with REQUEST-TYPE "Start" or "Oneshot". Similarly, it is invalid to send a REQUEST-TYPE "Stop" that does not correspond through REQUEST-ID to a REQUEST-TYPE "Start" message.

MNEMONIC.n.CRITERIA

If the MNEMONIC.n.CRITERIA are not specified in the Mnemonic Value Request Message, then the criteria will default to a value of 1 for Change only data.

PUBLISH-RATE

If the publish rate is not specified a default publish rate of 5 seconds will be used. A specified publish rate of zero is a request for the data to be published at the fastest rate possible by the data server/provider. The data provider may have a predetermined maximum publish rate, for example, no faster than 1 message per second, or it may decide to make an on-the-spot calculation of its capabilities based upon its current publishing responsibilities.

For example, the data provider may know that it is limited to an output rate of 1 megabyte per second. If it is currently near its maximum output rate and after calculating the additional load of the request it would exceed that rate, the data provider may reject the Mnemonic Value Request with a “Failed Completion” status in the RESPONSE-STATUS field, and an optional status of “Unable to meet demand” in the RETURN-VALUE field.

START-TIME and STOP-TIME

For some real-time mnemonic data requests, the requestor will need to know the time period of the desired data. As an example, a data requestor of a satellite with a 12-hour pass or even a satellite in constant ground contact will need a means to selectively limit the data it receives, rather than take all the data from the pass. To specify data with a time window, the START-TIME and STOP-TIME parameters are to be used.

DURATION

When a data requestor does not specify any START-TIME or STOP-TIME parameters, a potential issue with the Mnemonic Request, Response, and Value Messages is how to halt the endless publication of messages when the requestor fails to request the cessation of Mnemonic Value Data Messages. This could occur by poor design or through equipment failure where the requestor disappears and is no longer alive to request that publication be halted.

One option is to have the requestor specify the DURATION of time that the Mnemonics Value Messages should be published (the length of a pass in seconds, for example). At the conclusion of this time period the publisher would automatically cease publication. The advantage to this approach is that no matter what the reason for the requestor failing to request a halt to the publication of messages, they will automatically and eventually stop.

Or, the data provider may self-impose a default maximum duration. That is, the provider will only publish mnemonic values for, say a maximum of 30 minutes, or until a stop request is received. (Note that this may not be a good option for non-GUI types of processes.) If a data provider does self-impose a maximum duration, the FINAL-MESSAGE field should be set appropriately in the last message sent.

PUBLISH-RATE, MNEMONIC.n.CRITERIA and MNEMONIC.n.SAMPLE-RATE

There are two concepts of 'rates' in the Mnemonic Value Request Message. The PUBLISH-RATE determines how often the Mnemonic Value Data Message will be created/sent. Independently, the data associated with each mnemonic is gathered at different intervals, including on change, per sample or at a given SAMPLE-RATE. However the data for each mnemonic is gathered, it does not affect the PUBLISH-RATE of the Mnemonic Value Data Message. Rather, there may be multiple samples present for each mnemonic in each published Mnemonic Value Data Message.

8.9.2 Mnemonic Value Response Message

The Mnemonic Value Response Message is used to return the most current mnemonic values requested and to acknowledge receipt of and provide status for a Mnemonic Value Request Message. The Response Status in the Mnemonic Value Response Message indicates the success or failure of the component to process the Mnemonic Value Request Message.

The ordering of the mnemonics in the Mnemonic Value Response shall be the same as the receiving order specified in the Mnemonic Value Request Message. When an invalid mnemonic is detected in the Mnemonic Value Request Message the following shall apply to the Mnemonic Value Response Message:

- Set the RESPONSE-STATUS field to “5 Invalid Request”
- Set the MNEMONIC.n.STATUS field of the invalid mnemonic(s) to “3 Invalid”
- Set the MNEMONIC.n.STATUS field of all other valid mnemonics to “2 Valid, Nodata”
- Set the MNEMONIC.n.NUM-OF-SAMPLES to zero for all mnemonics
- The Mnemonic Value Data Messages shall not be published

The Mnemonic Value Response Message also provides a message time-stamp and the values of the requested mnemonics.

Regardless of the REQUEST-TYPE of the corresponding Mnemonic Value Request Message, the Mnemonic Value Response Message contains current values of the requested mnemonics.

The Number of Samples for a mnemonic can be either zero or one.

- In the case of an invalid mnemonic or a valid mnemonic with no data, the Number of Samples shall be set to zero.
- In the case of a valid mnemonic with a good data sample, the Number of Samples shall be set to one.
- If there is no data available for a mnemonic, the mnemonic status field will indicate a valid mnemonic with a no data condition and the subsequent data value fields will not be provided for this mnemonic.

Table 8-109. Mnemonic Value Response Message Summary

Sender	A C2MS compliant application such as a telemetry decommutation process
Senders Intended Usage	Reply
Receiver	GUI Subsystem, Command Verification process, Expert Subsystem, Analysis subsystem
Receivers Intended Usage	Subscribe
What	Acknowledgment and status of Mnemonic Value Request Message
When	Upon receipt of a Mnemonic Value Request Message

Example

1. Command verification process requests telemetry value to check if spacecraft command executed as expected.

2. Flight Dynamics process requests telemetry values used in the generation of Flight Dynamics products.

8.9.2.1 Mnemonic Value Response Message Subjects

Table 8-110. Mnemonic Value Response Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	MVAL	[DESTINATION-COMPONENT]	[Status]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	MVAL	TLM3	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	MVAL	APP5	1				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	RESP	MVAL	TLM3	*				

Table 8-111. Properties of the *Miscellaneous Elements* for the Mnemonic Value Response Message

Miscellaneous Element	Required / Optional	Description	Field Origination in Msg, if applicable
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components FD (the Data Requestor/Subscriber/Client) and TLM3 (the Data Provider/Publisher/Server) interact with the Mnemonic Value Response Message.

FD subscribe subject to receive the Mnemonic Value Response Message.

```
C2MS.*.*.MSSN.*.*.RESP.MVAL.FD.>    or

C2MS.*.*.*.*.*.RESP.MVAL.FD.>
```

TLM3 sends a response message to FD.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.MVAL.FD.2    or

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.MVAL.FD.4
```

8.9.2.2 Mnemonic Value Response Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Mnemonic Value Response Message.

Table 8-112. Mnemonic Value Response Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	MVAL

8.9.2.3 Mnemonic Value Response Message Contents

The following figure shows a UML object diagram of the Mnemonic Value Response Message with its required and optional fields.

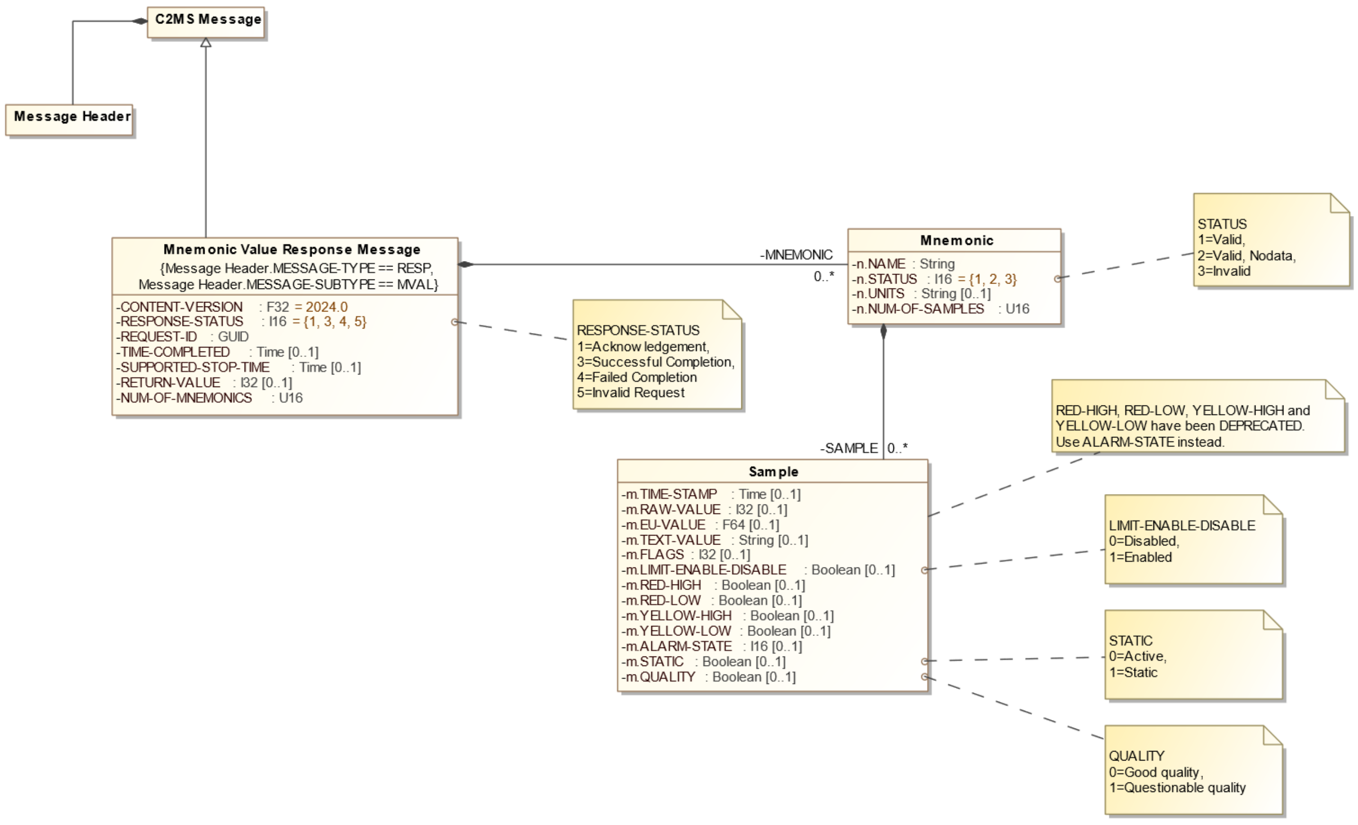

C2MS Message
Message Header
Mnemonic Value Response Message
{Message Header.MESSAGE-TYPE == RESP, Message Header.MESSAGE-SUBTYPE == MVAL}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1, 3, 4, 5}
-REQUEST-ID : GUID
-TIME-COMPLETED : Time [0..1]
-SUPPORTED-STOP-TIME : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-NUM-OF-MNEMONICS : U16
RESPONSE-STATUS
1=Acknowledgement,
3=Successful Completion,
4=Failed Completion
5=Invalid Request
-MNEMONIC
0..*
Mnemonic
-n.NAME : String
-n.STATUS : I16 = {1, 2, 3}
-n.UNITS : String [0..1]
-n.NUM-OF-SAMPLES : U16
STATUS
1=Valid,
2=Valid, Nodata,
3=Invalid
-SAMPLE
0..*
Sample
-m.TIME-STAMP : Time [0..1]
-m.RAW-VALUE : I32 [0..1]
-m.EU-VALUE : F64 [0..1]
-m.TEXT-VALUE : String [0..1]
-m.FLAGS : I32 [0..1]
-m.LIMIT-ENABLE-DISABLE : Boolean [0..1]
-m.RED-HIGH : Boolean [0..1]
-m.RED-LOW : Boolean [0..1]
-m.YELLOW-HIGH : Boolean [0..1]
-m.YELLOW-LOW : Boolean [0..1]
-m.ALARM-STATE : I16 [0..1]
-m.STATIC : Boolean [0..1]
-m.QUALITY : Boolean [0..1]
RED-HIGH, RED-LOW, YELLOW-HIGH and YELLOW-LOW have been DEPRECATED. Use ALARM-STATE instead.
LIMIT-ENABLE-DISABLE
0=Disabled,
1=Enabled
STATIC
0=Active,
1=Static
QUALITY
0=Good quality,
1=Questionable quality

Figure 8-23. Mnemonic Value Response Message Diagram

The following table describes additional field names, values, and notes for the Mnemonic Value Response Message.

Table 8-113. Mnemonic Value Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the Mnemonic Value Request Message that was processed. (“2” is not a valid value and has no meaning.)
			1	Acknowledgement	
			3	Successful Completion	
			4	Failed Completion	
			5	Invalid Request	
REQUEST-ID	GUID	R			This field’s value is to be the same as the REQUEST-ID in the associated REQ message.
TIME-COMPLETED	Time	O			Time application completed processing the request

Field Name	Type	Presence	Value	Description
SUPPORTED-STOP-TIME	Time	O		The service's calculated stop time. This is based on the DURATION or STOP-TIME specified in the request, but may be limited to what the service can support. For example, if the REQUEST specified a STOP-TIME a year in the future or a DURATION of 100 years, the service may calculate a reasonable SUPPORTED-STOP-TIME and return that value in this field. Reasonability is as determined by the service. This alerts the requestor that Mnemonic Value Data Messages will not be sent after the SUPPORTED-STOP-TIME.
RETURN-VALUE	I32	O		Return value or status based on the RESPONSE-STATUS. Useful to provide function call status or error code in the case of failed completion
NUM-OF-MNEMONICS	U16	R		Total number of mnemonics returned. Should echo the “NUM-OF-MNEMONICS” field in the request message.
MNEMONIC.n.NAME	String	D		Name of the ‘nth’ Mnemonic - “n” starts at “1”. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.STATUS	I16	D	**Value** **Description** 1 Valid 2 Valid, Nodata 3 Invalid	Status of the ‘nth’ mnemonic: valid mnemonic or valid mnemonic with no data or invalid mnemonic. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.UNITS	String	O		Units associated with the value converted to engineering units for the ‘nth’ mnemonic
MNEMONIC.n.NUM-OF-SAMPLES	U16	D		Number of data samples for the ‘nth’ mnemonic. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.SAMPLE.m.TIME-STAMP	Time	O		Time stamp for the first data sample of the ‘nth’ mnemonic - both “n” and "m" start at “1”.
MNEMONIC.n.SAMPLE.m.RAW-VALUE	I32	O		Raw value for the first data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.EU-VALUE	F64	O		Raw value converted to Engineering Units if engineering units conversion is present for

Field Name	Type	Presence	Value		Description
					the first data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.TEXT-VALUE	String	O			Raw value converted to a text string if text conversion is present for the first data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.FLAGS	I32	O			Flags native to the T&C component for the first data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.LIMIT-ENABLE-DISABLE	Boolean	O	**Value**	**Description**	Indicates the limit checking state for the first data sample of the ‘nth’ mnemonic
			0	Disabled	
			1	Enabled	
DEPRECATED MNEMONIC.n.SAMPLE.m.RED-HIGH	Boolean	O	**Value**	**Description**	Indicates the Red High limit status of the first data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE.m.RED-LOW	Boolean	O	**Value**	**Description**	Indicates the Red Low limit status of the first data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE.m.YELLOW-HIGH	Boolean	O	**Value**	**Description**	Indicates the Yellow High limit status of the first data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE.m.YELLOW-LOW	Boolean	O	**Value**	**Description**	Indicates the Yellow Low limit status of the first data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
MNEMONIC.n.SAMPLE.m.ALARM-STATE	I16	O	**Value**	**Description**	Indicates the limit state of the ‘mth’ data sample of the ‘nth’ mnemonic
			0	Unavailable	
			1	Normal	
			2	Warning	
			3	Distress	
			4	Critical	
MNEMONIC.n.SAMPLE.m.STATIC	Boolean	O	**Value**	**Description**	Indicates the static (stale) condition of the first data sample of the ‘nth’ mnemonic
			0	Active	
			1	Static	
MNEMONIC.n.SAMPLE.m.QUALITY	Boolean	O	**Value**	**Description**	Indicates the Quality of the first data sample of the ‘nth’ mnemonic
			0	Good quality	
			1	Questionable quality	

The following data attributes are not included in the Mnemonic Value Response or Mnemonic Value Data messages: Delta Limits, Rail Limits, Inverted Limits, and Foreground / Background colors for text values.

8.9.3 Mnemonic Value Data Message

The Mnemonic Value Data Message provides the telemetry or configuration mnemonic data that was requested in the Mnemonic Value Request Message.

The message is generated in response to receiving a Mnemonic Value Request Message to “start” publishing the Mnemonic values for one or more mnemonics and following the generation of a Mnemonic Value Response Message with successful completion status.

The Mnemonic Value Data Messages shall not be published if the Mnemonic Value Request Message contained any invalid mnemonics.

The Mnemonic Value Data Message will continue to be published at the requested distribution rate until a “Stop” Mnemonic Value Request Message is received, or the DURATION specified in the Mnemonic Value Request Message has expired.

The ordering of the mnemonics in the Mnemonic Value Data Message shall be the same as the receiving order specified in the Mnemonic Value Request Message.

The Mnemonic Value Data Message will contain one or more mnemonics and one or more data samples per mnemonic.

If there is no data available for a mnemonic, the mnemonic status field will indicate a valid mnemonic with a no data condition and the subsequent data value fields will not be provided for this mnemonic.

Table 8-114. Mnemonic Value Data Message Summary

Sender	A C2MS compliant application such as a telemetry decommutation process
Senders Intended Usage	Publish
Receiver	GUI Subsystem, Command Subsystem, Schedule Execution process, Expert Subsystem
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data or configuration data values
When	Upon interval requested at a minimum and dependent on data rate and/or replay rate and/or change rate

Example

1. Telemetry data to be displayed on a GUI page

2. Telemetry data for an analysis plot

8.9.3.1 Mnemonic Value Data Message Subjects

Table 8-115. Mnemonic Value Data Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	MVAL	[COMPONENT]	[request id]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	MVAL	TLM3	[GUID]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	MVAL	TLM2	[GUID]				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	MVAL	TLM3	[GUID]				

Note that "[GUID]" in the table above is a Subject Token String that conforms to the GUID type specified in this document, such as "b891bdac-964a-4f3e-957c-1a29ec4c7d50" or similar.

Table 8-116. Properties of the *Miscellaneous Elements* for the Mnemonic Value Data Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	ID associated with original request	REQUEST-ID
ME3	Not used		

Examples

After the successful exchange of the Mnemonic Value Request and Response Messages, the Mnemonic Value Data Message is published.

The Requestor/Subscriber/Client subscribes to receive the intended Mnemonic Value Data Messages.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.MVAL.*
```

The Data Provider/Publisher/Server sends out the Mnemonic Value Data Message with the following subject:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.MVAL.TLM3

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.MVAL.TLM2
```

8.9.3.2 Mnemonic Value Data Message Header

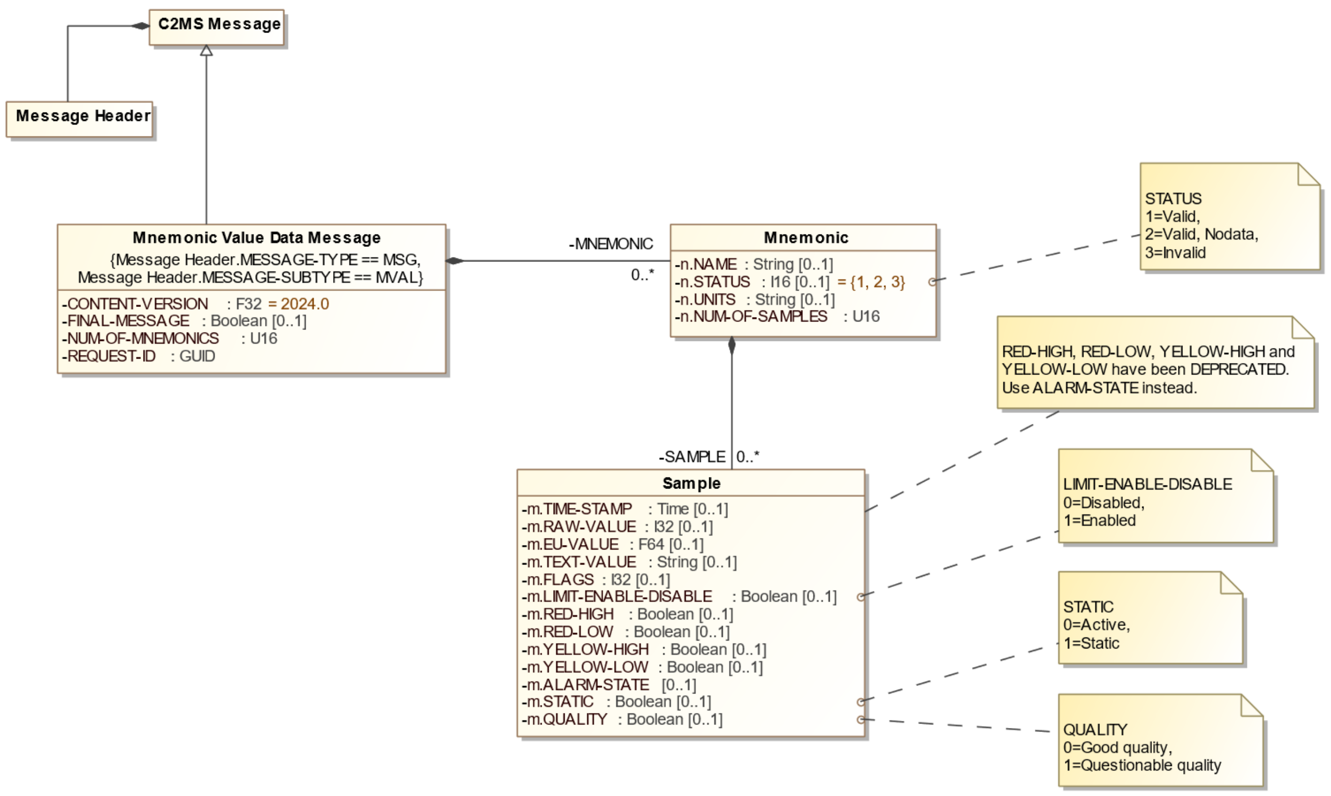
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Mnemonic Value Data Message.

Table 8-117. Mnemonic Value Data Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	MVAL

8.9.3.3 Mnemonic Value Data Message Contents

The following figure shows a UML object diagram of the Mnemonic Value Data Message with its required and optional fields.



C2MS Message
Message Header
Mnemonic Value Data Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == MVAL}
-CONTENT-VERSION : F32 = 2024.0
-FINAL-MESSAGE : Boolean [0..1]
-NUM-OF-MNEMONICS : U16
-REQUEST-ID : GUID
-MNEMONIC
0..*
Mnemonic
-n.NAME : String [0..1]
-n.STATUS : I16 [0..1] = {1, 2, 3}
-n.UNITS : String [0..1]
-n.NUM-OF-SAMPLES : U16
STATUS 1=Valid, 2=Valid, Nodata, 3=Invalid
RED-HIGH, RED-LOW, YELLOW-HIGH and YELLOW-LOW have been DEPRECATED. Use ALARM-STATE instead.
-SAMPLE 0..*
Sample
-m.TIME-STAMP : Time [0..1]
-m.RAW-VALUE : I32 [0..1]
-m.EU-VALUE : F64 [0..1]
-m.TEXT-VALUE : String [0..1]
-m.FLAGS : I32 [0..1]
-m.LIMIT-ENABLE-DISABLE : Boolean [0..1]
-m.RED-HIGH : Boolean [0..1]
-m.RED-LOW : Boolean [0..1]
-m.YELLOW-HIGH : Boolean [0..1]
-m.YELLOW-LOW : Boolean [0..1]
-m.ALARM-STATE [0..1]
-m.STATIC : Boolean [0..1]
-m.QUALITY : Boolean [0..1]
LIMIT-ENABLE-DISABLE 0=Disabled, 1=Enabled
STATIC 0=Active, 1=Static
QUALITY 0=Good quality, 1=Questionable quality

Figure 8-24. Mnemonic Value Data Message Diagram

The following table describes additional field names, values, and notes for the Mnemonic Value Data Message.

Table 8-118. Mnemonic Value Data Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
FINAL-MESSAGE	Boolean	O			When true, indicates last message in the series.
NUM-OF-MNEMONICS	U16	R			Total number of mnemonics in this message
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
MNEMONIC.n.NAME	String	O			Name of the 'nth' mnemonic - "n" starts at "1".
MNEMONIC.n.STATUS	I16	O	**Value**	**Description**	Status of the 'nth' mnemonic: valid mnemonic, or valid mnemonic with no data, or invalid mnemonic
			1	Valid	
			2	Valid, Nodata	
			3	Invalid	

Field Name	Type	Presence	Value		Description
MNEMONIC.n.UNITS	String	O			Units associated with the raw value converted to engineering units for the ‘nth’ mnemonic
MNEMONIC.n.NUM-OF-SAMPLES	U16	D			Number of data samples for the ‘nth’ mnemonic. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.SAMPLE.m.TIME-STAMP	Time	O			Time stamp for the ‘nth’ data sample of the ‘nth’ mnemonic - both “n” and "m" start at “1”.
MNEMONIC.n.SAMPLE.m.RAW-VALUE	I32	O			Raw value for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.EU-VALUE	F64	O			Raw value converted to Engineering Units, if engineering units conversion is present for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.TEXT-VALUE	String	O			Raw value converted to a text string if text conversion is present for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.FLAGS	I32	O			Flags native to the T&C component for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE.m.LIMIT-ENABLE-DISABLE	Boolean	O	**Value**	**Description**	Indicates the limit checking state for the ‘nth’ data sample of the ‘nth’ mnemonic
			0	Disabled	
			1	Enabled	
DEPRECATED MNEMONIC.n.SAMPLE.m.RED-HIGH	Boolean	O	**Value**	**Description**	Indicates the Red High limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE.m.RED-LOW	Boolean	O	**Value**	**Description**	Indicates the Red Low limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE.m.YELLOW-HIGH	Boolean	O	**Value**	**Description**	Indicates the Yellow High limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE.m.YELLOW-LOW	Boolean	O	**Value**	**Description**	Indicates the Yellow Low limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
MNEMONIC.n.SAMPLE.m.ALARM-STATE	I16	O	**Value**	**Description**	Indicates the limit state of the ‘mth’ data sample of the ‘nth’ mnemonic
			0	Unavailable	

Field Name	Type	Presence	Value		Description
			1	Normal	
			2	Warning	
			3	Distress	
			4	Critical	
MNEMONIC.n.SAMPLE.m.STATIC	Boolean	O	**Value**	**Description**	Indicates the static (stale) condition for the ‘nth’ data sample of the ‘nth’ mnemonic
			0	Active	
			1	Static	
MNEMONIC.n.SAMPLE.m.QUALITY	Boolean	O	**Value**	**Description**	Indicates the Quality for the ‘nth’ data sample of the ‘nth’ mnemonic
			0	Good quality	
			1	Questionable quality	

8.10 Archive Mnemonic Value Messages

The Archive Mnemonic Value Messages provide a mechanism for requesting and delivering mnemonic data that has been stored in an archive.

A requesting component, such as a trending system, may request a set of mnemonics from the data archive. The request is generally for a set of values over an interval of time and at a desired data-sampling rate.

The responding component, such as an Archive system, extracts the set of requested mnemonics from the data archive and provides them to the requesting component via a variety of available delivery methods.

The Archive Mnemonic Value Messages remove the burden from the requesting component of having to process (decommutate) the data from the archive and therefore having to know the specifics of the telemetry database and the structure of the data archive. The Archive Mnemonic Value Messages also remove the burden from the requesting component from hardware and software associated with the creation, management, and maintenance of a data archive.

The three specific Archive Mnemonic Value messages are:

- Archive Mnemonic Value Request Message
- Archive Mnemonic Value Response Message
- Archive Mnemonic Value Data Message

8.10.1 Archive Mnemonic Value Request Message

The Archive Mnemonic Value Request Message is used when an application requires mnemonic data from previously recorded data that has been stored in the Telemetry Archive.

A common use of the Archive Mnemonic Value Request Message is when an analysis component produces a trending product, such as Battery charge/discharge, or a propellant graph. The analysis component builds a request of the mnemonics to be retrieved from the telemetry archive and sends this request to an Archive Management component. The Archive Management Component may be a stand-alone component or it may be part of another subsystem, such as a T&C subsystem.

The Archive Management component will package the decommutated values and their associated attributes (if desired) for all the requested mnemonics over the time frame requested.

A number of extraction options and delivery methods are available. These include:

Time interval

- The requestor is required to specify the time span of the desired data.

Data selection

- The requested can be for raw, converted, or both types of data.
- Attributes (flags, limits, static, quality) can be included or not.
- Data sampling can specify all, upon change, or at a periodic sample rate.

Data delivery method

- One single response message, *or*

- As a stream of messages similar to a real-time mnemonic data value. If this delivery method is selected, the requestor can also select the speed of the data delivery, either as kilobits per second or as a ratio of the real-time rate.

Actual data or by reference

- The data can be within the response message or a URI can reference the location of a data file.

If the requestor has asked for a data file, the attributes of the file can be further specified.

- Using product specifications, and
- File specifications

In summary, a variety of data selection criteria and data delivery mechanisms are available to customize and best match the needs of the requestor.

Many of the fields in the Archive Mnemonic Value Request Message are optional and dependent on other fields, but they also have specified default values so that only a minimal number of fields need be actually specified to extract and deliver the data.

Of course, the more specific and customized the data request, the more fields that will need to be specified.

Table 8-119. Archive Mnemonic Value Request Message Summary

Sender	A C2MS compliant application such as an Analysis and Assessment component
Senders Intended Usage	Request
Receiver	Any Archive mnemonic processor such as an Archive Management component or a T&C component
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data, ground configuration data, or any data stored with a mnemonic name
When	As needed

Example

1. An Analysis process requests telemetry values to create a graph of a spacecraft instrument's performance.

2. A Flight Dynamics process requests telemetry values used in the generation of Flight Dynamics products.

The Archive Mnemonic Value Request Message consists of a Message Header and the message content.

The Message Header identifies the message as a C2MS Archive Mnemonic Value Request Message.

The message content specifies the time range, the data sampling criteria, and the mnemonic names. The message content also provides for the specification of the delivery method of the data.

8.10.1.1 Archive Mnemonic Value Request Message Subjects

Table 8-120. Archive Mnemonic Value Request Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	REQ	AMVAL	[DESTINATION-COMPONENT]					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	AMVAL	TLM3					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	AMVAL	TLM2					
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	REQ	AMVAL	>					

Table 8-121. Properties of the *Miscellaneous Elements* for the Archive Mnemonic Value Request Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2	Not used		

Examples

Two components, FD (Data Requestor/Subscriber/Client) and TLM3 (Data Provider/Publisher/Server), interact with the Archive Mnemonic Value Request Message.

FD sends the Archive Mnemonic Value Request Message to TLM3.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.AMVAL.TLM3
```

TLM3 subscribes to receive the Archive Mnemonic Value Request Message from FD.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.AMVAL.TLM3
```

or

```
C2MS.*.*.*.*.*.REQ.AMVAL.>
```

8.10.1.2 Archive Mnemonic Value Request Message Header

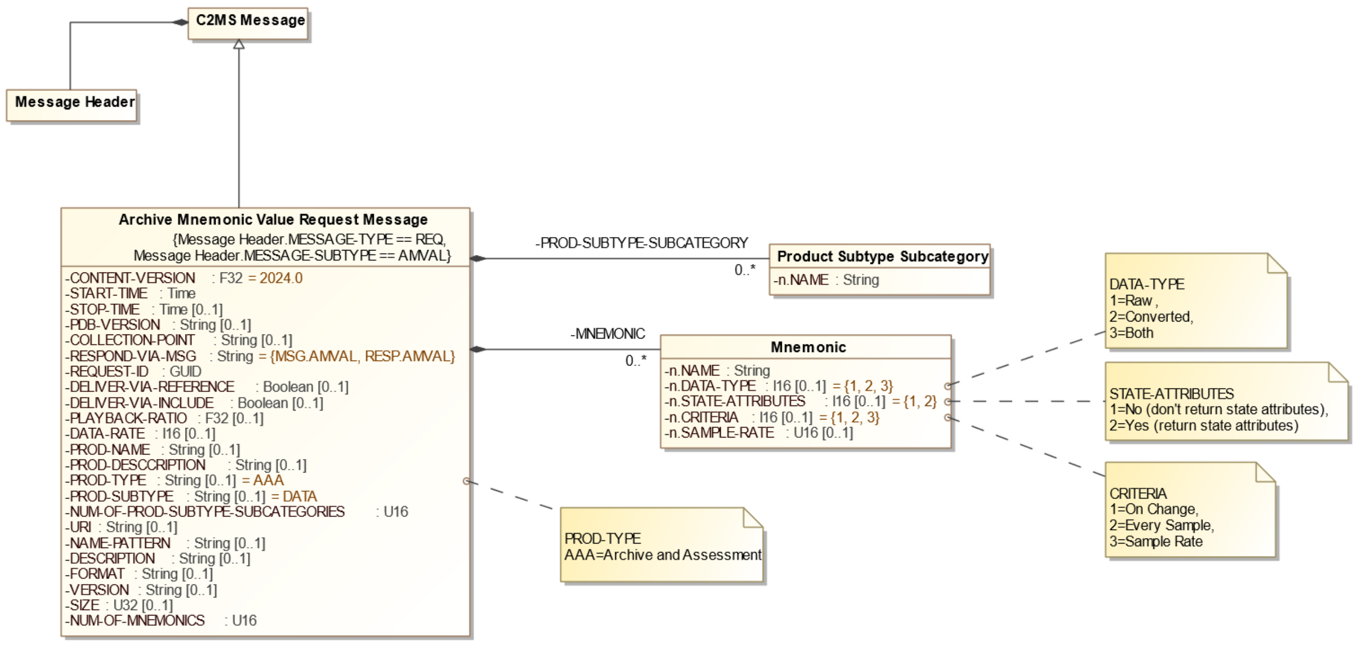
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Archive Mnemonic Value Request Message header.

Table 8-122. Archive Mnemonic Value Request Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	AMVAL

8.10.1.3 Archive Mnemonic Value Request Message Contents

The following figure shows a UML object diagram of the Archive Mnemonic Value Request Message with its required, optional, and dependent fields.



C2MS Message
Message Header
Archive Mnemonic Value Request Message
{Message Header.MESSAGE-TYPE == REQ,
Message Header.MESSAGE-SUBTYPE == AMVAL}
-CONTENT-VERSION : F32 = 2024.0
-START-TIME : Time
-STOP-TIME : Time [0..1]
-PDB-VERSION : String [0..1]
-COLLECTION-POINT : String [0..1]
-RESPOND-VIA-MSG : String = {MSG.AMVAL, RESP.AMVAL}
-REQUEST-ID : GUID
-DELIVER-VIA-REFERENCE : Boolean [0..1]
-DELIVER-VIA-INCLUDE : Boolean [0..1]
-PLAYBACK-RATIO : F32 [0..1]
-DATA-RATE : I16 [0..1]
-PROD-NAME : String [0..1]
-PROD-DESCCRIPTION : String [0..1]
-PROD-TYPE : String [0..1] = AAA
-PROD-SUBTYPE : String [0..1] = DATA
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-URI : String [0..1]
-NAME-PATTERN : String [0..1]
-DESCRIPTION : String [0..1]
-FORMAT : String [0..1]
-VERSION : String [0..1]
-SIZE : U32 [0..1]
-NUM-OF-MNEMONICS : U16
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
-MNEMONIC
0..*
Mnemonic
-n.NAME : String
-n.DATA-TYPE : I16 [0..1] = {1, 2, 3}
-n.STATE-ATTRIBUTES : I16 [0..1] = {1, 2}
-n.CRITERIA : I16 [0..1] = {1, 2, 3}
-n.SAMPLE-RATE : U16 [0..1]
PROD-TYPE
AAA=Archive and Assessment
DATA-TYPE
1=Raw ,
2=Converted,
3=Both
STATE-ATTRIBUTES
1=No (don't return state attributes),
2=Yes (return state attributes)
CRITERIA
1=On Change,
2=Every Sample,
3=Sample Rate

Figure 8-25. Archive Mnemonic Value Request Message Diagram

The following table describes additional field names, values, and notes for the Archive Mnemonic Value Request Message.

Table 8-123. Archive Mnemonic Value Request Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
START-TIME	Time	R		Requested start time of the mnemonic values to be retrieved from the telemetry archive.
STOP-TIME	Time	O		Requested stop time of the mnemonic values to be retrieved from the telemetry archive. Defaults to the end of the telemetry archive
PDB-VERSION	String	O		Project Database version to be used by the responder when processing the archived data. Defaults to the PDB version used when the data was archived.
COLLECTION-POINT	String	O		Receiver, device, point, path, etc. where data was received. Used to distinguish data simultaneously received at multiple collection points.
RESPOND-VIA-MSG	String	R	**Value** | **Description** “MSG.AMVAL” | AMVAL Message	Indicates the message to use to deliver the mnemonic data. MSG will

Field Name	Type	Presence	Value		Description
			"RESP.AMVAL"	AMVAL Response Message	be a stream of messages; RESP will be a single response message.
REQUEST-ID	GUID	R			ID to identify the request message
DELIVER-VIA-REFERENCE	Boolean	D			This parameter is used only if "RESP.AMVAL" is selected above. Indicates if the data will be referenced by a URI in the single response message. Defaults to False.
DELIVER-VIA-INCLUDE	Boolean	D			This parameter is used only if "RESP.AMVAL" is selected above. Indicates if the data is to be included in the single response message. Defaults to True.
PLAYBACK-RATIO	F32	D	> 0 and < 1 is slower than real-time rate = 1 is equal to the real-time rate > 1 is faster than real-time rate		If "MSG.AMVAL" is selected above, specifies the speed of data delivery as a ratio of playback rate to real-time rate.
DATA-RATE	I16	D	> 0		If "MSG.AMVAL" is selected above, specifies the speed of data delivery in Kilobits per second
PROD-NAME	String	D			The "PROD-" fields are optionally used when the "RESP.AMVAL" has been specified above. Name of the product being requested.
PROD-DESCRIPTION	String	O			Description of the product in text or xml
PROD-TYPE	String	D	**Value**	**Description**	Product type and subtype being requested. (See Table A-2 Product Categories.)
			AAA	Archive and Assessment	
PROD-SUBTYPE	String	D	DATA		
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R			Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6*, etc. in Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D			First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc. of the Product Message) - "n" starts at "1". This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
URI	String	D			Location where the requesting component is asking for the product file(s) to be stored. Could be a web address, directory, or folder specification
NAME-PATTERN	String	D			Describes the name of the output file
DESCRIPTION	String	D			Description of the file in text or xml
FORMAT	String	D			Describes the file format
VERSION	String	D			Identifies the version of the file

Field Name	Type	Presence	Value	Description
SIZE	U32	D	Kilobytes	Maximum size of the file acceptable to the requester.
NUM-OF-MNEMONICS	U16	R		Total number of mnemonics being requested
MNEMONIC.n.NAME	String	D		Name of the mnemonic - "n" starts at "1". This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.DATA-TYPE	I16	O	**Value** / **Description** 1 / Raw 2 / Converted 3 / Both	Indicates the data type to be returned, either the raw value, or the converted value (Engineering Units or Text converted), or both. Defaults to both.
MNEMONIC.n.STATE-ATTRIBUTES	I16	O	**Value** / **Description** 1 / No 2 / Yes	Indicates if the State Attributes (flags, limits, static flag, and data quality) of the mnemonic are to be returned. Defaults to No.
MNEMONIC.n.CRITERIA	I16	O	**Value** / **Description** 1 / Change (value, flags, status) 2 / Every Sample 3 / Sample Rate	Identification of how data should be sampled for the mnemonic. Includes either upon change of data (value, flags, or status), or every sample, or at a specified sampling rate. Defaults to "Change" only data.
MNEMONIC.n.SAMPLE-RATE	U16	D	1+	If CRITERIA is specified as "Sample Rate", this field will specify the data sampling rate for the mnemonic.in milliseconds

For an explanation on how the START-TIME and STOP-TIME could operate, see Table 8-20 Examples of Start and Stop Times.

The archived mnemonic data can be delivered in a stream of messages (as described below in "Stream of Messages Data Delivery"), akin to the stream of real-time mnemonic data value messages; or a single response message (as described below in "Single Response Message Data Delivery").

STREAM OF MESSAGES DATA DELIVERY

The advantage of delivering the archived mnemonic data as a stream of messages is that the processing can be similar if not identical to the procedure used with the real-time Mnemonic Value Data Messages. To use this delivery mechanism, specify the following:

- Set the field RESPOND-VIA-MSG to the value "MSG.AMVAL".
- Optionally, specify the speed of data delivery with either of the fields "PLAYBACK-RATIO" or "DATA-RATE".
- Identify the number and names of the mnemonics along with their extraction criteria.

SINGLE RESPONSE MESSAGE DATA DELIVERY

The advantage of delivering the archived mnemonic data in a single response message is that the data is entirely contained within one location and can be processed in bulk. When using this data delivery mechanism, the requestor has a few options on how and where the data is to be delivered. The requested data will be delivered all at once. The requestor can ask for the data:

1. Within the single response message

2. By reference, using a URI within the single response message

3. Both methods

To accomplish the desired result, each of these options is explained below.

FIRST, for all the above options using a single response message:

- Set the field RESPOND-VIA-MSG to the value "RESP.AMVAL"

SECOND, choose one of the three following data delivery options:

1. **INCLUDE WITHIN MESSAGE**: To include only the data within the single response message (with no URI reference), the requestor should do the following:

    - Nothing. The "DELIVER-VIA-" fields will default to include the data within the single response message with no URI reference. No other field under the Product Distribution Options section is required to be specified.

2. **BY REFERENCE**: To only have the data file specified by reference and NOT be included in the single response message, the requestor should do the following:

    - Set the field DELIVER-VIA-REFERENCE to the value "Yes/True".

    - Set the field DELIVER-VIA-INCLUDE to the value "No/False".

3. **BOTH**: To have both the data included in the single response message AND specified as a reference the requestor should do the following:

    - Set the field DELIVER-VIA-REFERENCE to the value "Yes/True" (The field DELIVER-VIA-INCLUDE will default to the value "Yes/True").

THIRD, for all the options above, the requestor must do the following:

- Identify the number and names of the mnemonics along with their extraction criteria..

Other features of note for this request message are:

- When specifying the mnemonics, if the MNEMONIC.n.CRITERIA is not specified in the Archive Mnemonic Value Request Message, then the criteria will default to a value of 1 for Change only data.

- If a URI has been specified in the request, it is assumed that the DELIVER-VIA-REFERENCE field was set to "Yes/True". If the URI was specified, the resulting archive mnemonic value product will be "pushed" to the location specified by the URI.

- If the URI has not been specified in the request, but the DELIVER-VIA-REFERENCE field was set to "Yes/True", then the resulting archive mnemonic value product will be copied to a URI location designated by the provider of the data. This URI location must also be included in the Archive Mnemonic Value Response Message. The requestor of the data file can "pull" the file using the provided URI.

- If the component servicing the Archive Mnemonic Value Request Message supports several formats or versions of a product, then the requesting component can specify the format or version of the resulting product in the FORMAT and VERSION fields. If the FORMAT and/or VERSION fields are not specified, then the component servicing the Archive Mnemonic Value Request message shall default to the latest format or version of its product.

8.10.2 Archive Mnemonic Value Response Message

An Archive Provider in response to an Archive Mnemonic Value Request Message sends an Archive Mnemonic Value Response Message.

The job of the Archive Mnemonic Value Response Message is to provide acknowledgment of the Archive Mnemonic Value Request Message, the overall status of the completed action, the specific status of each requested mnemonic, and optionally, the resulting data and associated attributes.

A series of Archive Mnemonic Value Response Messages may be required. In this case, an initial acknowledgement response message is issued, followed by interim or interactive "working" response type messages to let the requesting application know that the request is still being processed, and finally a completion response type message.

If an audit trail or operator notification is required, the requesting application is responsible for generating a Log Message indicating the result of the Archive Mnemonic Value Request Message.

Please see Section 6.4 C2MS Messages: Their Characteristics and Interactions for a general discussion on these types of messages.

Table 8-124. Archive Mnemonic Value Response Message Summary

Sender	A C2MS compliant application that has access to a telemetry archive such as a T&C component
Senders Intended Usage	Reply
Receiver	Assessment and Analysis component
Receivers Intended Usage	Subscribe
What	Acknowledgment and status of Archive Mnemonic Value Request Message
When	Upon receipt of an Archive Mnemonic Value Request Message and/or completion of the Request

Examples

1. An Archive Manager responds to a component in the Assessment and Analysis subsystem that requested battery telemetry values to check the spacecraft battery rate of charge/discharge.

2. The Archive Manager responds to a component in the Flight Dynamics subsystem that requested telemetry values to be used in the generation of a Flight Dynamics product.

8.10.2.1 Archive Mnemonic Value Response Message Subjects

Table 8-125. Archive Mnemonic Value Response Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	AMVAL	[DESTINATIO N-COMPONEN T]	[Response Status: 1-ack, …4-failed]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	AMVAL	FD	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	AMVAL	FD	1				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	RESP	AMVAL	*	>				

Table 8-126. Properties of the *Miscellaneous Elements* for the Archive Mnemonic Value Response Message

Miscellaneous Element	Required / Optional	Description	Field Origination in Msg, if applicable
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components, TLM2 (Data Provider/Publisher/Server) and FD (Data Requestor/Subscriber/Client), interact with the Archive Mnemonic Value Response Message.

TLM2 subject name to send two Archive Mnemonic Value Response Messages to FD.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.AMVAL.FD.2
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.AMVAL.FD.3
```

FD subscribes to receive the Archive Mnemonic Value Response Message from TLM2.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.AMVAL.FD.>
```

8.10.2.2 Archive Mnemonic Value Response Message Header

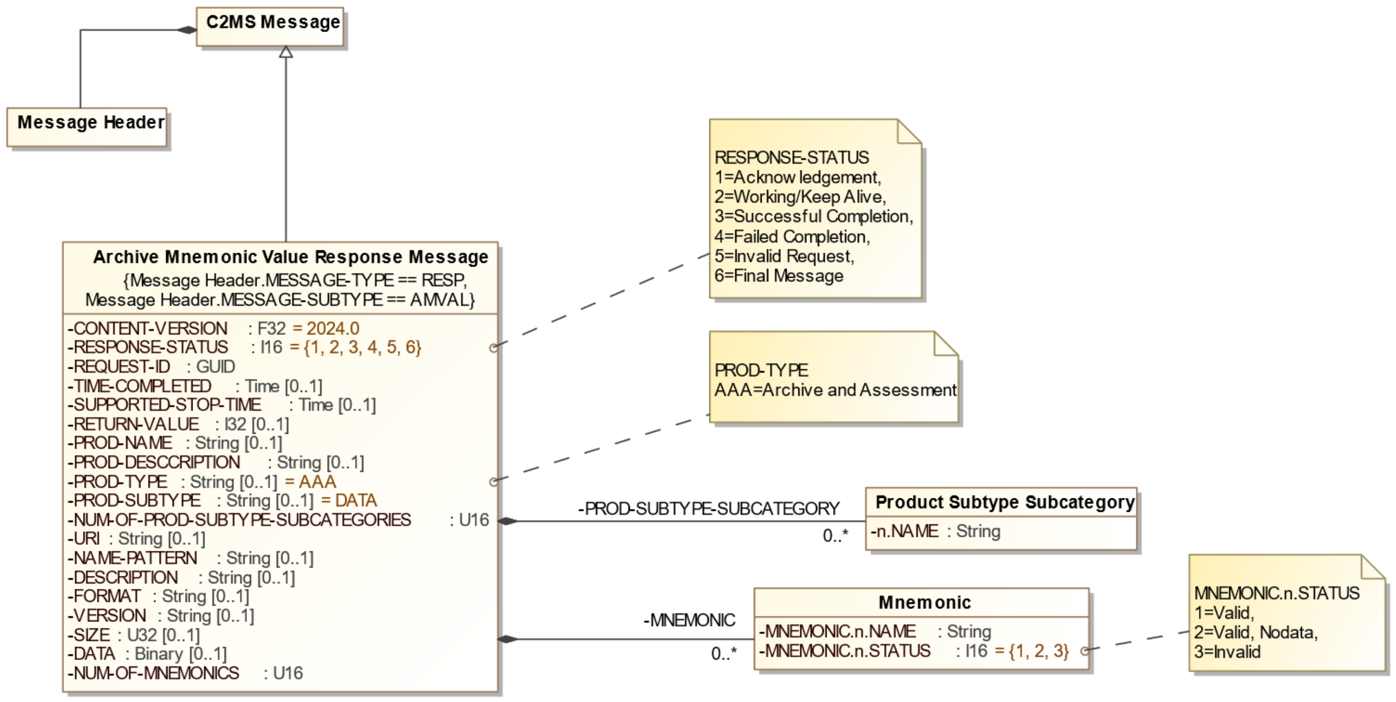
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Archive Mnemonic Value Response Message.

Table 8-127. Archive Mnemonic Value Response Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	AMVAL

8.10.2.3 Archive Mnemonic Value Response Message Contents

The following figure shows a UML object diagram of the Archive Mnemonic Value Request Message with its required, optional, and dependent fields.



C2MS Message
Message Header
Archive Mnemonic Value Response Message
{Message Header.MESSAGE-TYPE == RESP,
Message Header.MESSAGE-SUBTYPE == AMVAL}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1, 2, 3, 4, 5, 6}
-REQUEST-ID : GUID
-TIME-COMPLETED : Time [0..1]
-SUPPORTED-STOP-TIME : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-PROD-NAME : String [0..1]
-PROD-DESCCRIPTION : String [0..1]
-PROD-TYPE : String [0..1] = AAA
-PROD-SUBTYPE : String [0..1] = DATA
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-URI : String [0..1]
-NAME-PATTERN : String [0..1]
-DESCRIPTION : String [0..1]
-FORMAT : String [0..1]
-VERSION : String [0..1]
-SIZE : U32 [0..1]
-DATA : Binary [0..1]
-NUM-OF-MNEMONICS : U16
RESPONSE-STATUS
1=Acknowledgement,
2=Working/Keep Alive,
3=Successful Completion,
4=Failed Completion,
5=Invalid Request,
6=Final Message
PROD-TYPE
AAA=Archive and Assessment
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
-MNEMONIC
0..*
Mnemonic
-MNEMONIC.n.NAME : String
-MNEMONIC.n.STATUS : I16 = {1, 2, 3}
MNEMONIC.n.STATUS
1=Valid,
2=Valid, Nodata,
3=Invalid

Figure 8-26. Archive Mnemonic Value Response Message Diagram

The following table describes additional field names, values, and notes for the Archive Mnemonic Value Response Message.

Table 8-128. Archive Mnemonic Value Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the Archive Mnemonic Value Request Message that was processed.
			1	Acknowledgement	
			2	Working / Keep Alive	
			3	Successful Completion	
			4	Failed Completion	
			5	Invalid Request	
			6	Final Message	
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
TIME-COMPLETED	Time	O			Time application completed processing the request
SUPPORTED-STOP-TIME	Time	O			The service's calculated stop time. This is based on the DURATION or STOP-TIME specified in the request, but may be limited to what the service can

Field Name	Type	Presence	Value	Description
				support. For example, if the REQUEST specified a STOP-TIME a year in the future or a DURATION of 100 years, the service may calculate a reasonable SUPPORTED-STOP-TIME and return that value in this field. Reasonability is as determined by the service. This alerts the requestor that Archive Mnemonic Value Data Messages will not be sent after the SUPPORTED-STOP-TIME.
RETURN-VALUE	I32	O		Return value or status based on the RESPONSE-STATUS. Useful to provide function call status or error code in the case of failed completion
PROD-NAME	String	O		Name of the product
PROD-DESCRIPTION	String	O		Description of the product in text or xml
PROD-TYPE	String	O	**Value** / **Description** AAA / Archive and Assessment	Product type and subtype being requested. (See Table A-2, Product Categories).
PROD-SUBTYPE	String	O	DATA	
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R		Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6*, etc. in Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D		First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc. of the Product Message) - “n” starts at “1”. This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
URI	String	O		URI specifying the location where the (single) output file product is stored
NAME-PATTERN	String	O		Describes the name of the output file
DESCRIPTION	String	O		Description of the file in text or xml
FORMAT	String	O		For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
VERSION	String	O		Identifies the version of the file
SIZE	U32	O	Kilobytes	Actual size of the file
DATA	Binary	O		The file content
NUM-OF-MNEMONICS	U16	R		Total number of mnemonics returned
MNEMONIC.n.NAME	String	D		Name of the ‘nth’ Mnemonic - “n” starts at “1”. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.STATUS	I16	D	**Value** / **Description** 1 / Valid 2 / Valid, Nodata 3 / Invalid	Status of the ‘nth’ mnemonic: valid mnemonic or valid mnemonic with no data or invalid mnemonic. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.

The RESPONSE-STATUS field in the Archive Mnemonic Value Response Message indicates the success or failure of the component to process the Archive Mnemonic Value Request Message.

The values returned in the RESPONSE-STATUS field indicate if the request message was received, valid, invalid, able to be successfully and completely processed, or if the processing failed.

The ordering of the mnemonics in the Archive Mnemonic Value Response shall be the same as the receiving order specified in the Archive Mnemonic Value Request Message.

If any of the requested mnemonics in the Archive Mnemonic Value Request Message are invalid, the following are to occur:

- Set the RESPONSE-STATUS field to "5" or "Invalid Request".
- Set the status field of the invalid mnemonic(s) to "3" or "Invalid" (MNEMONIC.n.STATUS).
- Set the status field of all other valid mnemonics to "2" or "Valid, Nodata".
- The Archive Mnemonic Value product shall not be generated.

The MNEMONIC.n.STATUS field provides the status of each requested mnemonic. These dependent fields indicate:

- Valid – mnemonic was validated and data was located that met the criteria in the Archive Mnemonic Request Message.
- Valid, nodata – mnemonic was validated, but no data met the criteria in the corresponding request message.
- Invalid – mnemonic was not found in the database or list of mnemonics.

The following table indicates when the dependent fields in the response message are required.

Table 8-129. Relationship between RESPONSE-STATUS and Dependent Fields

Value	Description	Dependent Fields Required?
1	Acknowledgement	N
2	Working/Keep Alive	N
3	Successful Completion	Y
4	Failed Completion	N
5	Invalid Request	Y
6	Final Message	N

If a request is invalid (RESPONSE-STATUS field = "Invalid Request") it could be because one or more of the requested mnemonics was invalid. For this return status, the responder should provide all the requested mnemonics and the status of each.

The requestor has the option of specifying the URI where the responder should place the product file. If the URI is not specified, the responder will place the product file in its own designated location and return that location in the URI field of the response message. If the requestor specifies the URI, the responder will place the product file in that location, if possible, and return that same URI from the request message in the response message along with the corresponding RESPONSE-STATUS and RETURN-VALUE values.

If the responder is unable to place the product file in the specified URI location, the responder will place the file in an alternate URI location and return the URI in the response message along with the corresponding RESPONSE-STATUS and RETURN-VALUE values.

It is possible that the responder cannot access (write to) the URI specified by the requestor, and neither can the requestor access (read from) the alternate URI chosen by the responder in which case they will need to work out access and protection issues.

When the RESPONSE-STATUS is successful, the RETURN-VALUE can have the following status indicators:

1 – Product file was placed in requestor’s designated location

2 – Product file was placed in provider’s designated location

3 – Product file was generated in format other than that requested

The following table shows the relationship between the URI, RESPONSE-STATUS, and the RETURN-VALUE.

Table 8-130. Interpretation of the RESPONSE-STATUS and RETURN-VALUE Fields

User Specified the URI	RESPONSE-STATUS	RETURN-VALUE	URI	Action
N	Successful	2	Product was generated and placed in URI chosen by responder	Requestor should retrieve file at responder’s URI location
Y	Successful	1	Product was generated and placed in URI specified by requestor	Requestor should retrieve file at the specified URI
Y	Successful	2	Product was generated but placed in alternate URI chosen by responder	Requestor should retrieve file at responder’s URI location
n/a	Successful	3	n/a	Requestor should retrieve file at the specified URI.

The requestor also has the option of specifying the format and version of the product file.

If the FORMAT and VERSION are not specified, the responder will use the latest file format and version to build the product.

If the requestor specified the FORMAT and VERSION, the responder will generate the product in the desired format and version, if possible.

If the responder is unable to generate the product in the format and version requested, the responder will generate the product in the latest format and version along with the corresponding RESPONSE-STATUS and RETURN-VALUE.

8.10.3 Archive Mnemonic Value Data Message

The Archive Mnemonic Value Data Message provides the telemetry or configuration mnemonic data that was requested in the Archive Mnemonic Value Request Message.

The messages are generated in response to receiving an Archive Mnemonic Value Request Message to publish the requested Mnemonic values in a stream of messages. In this case, the Archive Mnemonic Value Request Message specified the delivery mechanism to be a stream of messages – similar to the real-time Mnemonic Value Data Messages.

The following figure shows a UML sequence diagram for the different Archive Mnemonic Value Messages and how the message protocol between the data requestor and data provider would occur.

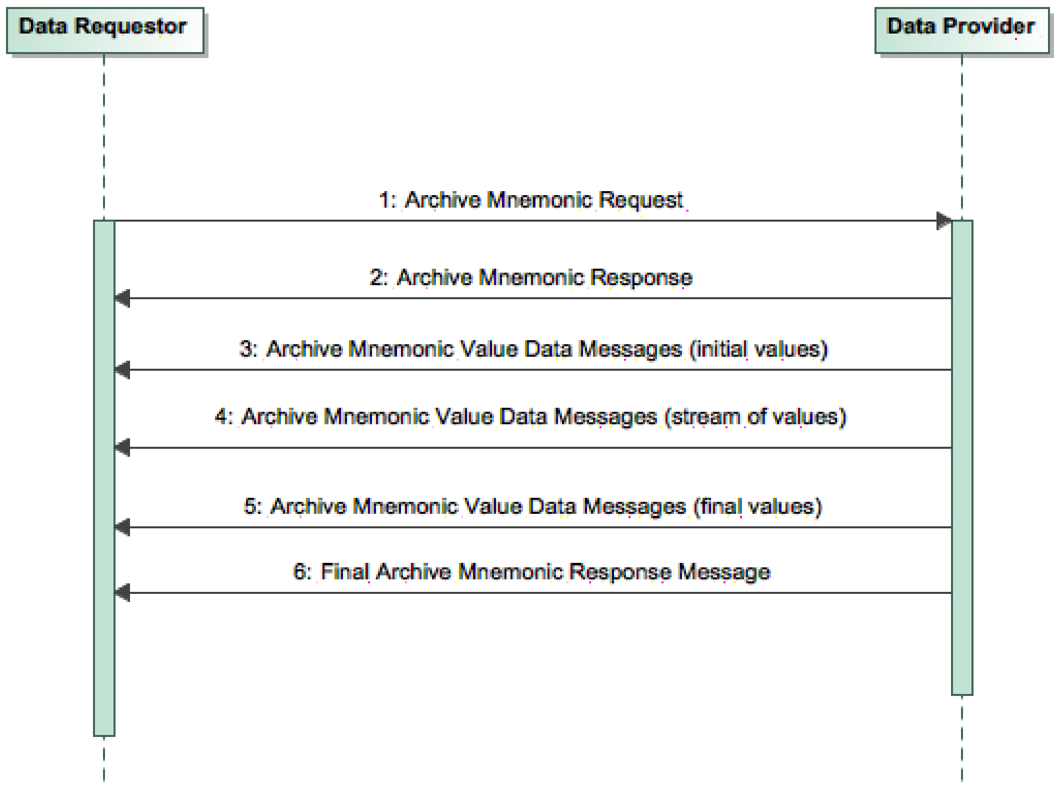

Data Requestor
Data Provider
1: Archive Mnemonic Request
2: Archive Mnemonic Response
3: Archive Mnemonic Value Data Messages (initial values)
4: Archive Mnemonic Value Data Messages (stream of values)
5: Archive Mnemonic Value Data Messages (final values)
6: Final Archive Mnemonic Response Message

Figure 8-27. Archive Mnemonic Value Message Sequence Diagram

The previous diagram shows an initial exchange of the Archive Mnemonic Value Request Message and Archive Mnemonic Value Response Message. This is followed by a stream of Archive Mnemonic Value Data Messages. A final Archive Mnemonic Value Response Message is optional.

The stream of Archive Mnemonic Value Data Messages follows the successful exchange of the Archive Mnemonic Value Request and Response Messages. The Archive Mnemonic Value Data Messages shall not be published if the Archive Mnemonic Value Request Message contained any invalid mnemonics. The Archive Mnemonic Value Data Messages will continue to be published at the specified rate until all requested data has been published. The ordering of the mnemonics in the Archive Mnemonic Value Data Message shall be the same as the order specified in the Archive Mnemonic Value Request Message.

The Archive Mnemonic Value Data Message will contain one to many mnemonics and one to many data samples per mnemonic.

If there is no data available for a mnemonic, the MNEMONIC.n.STATUS field will indicate a “Valid, Nodata” condition exists and the subsequent associated data value fields will not be provided for this mnemonic.

Table 8-131. Archive Mnemonic Value Data Message Summary

Sender	A C2MS compliant application that has access to a telemetry archive such as a T&C component
Senders Intended Usage	Publish
Receiver	GUI Subsystem, Command Subsystem, Schedule Execution process, Expert Subsystem, Assessment and Analysis
Receivers Intended Usage	Subscribe
What	Spacecraft health and safety data, configuration data values, any mnemonic data value
When	After sending successful Archive Mnemonic Value Response Message. Then, at specified interval or requested rate.

Examples

1. Telemetry data to be displayed on a GUI page
2. Telemetry data for an analysis plot

8.10.3.1 Archive Mnemonic Value Data Message Subjects

Table 8-132. Archive Mnemonic Value Data Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	AMVAL	[COMPONENT]	[request id]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	AMVAL	TLM3	[GUID]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	AMVAL	TLM2	[GUID]				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	AMVAL	TLM3	[GUID]				

Note that "[GUID]" in the table above is a Subject Token String that conforms to the GUID type specified in this document, such as "b891bdac-964a-4f3e-957c-1a29ec4c7d50" or similar.

Table 8-133. Properties of the *Miscellaneous Elements* for the Archive Mnemonic Value Data Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Optional	ID associated with original request	REQUEST-ID

Examples

After the successful exchange of the Archive Mnemonic Value Request and Response Messages, the Archive Mnemonic Value Data Messages are published.

Two different Data Provider/Publisher/Servers send out the Archive Mnemonic Value Data Messages with the following subjects:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.AMVAL.TLM3
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.AMVAL.TLM2
```

The Requestor/Subscriber/Client subscribes to receive the intended Archive Mnemonic Value Data Messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.AMVAL.*
```

8.10.3.2 Archive Mnemonic Value Data Message Header

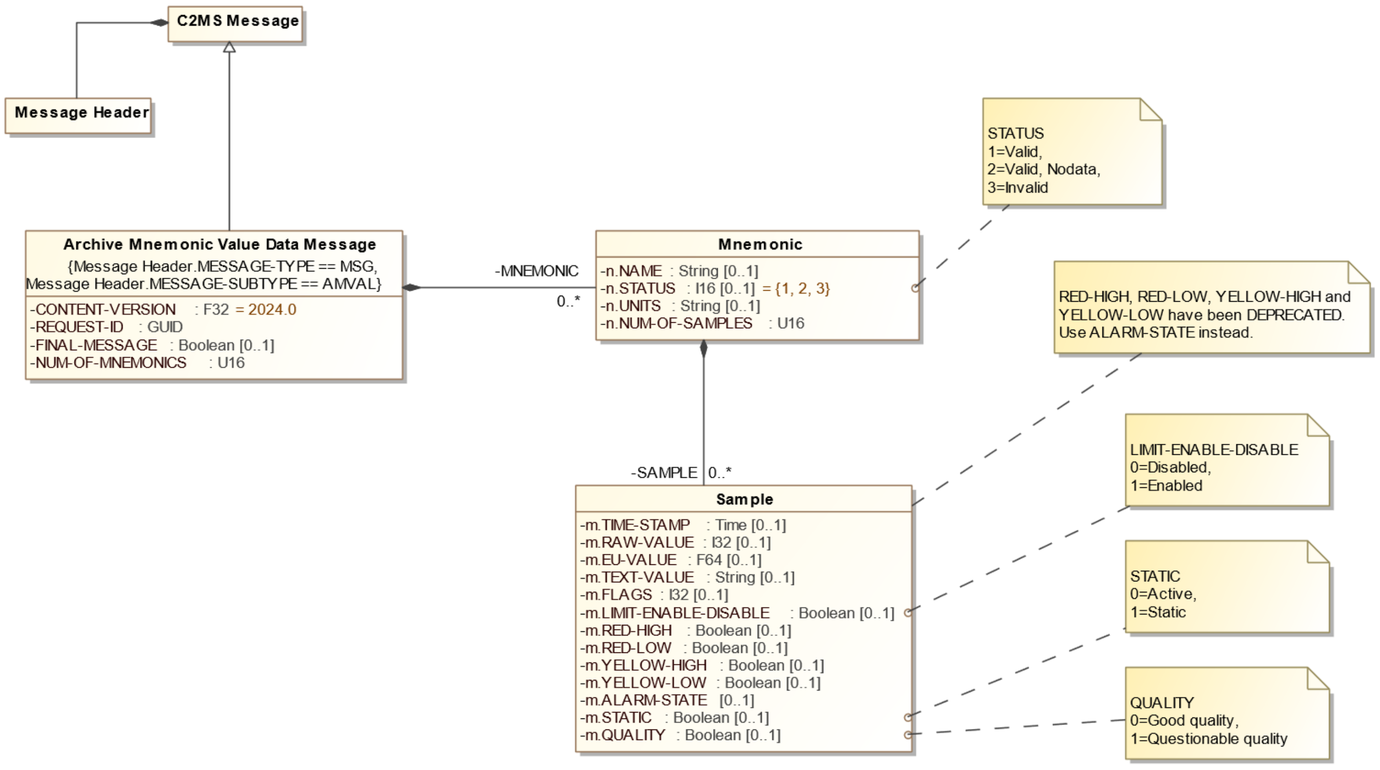
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Archive Mnemonic Value Data Message header.

Table 8-134. Archive Mnemonic Value Data Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	AMVAL

8.10.3.3 Archive Mnemonic Value Data Message Contents

The following figure shows a UML object diagram of the Archive Mnemonic Value Data Message with its required and optional fields.



C2MS Message
Message Header
Archive Mnemonic Value Data Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == AMVAL}
-CONTENT-VERSION : F32 = 2024.0
-REQUEST-ID : GUID
-FINAL-MESSAGE : Boolean [0..1]
-NUM-OF-MNEMONICS : U16
-MNEMONIC
0..*
Mnemonic
-n.NAME : String [0..1]
-n.STATUS : I16 [0..1] = {1, 2, 3}
-n.UNITS : String [0..1]
-n.NUM-OF-SAMPLES : U16
STATUS
1=Valid,
2=Valid, Nodata,
3=Invalid
RED-HIGH, RED-LOW, YELLOW-HIGH and YELLOW-LOW have been DEPRECATED. Use ALARM-STATE instead.
-SAMPLE
0..*
Sample
-m.TIME-STAMP : Time [0..1]
-m.RAW-VALUE : I32 [0..1]
-m.EU-VALUE : F64 [0..1]
-m.TEXT-VALUE : String [0..1]
-m.FLAGS : I32 [0..1]
-m.LIMIT-ENABLE-DISABLE : Boolean [0..1]
-m.RED-HIGH : Boolean [0..1]
-m.RED-LOW : Boolean [0..1]
-m.YELLOW-HIGH : Boolean [0..1]
-m.YELLOW-LOW : Boolean [0..1]
-m.ALARM-STATE [0..1]
-m.STATIC : Boolean [0..1]
-m.QUALITY : Boolean [0..1]
LIMIT-ENABLE-DISABLE
0=Disabled,
1=Enabled
STATIC
0=Active,
1=Static
QUALITY
0=Good quality,
1=Questionable quality

Figure 8-28. Archive Mnemonic Value Data Message Diagram

The following table describes additional field names, values, and notes for the Archive Mnemonic Value Data Message.

Table 8-135. Archive Mnemonic Value Data Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
FINAL-MESSAGE	Boolean	O			When true, indicates the last message in the stream.
NUM-OF-MNEMONICS	U16	R			Total number of mnemonics in this message
MNEMONIC.n.NAME	String	O			Name of the 'nth' mnemonic - "n" starts at "1".
MNEMONIC.n.STATUS	I16	O	**Value**	**Description**	Status of the 'nth' mnemonic: valid mnemonic, or valid mnemonic with nodata, or invalid mnemonic
			1	Valid	
			2	Valid, Nodata	
			3	Invalid	
MNEMONIC.n.UNITS	String	O			Units associated with the raw value converted to engineering units for the 'nth' mnemonic

Field Name	Type	Presence	Value		Description
MNEMONIC.n.NUM-OF-SAMPLES	U16	D			Number of data samples for the ‘nth’ mnemonic. This field is required for each mnemonic when NUM-OF-MNEMONICS > 0.
MNEMONIC.n.SAMPLE .m.TIME-STAMP	Time	O			Time stamp for the ‘nth’ data sample of the ‘nth’ mnemonic - both “n” and "m" start at “1”.
MNEMONIC.n.SAMPLE .m.RAW-VALUE	I32	O			Raw value for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE .m.EU-VALUE	F64	O			Raw value converted to Engineering Units if engineering units conversion is present for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE .m.TEXT-VALUE	String	O			Raw value converted to a text string if text conversion is present for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE .m.FLAGS	I32	O			Flags native to the T&C component for the ‘nth’ data sample of the ‘nth’ mnemonic
MNEMONIC.n.SAMPLE .m.LIMIT-ENABLE-DISABLE	Boolean	O	**Value**	**Description**	Indicates the limit checking state for the ‘nth’ data sample of the ‘nth’ mnemonic
			0	Disabled	
			1	Enabled	
DEPRECATED MNEMONIC.n.SAMPLE .m.RED-HIGH	Boolean	O	**Value**	**Description**	Indicates the Red High limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE .m.RED-LOW	Boolean	O	**Value**	**Description**	Indicates the Red Low limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE .m.YELLOW-HIGH	Boolean	O	**Value**	**Description**	Indicates the Yellow High limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
DEPRECATED MNEMONIC.n.SAMPLE .m.YELLOW-LOW	Boolean	O	**Value**	**Description**	Indicates the Yellow Low limit status for the ‘nth’ data sample of the ‘nth’ mnemonic. This field has been deprecated. Use MNEMONIC.n.SAMPLE.m.ALARM-STATE, instead.
			0	In-limits	
			1	Out-of-limits	
MNEMONIC.n.SAMPLE .m.ALARM-STATE	I16	O	**Value**	**Description**	Indicates the limit state of the ‘mth’ data sample of the ‘nth’ mnemonic
			0	Unavailable	
			1	Normal	
			2	Warning	
			3	Distress	
			4	Critical	
MNEMONIC.n.SAMPLE .m.STATIC	Boolean	O	**Value**	**Description**	Indicates the static (stale) condition for the ‘nth’ data sample of the ‘nth’ mnemonic
			0	Active	
			1	Static	
MNEMONIC.n.SAMPLE .m.QUALITY	Boolean	O	**Value**	**Description**	Indicates the Quality for the ‘nth’ data sample of the ‘nth’ mnemonic
			0	Good quality	
			1	Questionable quality	

8.11 Satellite Command Messages

The Command Request, Command Response, and Command Echo Messages are used to send satellite commands and return status among components within the space-ground system.

Command Messages, which originate within a mission's operation center, transport satellite or spacecraft commands over the ground network for transmission to the satellite.

Due to bandwidth narrowing on the uplink that may be considerably less than what is available on the ground, the Command Messages may not be uplinked to the satellite in the same exact format. It is possible that the message as transported through the ground network may be stripped, compacted, or otherwise reduced in volume for transmission to the satellite.

Similarly, but in reverse, downlinked data may be expanded or converted from binary or a compacted format into more verbose or descriptive standard message formats. Thus, a bridge task may serve as a middleman / interpreter / converter between messages and data transferred between the ground and satellites.

8.11.1 Command Request Message

The Command Request Message is the mechanism to send a satellite command from one component to another within the space-ground system. A satellite command may pass through a number of evolutionary steps that include scheduling, building, creating, validating, transporting, execution, and verification.

A satellite command can be input by a flight operations team member through a GUI or command line, or as part of the internal logic of a component. They may be grouped together with processing or execution logic in a file, procedure, or command schedule.

The Command Request Message is used to package a command in whatever circumstance it is found and transport it to the next component for processing.

Thus, a Command Request Messages may originate from a number of sources such as a GUI / work position, a command line, a schedule, procedure, or any number of places within a space-ground system. It may be used by and pass through a number of components before arriving at its satellite destination.

Table 8-136. Command Request Message Summary

Sender	A C2MS compliant application responsible for generating or processing a satellite command
Senders Intended Usage	Request processing of a satellite command
Receiver	Application providing a satellite command service such as building, executing, verifying, or transporting
Receivers Intended Usage	Subscribe
What	Action request initiated by user, software, procedure, command schedule, etc.
When	Normally, at scheduled time, or as circumstances warrant

Examples

1. An operator input issuing a satellite command.

2. A command schedule execution component.
3. A request issued from a procedure.

8.11.1.1 Command Request Message Subjects

Table 8-137. Command Request Message Subject Naming

	Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	REQ	CMD	[DESTINATION-COMPONENT]					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	CMD	COMMOUT					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	CMD	ANTENNA5					
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	REQ	CMD	COMMOUT					

Table 8-138. Properties of the *Miscellaneous Elements* for the Command Request Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2 …	Not used		

Examples

Two components, CMDEXEC and CMDOUT, interact with the Command Request Message.

CMDEXEC subject to send the Command Request to CMDOUT:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.CMD.CMDOUT
```

CMDOUT subject to receive its own Command Request Messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.CMD.CMDOUT
```

CMDOUT subject to receive any CMDOUT Request Message:

```
C2MS.*.*.*.*.*.REQ.*.CMDOUT
```

CMDOUT subject to receive any kind (REQ or RESP) of Command messages:

```
C2MS.*.*.*.*.*.*.CMD.CMDOUT
```

8.11.1.2 Command Request Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Command Request Message header.

Table 8-139. Command Request Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	CMD

8.11.1.3 Command Request Message Contents

The following figure shows a UML object diagram of the Command Request Message with its required, optional, and dependent fields.

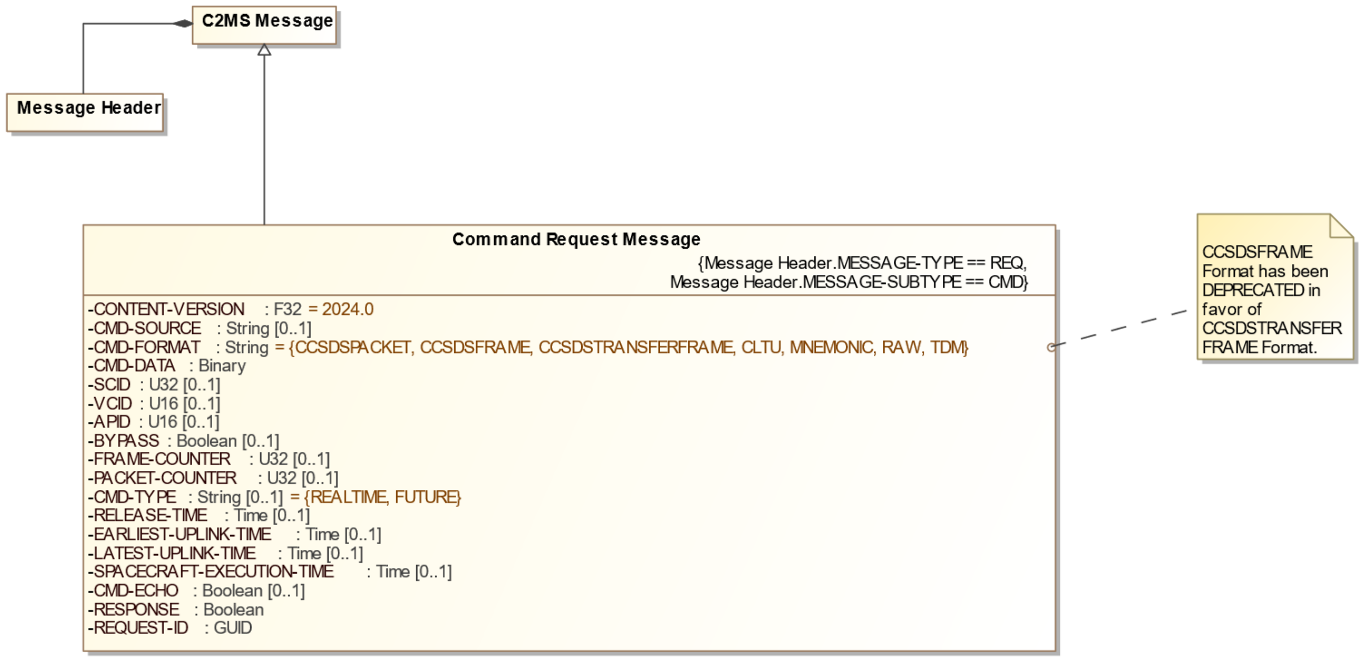

C2MS Message
Message Header
Command Request Message
{Message Header.MESSAGE-TYPE == REQ, Message Header.MESSAGE-SUBTYPE == CMD}
-CONTENT-VERSION : F32 = 2024.0
-CMD-SOURCE : String [0..1]
-CMD-FORMAT : String = {CCSDSPACKET, CCSDSFRAME, CCSDSTRANSFERFRAME, CLTU, MNEMONIC, RAW, TDM}
-CMD-DATA : Binary
-SCID : U32 [0..1]
-VCID : U16 [0..1]
-APID : U16 [0..1]
-BYPASS : Boolean [0..1]
-FRAME-COUNTER : U32 [0..1]
-PACKET-COUNTER : U32 [0..1]
-CMD-TYPE : String [0..1] = {REALTIME, FUTURE}
-RELEASE-TIME : Time [0..1]
-EARLIEST-UPLINK-TIME : Time [0..1]
-LATEST-UPLINK-TIME : Time [0..1]
-SPACECRAFT-EXECUTION-TIME : Time [0..1]
-CMD-ECHO : Boolean [0..1]
-RESPONSE : Boolean
-REQUEST-ID : GUID
CCSDSFRAME Format has been DEPRECATED in favor of CCSDSTRANSFER FRAME Format.

Figure 8-29. Command Request Message Contents Diagram

The following table describes additional field names, values, and notes for the Command Request Message.

Table 8-140. Command Request Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
CMD-SOURCE	String	O		Which user/work position/proc/schedule the message originated from
CMD-FORMAT	String	R	[CCSDSPACKET, (**DEPRECATED**) CCSDSFRAME, CCSDSTRANSFERFRAME, CLTU, MNEMONIC, RAW, TDM]	Type of command. CCSDSFRAME has been deprecated. Use CCSDSTRANSFERFRAME format instead.
CMD-DATA	Binary	R		Command data
SCID	U32	O		CCSDS Spacecraft Identifier. SCID applies to CCSDSPACKET, CCSDSFRAME, CCSDSTRANSFERFRAME, and CLTU CMD-FORMAT.
VCID	U16	O		CCSDS Virtual Channel ID
APID	U16	O		CCSDS Application Process Identifier . APID specifically applies to the CCSDSPACKET CMD-FORMAT.
BYPASS	Boolean	O		CCSDS COP-1 flag for “Bypass of Acceptance Check”, i.e. without verification

Field Name	Type	Presence	Value	Description
FRAME-COUNTER	U32	O		Reset to next expected command counter
PACKET-COUNTER	U32	O		
CMD-TYPE	String	O	[REALTIME, FUTURE]	If REALTIME, execute upon receipt. If FUTURE, execute at SPACECRAFT-EXECTION-TIME.
RELEASE-TIME	Time	O		Time the command should begin being released from the front-end processor to the remote tracking station.
EARLIEST-UPLINK-TIME	Time	O		Absolute or relative time can apply.
LATEST-UPLINK-TIME	Time	O		Absolute or relative time can apply.
SPACECRAFT-EXECUTION-TIME	Time	D		Required if CMD-TYPE = 'FUTURE". Absolute or relative time can apply.
CMD-ECHO	Boolean	O		Indicate if an ECHO should be sent at any configured point(s) along the ground chain as the command is transmitted.
RESPONSE	Boolean	R		Indicates if a response is required.
REQUEST-ID	GUID	R		ID to identify the request message

8.11.2 Command Response Message

A Command Response Message is sent by an application in response to a Command Request Message. The Command Response Message provides acknowledgement of the Command Request Message and a status of the action completed. Since the building, transmission, execution, and verification of a satellite command may involve a number of components, the status that is returned may be for any one of these steps in the process.

Table 8-141. Command Response Message Summary

Sender	A C2MS application that received the Command Request Message
Senders Intended Usage	Reply to the Command Request Message
Receiver	Application that issued the Command Request Message, or an application collecting Command Response Messages for audit trail purposes
Receivers Intended Usage	Subscribe
What	Provide success, failure, or interim status of the progress of the satellite command that was issued/requested
When	Upon receipt of the Command Request Message, or completion of this step of processing

Examples

1. Acknowledge receipt of the satellite command.
2. Return status of this step in the sequence of sending a satellite command.

8.11.2.1 Command Response Message Subjects

Table 8-142. Command Response Message Subject Naming

	Subject Standard	**Domain Elements**		**Mission Elements**			**Message Elements**		***Miscellaneous Elements***					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	CMD	[DESTINATION-COMPONENT]	[Status]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	CMD	CMDOUT	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	CMD	CMDEXEC	4				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	RESP	CMD	CMDEXEC	*				

Table 8-143. Properties of the *Miscellaneous Elements* for the Command Response Message

Miscellaneous Element	Required / Optional	Description	Field Origination in Msg, if applicable
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components, CMDEXEC and CMDOUT, interact with the Command Response message.

CMDOUT subject to send the Command Response to CMDEXEC:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.CMD.CMDEXEC.3
```

CMDEXEC subscribes to receive its own Command Response Messages:

```
C2MS.*.*.*.*.*.RESP.CMD.CMDEXEC.>
```

8.11.2.2 Command Response Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Command Response Message header.

Table 8-144. Command Response Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	CMD

8.11.2.3 Command Response Message Contents

The following figure shows a UML object diagram of the Command Response Message with its required and optional fields.

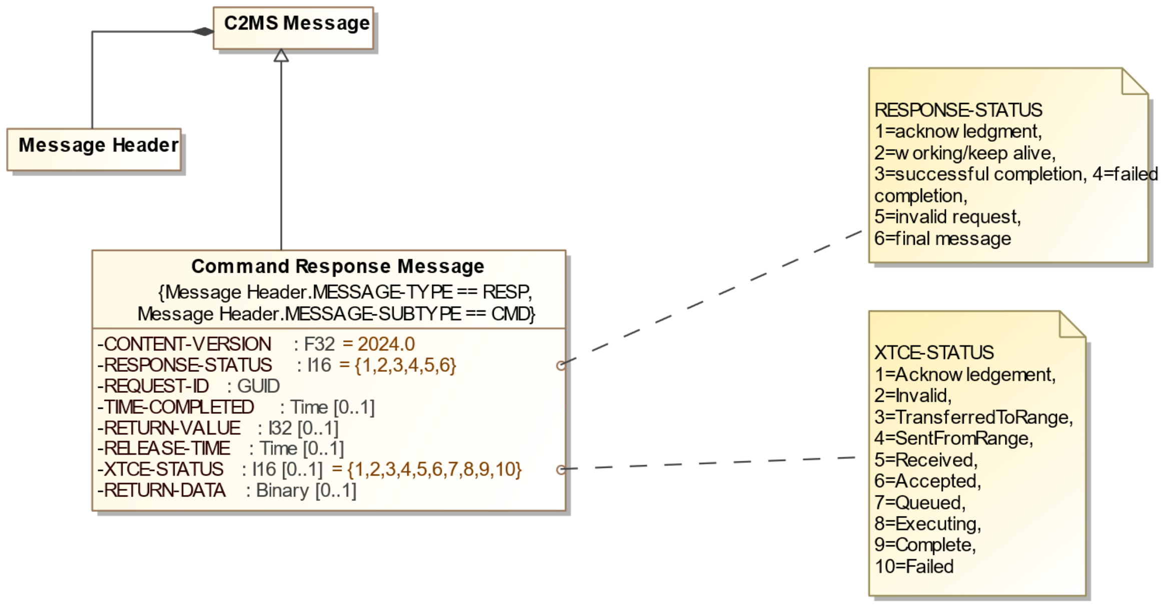

C2MS Message
Message Header
Command Response Message
{Message Header.MESSAGE-TYPE == RESP,
Message Header.MESSAGE-SUBTYPE == CMD}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1,2,3,4,5,6}
-REQUEST-ID : GUID
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-RELEASE-TIME : Time [0..1]
-XTCE-STATUS : I16 [0..1] = {1,2,3,4,5,6,7,8,9,10}
-RETURN-DATA : Binary [0..1]
RESPONSE-STATUS
1=acknow ledgment,
2=w orking/keep alive,
3=successful completion, 4=failed completion,
5=invalid request,
6=final message
XTCE-STATUS
1=Acknow ledgement,
2=Invalid,
3=TransferredToRange,
4=SentFromRange,
5=Received,
6=Accepted,
7=Queued,
8=Executing,
9=Complete,
10=Failed

Figure 8-30. Command Response Message Diagram

The following table describes additional field names, values, and notes for the Command Response Message.

Table 8-145. Command Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the command being processed
			1	Acknowledgement	
			2	Working/Keep alive	
			3	Successful completion	
			4	Failed completion	
			5	Invalid request	
			6	Final Message	

Field Name	Type	Presence	Value		Description
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
TIME-COMPLETED	Time	O			Time application completed processing the Command
RETURN-VALUE	I32	O			Return value or status based on the RESPONSE-STATUS. Useful to provide function call status or error code in the case of failed completion
RELEASE-TIME	Time	O			Time command finished being released from the front-end processor to the remote tracking station.
XTCE-STATUS	I16	O	**Value**	**Description**	Status codes from the OMG XML Telemetric and Command data Exchange data specification.
			1	Acknowledgement	
			2	Invalid	
			3	TransferredToRange	
			4	SentFromRange	
			5	Received	
			6	Accepted	
			7	Queued	
			8	Executing	
			9	Complete	
			10	Failed	
RETURN-DATA	Binary	O			Additional data that may be desired along with the completion status

8.11.3 Command Echo Message

A Command Echo Message is sent by an application that receives echo data, such as ground station equipment that collects data from the antenna (either looped back at the ground site or the spacecraft itself). This message is nominally sent after a Command Request is processed by the final destination (e.g., antenna or spacecraft). The Command Echo Message can also be generated without a prior Command Request Message being sent; this is known as an "unsolicited echo", and can be generated by ground station equipment as a result of the antenna receiving noise or interference. The purpose of this message is to carry the actual command data that was received by the final destination, and supply it to the spacecraft operations center for comparison to the original command to ensure integrity of the command bits received at the destination.

Table 8-146. Command Echo Message Summary

Sender	A C2MS application that (may have) received the Command Request Message
Senders Intended Usage	Reply to the Command Request Message or send unsolicited
Receiver	Application that issued the Command Request Message, or an application collecting Command Echo Messages for audit trail purposes
Receivers Intended Usage	Subscribe
What	Provide success, failure, or interim status of the progress of the satellite command that was issued/requested
When	Upon receipt of the Command Request Message, or completion of this step of processing or unsolicited (see description)

Examples

1. Acknowledge receipt of the satellite command.
2. Return status of this step in the sequence of sending a satellite command.

8.11.3.1 Command Echo Message Subjects

Table 8-147. Command Echo Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	CMDECHO	[COMPONENT]	[Status]	[request id]			
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	CMDECHO	CMDOUT	1	[GUID]			
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	CMDECHO	CMDEXEC	4	[GUID]			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	MSSN	*	SAT1	MSG	CMDECHO	CMDEXEC	*	[GUID]			

Note that "[GUID]" in the table above is a Subject Token String that conforms to the GUID type specified in this document, such as "b891bdac-964a-4f3e-957c-1a29ec4c7d50" or similar.

Table 8-148. Properties of the *Miscellaneous Elements* for the Command Echo Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field Origination in Msg, if applicable**
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Status type supplied by Responder	CMD-ECHO-RESULT
ME3	Optional	ID associated with original request	REQUEST-ID

Examples

Two components, CMDEXEC and CMDOUT, interact with the Command Echo message.

CMDOUT subject to send the Command Echo to CMDEXEC:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.CMDECHO.CMDEXEC.GOOD
```

CMDEXEC subscribes to receive its own Command Echo Messages:

```
C2MS.*.*.*.*.*.MSG.CMDECHO.CMDEXEC.>
```

8.11.3.2 Command Echo Message Header

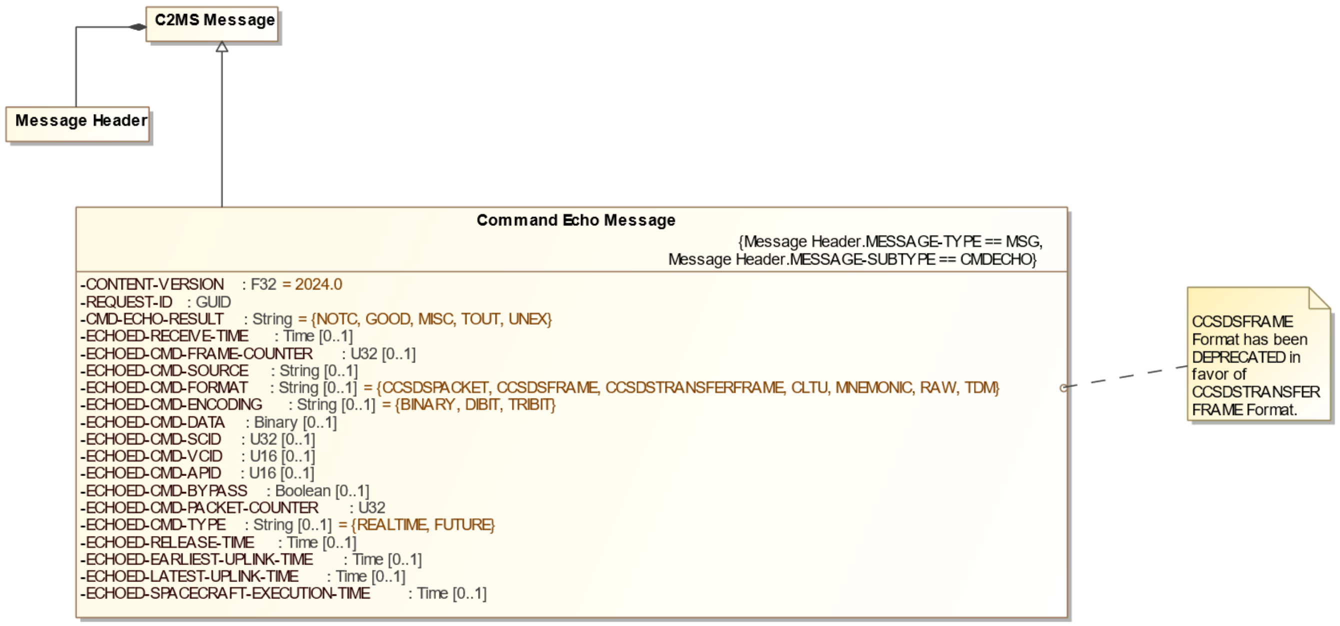
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Command Echo Message header.

Table 8-149. Command Echo Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	CMDECHO

8.11.3.3 Command Echo Message Contents

The following figure shows a UML object diagram of the Command Echo Message with its required and optional fields.



C2MS Message
Message Header
Command Echo Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == CMDECHO}
-CONTENT-VERSION : F32 = 2024.0
-REQUEST-ID : GUID
-CMD-ECHO-RESULT : String = {NOTC, GOOD, MISC, TOUT, UNEX}
-ECHOED-RECEIVE-TIME : Time [0..1]
-ECHOED-CMD-FRAME-COUNTER : U32 [0..1]
-ECHOED-CMD-SOURCE : String [0..1]
-ECHOED-CMD-FORMAT : String [0..1] = {CCSDSPACKET, CCSDSFRAME, CCSDSTRANSFERFRAME, CLTU, MNEMONIC, RAW, TDM}
-ECHOED-CMD-ENCODING : String [0..1] = {BINARY, DIBIT, TRIBIT}
-ECHOED-CMD-DATA : Binary [0..1]
-ECHOED-CMD-SCID : U32 [0..1]
-ECHOED-CMD-VCID : U16 [0..1]
-ECHOED-CMD-APID : U16 [0..1]
-ECHOED-CMD-BYPASS : Boolean [0..1]
-ECHOED-CMD-PACKET-COUNTER : U32
-ECHOED-CMD-TYPE : String [0..1] = {REALTIME, FUTURE}
-ECHOED-RELEASE-TIME : Time [0..1]
-ECHOED-EARLIEST-UPLINK-TIME : Time [0..1]
-ECHOED-LATEST-UPLINK-TIME : Time [0..1]
-ECHOED-SPACECRAFT-EXECUTION-TIME : Time [0..1]
CCSDSFRAME Format has been DEPRECATED in favor of CCSDSTRANSFER FRAME Format.

Figure 8-31. Command Echo Message Diagram

The following table describes additional field names, values, and notes for the Command Echo Message.

Table 8-150. Command Echo Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
CMD-ECHO-RESULT	String	R	**Value**	**Description**	The command echo result
			NOTC	Not Compared	
			GOOD	Good Compare	
			MISC	Miscompare	

Field Name	Type	Presence	Value	Description
			TOUT — Timed out waiting for echo	
			UNEX — Unexpected echo received	
ECHOED-RECEIVE-TIME	Time	O		Time the echo was received or when the timeout occurred
ECHOED-CMD-FRAME-COUNTER	U32	O		The echoed command counter; i.e., the command id
ECHOED-CMD-SOURCE	String	O		Which user/workposition/proc/schedule the message originated from
ECHOED-CMD-FORMAT	String	O	[CCSDSPACKET, (**DEPRECATED**) CCSDSFRAME, CCSDSTRANSFERFRAME, CLTU, MNEMONIC, RAW, TDM]	Type of command. CCSDSFRAME has been deprecated. Use CCSDSTRANSFERFRAME format instead.
ECHOED-CMD-ENCODING	String	O	[BINARY, DIBIT, TRIBIT]	Type of command encoding; required if ECHOED-CMD-DATA is present
ECHOED-CMD-DATA	Binary	O		Received command echo data used to compare with uplinked CMD-DATA
ECHOED-CMD-SCID	U32	O		CCSDS Spacecraft Identifier. SCID applies to CSDSPACKET, CCSDSFRAME, CCSDSTRANSFERFRAME, and CLTU ECHOED-CMD-FORMAT.
ECHOED-CMD-VCID	U16	O		CCSDS Virtual Channel ID
ECHOED-CMD-APID	U16	O		CCSDS Application Process Identifier. APID specifically applies to the CCSDSPACKET ECHOED-CMD-FORMAT.
ECHOED-CMD-BYPASS	Boolean	O		CCSDS COP-1 flag for “Bypass of Acceptance Check”, i.e., without verification
ECHOED-CMD-PACKET-COUNTER	U32	O		
ECHOED-CMD-TYPE	String	O	[REALTIME, FUTURE]	If REALTIME, execute upon receipt; if FUTURE, execute at ECHOED-SPACECRAFT-EXECUTION-TIME
ECHOED-RELEASE-TIME	Time	O		Time the command should begin being released from the front end processor to the remote tracking station
ECHOED-EARLIEST-UPLINK-TIME	Time	O		Absolute or relative time can apply
ECHOED-LATEST-UPLINK-TIME	Time	O		Absolute or relative time can apply
ECHOED-SPACECRAFT-EXECUTION-TIME	Time	O		Required if ECHOED-CMD-TYPE=”FUTURE”; absolute or relative time can apply

8.12 Product Messages

8.12.1 Product Request Message

C2MS has defined the following messages to facilitate the needs of product producers and consumers.

Product Request Message – used to request a product.

Product Response Message – used to return status of the request, and optionally to provide the product.

Product Message – used to:

1. Announce the **availability** of a generated product
2. Announce a product is **accessible** by providing the location with a Uniform Resource Identifier (URI), *or*
3. Provide the Product in the message or as an **attachment**.

Table 8-151. Uses of the Product Message

Usage	User Required Action
Available	Must request the product
Accessible	Use the URI to get the product
Attachment	Extract the product from the message

The Product Message is published either:

1. After the exchange of the Product Request and Product Response Messages, *or*

2. Unsolicited

The Product Response Message and the Product Message are used to distribute products. These messages are used for a single product that may contain a multiple number of files. Generally, the contents of the different messages are as follows:

Product Request Message

- Requests the distribution of product(s) – might require the producer to generate
- Specifies attributes to describe the requested product(s)
- Provides information to direct the means of distribution and/or target the distribution location
- Optionally, includes precursor products (files) that are used to generate the requested product

Product Response Message

- Return Status of the Product Request
- Optionally, contains the actual product or product location information and the product attributes

Product Message

- Contains the actual product or product location information
- Contains product attributes

The C2MS Product Request Message effectively requests the distribution of a product. It may incidentally require the generation of that product by the producer if it does not already exist.

The C2MS Product Response Message and the Product Message incorporate a framework to identify the number of files per product. The messages also allow for determining the location of the distribution. The requestor could specify the location or allow the producer to specify the location. Of course, these locations are dependent on the granted access and authorization of components to these designated locations.

8.12.1.1 Product Request Message Subjects

Table 8-152. Product Request Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	REQ	PROD	[DESTINATION-COMPONENT]					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	PROD	USER10					
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	PROD	APP5					
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	REQ	PROD	PLOTGEN					

Table 8-153. Properties of the Miscellaneous Elements for the Product Request Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Responder	DESTINATION-COMPONENT from header
ME2	Not used		

Examples

Two components, USER10 and PLOTGEN interact with the Product Request Message.

USER10 (Data Requestor/Subscriber/Client) sends a request to PLOTGEN the product Provider/Publisher/Server.

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.PROD.PLOTGEN
```

PLOTGEN subscribe subject to receive the Product Request Message.

```
C2MS.*.*.*.*.*.REQ.PROD.PLOTGEN
```

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.*.PLOTGEN` (PLOTGEN will receive any REQ message)

8.12.1.2 Product Request Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Product Request Message header.

Table 8-154. Product Request Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	PROD

8.12.1.3 Product Request Message Contents

The following figure shows a UML object diagram of the Product Request Message with its required and optional fields.

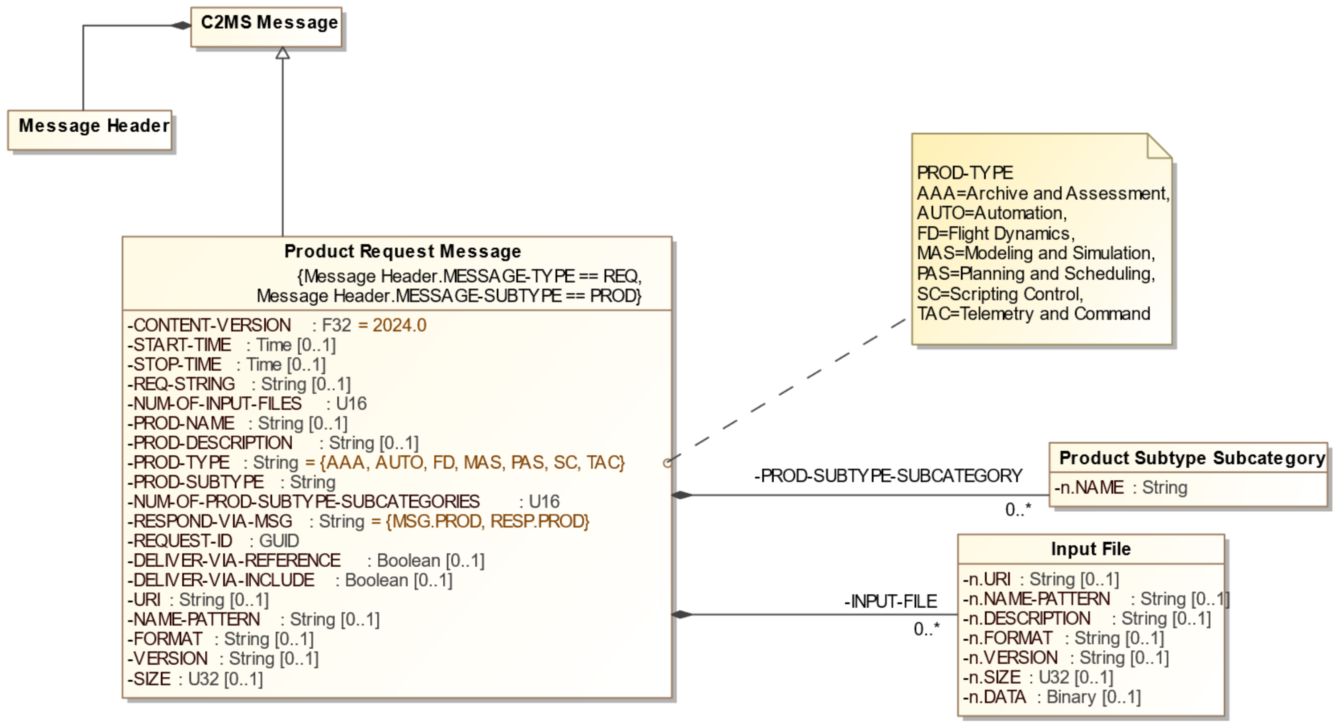

C2MS Message
Message Header
Product Request Message
{Message Header.MESSAGE-TYPE == REQ, Message Header.MESSAGE-SUBTYPE == PROD}
-CONTENT-VERSION : F32 = 2024.0
-START-TIME : Time [0..1]
-STOP-TIME : Time [0..1]
-REQ-STRING : String [0..1]
-NUM-OF-INPUT-FILES : U16
-PROD-NAME : String [0..1]
-PROD-DESCRIPTION : String [0..1]
-PROD-TYPE : String = {AAA, AUTO, FD, MAS, PAS, SC, TAC}
-PROD-SUBTYPE : String
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-RESPOND-VIA-MSG : String = {MSG.PROD, RESP.PROD}
-REQUEST-ID : GUID
-DELIVER-VIA-REFERENCE : Boolean [0..1]
-DELIVER-VIA-INCLUDE : Boolean [0..1]
-URI : String [0..1]
-NAME-PATTERN : String [0..1]
-FORMAT : String [0..1]
-VERSION : String [0..1]
-SIZE : U32 [0..1]
PROD-TYPE
AAA=Archive and Assessment,
AUTO=Automation,
FD=Flight Dynamics,
MAS=Modeling and Simulation,
PAS=Planning and Scheduling,
SC=Scripting Control,
TAC=Telemetry and Command
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
-INPUT-FILE
0..*
Input File
-n.URI : String [0..1]
-n.NAME-PATTERN : String [0..1]
-n.DESCRIPTION : String [0..1]
-n.FORMAT : String [0..1]
-n.VERSION : String [0..1]
-n.SIZE : U32 [0..1]
-n.DATA : Binary [0..1]

Figure 8-32. Product Request Message Diagram

The following table describes additional field names, values, and notes for the Product Request Message.

Table 8-155. Product Request Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2024	Version number for this message content description
START-TIME	Time	O		Requested start time for the scope of the product to cover.
STOP-TIME	Time	O		Requested stop time for the scope of the product to cover.
REQ-STRING	String	O		For application use as defined by the product provider. The string will define a, directive string, or some other keyword syntax made known by the provider.
NUM-OF-INPUT-FILES	U16	R	0+	Indicates the number of files included in this request message.
INPUT-FILE.n.URI	String	O		URI specifying the location where the file of the product is stored - “n” starts at “1”.
INPUT-FILE.n.NAME-PATTERN	String	O		Describes the file name
INPUT-FILE.n.DESCRIPTION	String	O		Description of the file in text or xml

Field Name	Type	Presence	Value	Description
INPUT-FILE.n.FORMAT	String	O		For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
INPUT-FILE.n.VERSION	String	O		Identifies the version ID of the file
INPUT-FILE.n.SIZE	U32	O	KB	Size of the included file
INPUT-FILE.n.DATA	Binary	O		The file content
PROD-NAME	String	O		Name of the product
PROD-DESCRIPTION	String	O		Description of the product in text or xml
PROD-TYPE	String	R	**Value** – **Description** AAA – Archive and Assessment AUTO – Automation FD – Flight Dynamics MAS – Modeling and Simulation PAS – Planning and Scheduling SC – Scripting Control TAC – Telemetry and Command	Product type being requested. (See Table A-2, Product Categories).
PROD-SUBTYPE	String	R	Product type and subtype being requested. (See Table A-2 Product Categories.)	Product subtype being requested. (See Table A-2, Product Categories).
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R	1+	Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6*, etc. in Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D		First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc. of the Product Message) - "n" starts at "1". This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
RESPOND-VIA-MSG	String	R	**Value** – **Description** "MSG.PROD" – Product Message "RESP.PROD" – Product Response Message	Indicates the message to use to deliver the product.
REQUEST-ID	GUID	R		ID to identify the request message
DELIVER-VIA-REFERENCE	Boolean	O		Indicates if the product will be referenced by a URI in the message specified above.
DELIVER-VIA-INCLUDE	Boolean	O		Indicates if the product is to be included in the message specified above.
URI	String	O		Location where the requesting component is asking for the product file(s) to be stored. Could be a web address, directory, or folder specification
NAME-PATTERN	String	O		Describes the file name

Field Name	Type	Presence	Value	Description
FORMAT	String	O		For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
VERSION	String	O		Identifies the version ID of the file
SIZE	U32	O	KB	Maximum size of the file acceptable to the requester. Size specified in KB.

For an explanation on how the START-TIME and STOP-TIME could operate, see Table 8-20 Examples of Start and Stop Times.

The Product service provider should **take care not to allow a database query, script expression, or other executable string in REQ-STRING** as this would create an exploitable security concern. Specifically, the service requestor would thereby be able to provide an expression to be executed directly on the service provider at the service provider's level of privilege. Instead, REQ-STRING should only be used to convey some keyword to the service provider to indicate the type of action to be performed.

8.12.2 Product Response Message

A thorough description of this message is given in Section 8.12 Product Messages.

8.12.2.1 Product Response Message Subjects

Table 8-156. Product Response Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	PROD	[DESTINATION-COMPONENT]	[Status]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	PROD	USER1	1				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	PROD	SCHED	1				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	RESP	PROD	JOE	*				

Table 8-157. Properties of the *Miscellaneous Elements* for the Product Response Message

Miscellaneous Element	Required / Optional	Description	Field Origination in Msg, if applicable
ME1	Required	Component name of Requestor	DESTINATION-COMPONENT from header
ME2	Required	Status type supplied by Responder	RESPONSE-STATUS

Examples

Two components, the Scheduler (SCHED) (the Data Requestor/Subscriber/Client) and FD (the Data Provider/Publisher/Server) interact with the Product Response Message.

FD sends a response message to the Scheduler (SCHED)

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.PROD.SCHED.1` *or*

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.PROD.SCHED.4`

SCHED subscribes to receive the Product Response Message.

`C2MS.*.*.MSSN.*.*.RESP.PROD.SCHED.>` *or*

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.PROD.SCHED.>`

8.12.2.2 Product Response Message Header

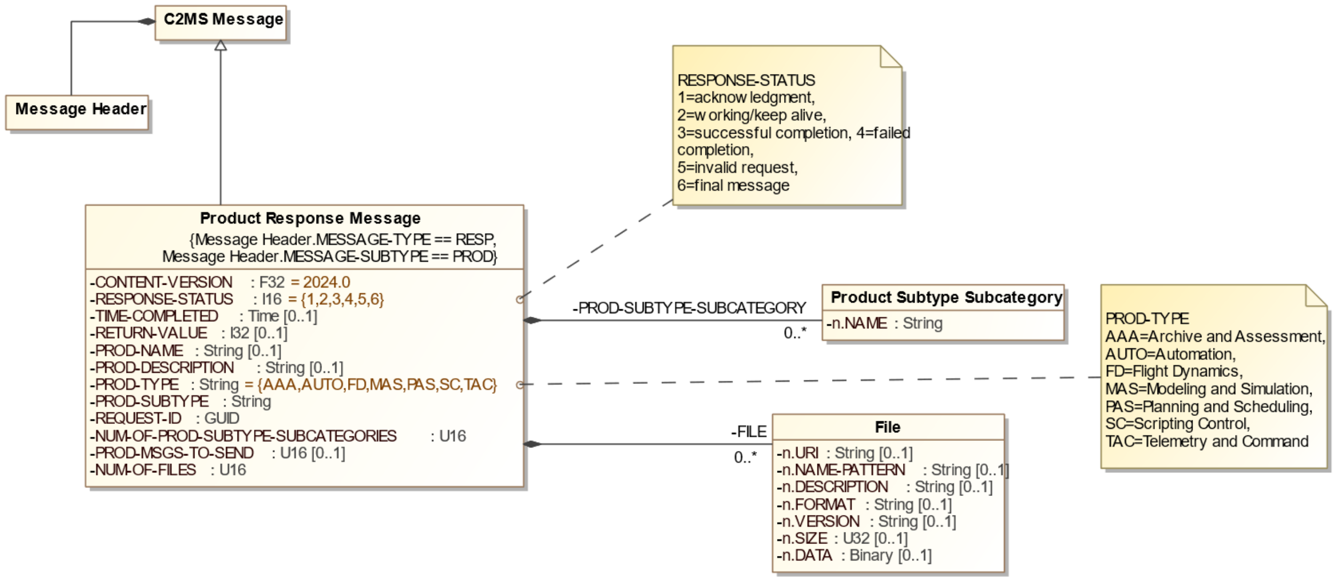
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Product Response Message.

Table 8-158. Product Response Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	PROD

8.12.2.3 Product Response Message Contents

The following figure shows a UML object diagram of the Product Response Message with its required and optional fields.



C2MS Message
Message Header
Product Response Message
{Message Header.MESSAGE-TYPE == RESP, Message Header.MESSAGE-SUBTYPE == PROD}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1,2,3,4,5,6}
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-PROD-NAME : String [0..1]
-PROD-DESCRIPTION : String [0..1]
-PROD-TYPE : String = {AAA,AUTO,FD,MAS,PAS,SC,TAC}
-PROD-SUBTYPE : String
-REQUEST-ID : GUID
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-PROD-MSGS-TO-SEND : U16 [0..1]
-NUM-OF-FILES : U16
RESPONSE-STATUS
1=acknowledgment,
2=working/keep alive,
3=successful completion, 4=failed completion,
5=invalid request,
6=final message
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
PROD-TYPE
AAA=Archive and Assessment,
AUTO=Automation,
FD=Flight Dynamics,
MAS=Modeling and Simulation,
PAS=Planning and Scheduling,
SC=Scripting Control,
TAC=Telemetry and Command
-FILE
0..*
File
-n.URI : String [0..1]
-n.NAME-PATTERN : String [0..1]
-n.DESCRIPTION : String [0..1]
-n.FORMAT : String [0..1]
-n.VERSION : String [0..1]
-n.SIZE : U32 [0..1]
-n.DATA : Binary [0..1]

Figure 8-33. Product Response Message Diagram

The following table describes additional field names, values, and notes for the Product Response Message.

Table 8-159. Product Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the Product Request Message that was processed.
			1	Acknowledgement	
			2	Working / Keep Alive	
			3	Successful Completion	
			4	Failed Completion	
			5	Invalid Request	
			6	Final Message	
TIME-COMPLETED	Time	O			Time application completed processing the request
RETURN-VALUE	I32	O			Return value or status based on the RESPONSE-STATUS. Used to provide function call status or error code in the case of failed completion
PROD-NAME	String	O			Name of the product
PROD-DESCRIPTION	String	O			Description of the product in text or xml
PROD-TYPE	String	R	**Value**	**Description**	Echo of the PROD-TYPE field from the Product Request message.
			AAA	Archive and Analysis	
			AUTO	Automation	
			FD	Flight Dynamics	

Field Name	Type	Presence	Value		Description
			MAS	Modeling and Simulation	
			PAS	Planning and Scheduling	
			SC	Scripting Control	
			TAC	Telemetry and Command	
PROD-SUBTYPE	String	R	Product type and subtype being requested. (See Table A-2, Product Categories).		Product type and subtype being requested. (See Table A-2, Product Categories).
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R	0+		Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6,* etc. in Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D			First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc. of the Product Message) - "n" starts at "1". This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
PROD-MSGS-TO-SEND	U16	O	0+		Indicates the number of Product Messages that will be published to satisfy the PROD REQ. A value of "-1" can be used to indicate "Unknown".
NUM-OF-FILES	U16	R	0+		Indicates the number of files included in this response message.
FILE.n.URI	String	O			URI specifying the location where the file of the product is stored
FILE.n.NAME-PATTERN	String	O			Describes the file name
FILE.n.DESCRIPTION	String	O			Description of the file in text or xml
FILE.n.FORMAT	String	O			For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
FILE.n.VERSION	String	O			Identifies the version ID of the file
FILE.n.SIZE	U32	O	KB		Size of the included file
FILE.n.DATA	Binary	O			The file content

Note: The C2MS Product Message and Product Response Messages are used for a single product, that is, one product per message. The Product Response and Product Messages allow for multiple files per product.

Table 8-160. Meaning of RESPONSE-STATUS and RETURN-VALUE with Recommended Actions

User Specified the URI	RESPONSE-STATUS	RETURN-VALUE	URI	Action
N	Successful	2 (The only meaningful value)	Product was generated and placed in URI chosen by responder	Requestor should retrieve file at responder's URI location
Y	Successful	1	Product was generated and placed in URI specified by requestor	Requestor should retrieve file at the specified URI
Y	Successful	2	Product was generated but placed in alternate URI chosen by responder	Requestor should retrieve file at responder's URI location
Y or N	Failed	3	Product exceeded maximum requested file size or the default maximum file size	

8.12.3 Product Message

The Product Message can be used by itself or in conjunction with the Product Request and Product Response Messages. When the Product Message is used by itself, it can contain a notification of product availability or contain the product itself. This is dependent upon the system design and mechanism chosen for product delivery.

The Product Message can also be used with the Product Request and Product Response Messages in one of the two Triad sequences. Some examples of the use of the set of Product Messages are provided in the following table.

Table 8-161. Example Scenarios Using the Set of Product Messages

Service	Message Exchange Pattern	Step 1 Message	Step 2 Message	Step 3 Message
Announce Product Availability	Publish	**Product Message:** Contains information about new product	Option 1: Consumer can retrieve product Option 2: Consumer must request product	
Deliver Product Automatically	Publish	**Product Message:** Contains product		
Deliver Available Product Upon Request	Request Response	**Product Request:** Consumer requests product	**Product Response:** Producer delivers product	
Generate and Deliver Product Upon Request	Triad 1 (Req/ Resp/ Msg)	**Product Request:** Consumer requests product generation and delivery	**Product Response:** Producer responds with status, begins product generation	**Product Message:** Producer delivers generated product

Service	Message Exchange Pattern	Step 1 Message	Step 2 Message	Step 3 Message
Announce and Deliver Product Upon Request	Triad 2 (Msg/ Req/ Resp)	**Product Message:**	**Product Request:**	**Product Response:**
		Producer announces product availability	Consumer requests product	Producer delivers product

8.12.3.1 Product Message Subjects

Table 8-162. Product Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*						
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***	***ME7***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constellation]	[sat]	MSG	PROD	[COMPONENT]	[PROD-NAME]	[PROD-TYPE]	[PROD-SUBTYPE]			[request id]
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	PROD	FD	DAY304	FD	ORBEVT			[GUID]
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	PROD	FD	MAN55	FD	MAN			[GUID]
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	PROD	FD	*	FD	ORBEVT			[GUID]

Note that "[GUID]" in the table above is a Subject Token String that conforms to the GUID type specified in this document, such as "b891bdac-964a-4f3e-957c-1a29ec4c7d50" or similar.

Table 8-163. Properties of the *Miscellaneous Elements* for the Product Message

Miscellaneous Element	**Required / Optional**	**Used For**	**Description**	**Field in Msg, if applicable**
ME1	Required	Publishing Component	Component name of Publisher	COMPONENT from header
ME2	Required	Product Name	The Name of the product	PROD-NAME
ME3	Required	Product Type or Class	Categorization of the Product type. (See Table A-2, Product Categories).	PROD-TYPE
ME4	Required	Product Subtype or Subclass	Sub-categorization of the Product Type. See above.	PROD-SUBTYPE
ME5	As necessary	Product Subtype Subcategory 1	Sub-categorization of the PROD-SUBTYPE	See "*ME5* and *ME6*" note below.
ME6	As necessary	Product Subtype Subcategory 2	Sub-categorization of the above	See "*ME5* and *ME6*" note below.
ME7	Optional	Request ID	ID associated with original request	REQUEST-ID

***ME5* and *ME6* note:** The subject elements *ME5, ME6, ME7* and so on are used to categorize and sub-delineate the products. As many subject elements as necessary can be used to categorize the variety and potentially voluminous number of products. The subscriber may not always know the *ME2* element (PROD-NAME) or the number of subject elements used beyond the basic categorization of product type (*ME3*) and product subtype (*ME4*). In this case, the subscriber should wildcard (*) the *ME2* element and open end (>) the subject elements beyond *ME4*.

Examples

The Data Provider/Publisher/Server sends out the unsolicited Product Message with the following subject:

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.PROD.`*`FD.ORBEVT.FD.OE`*

The Requestor/Subscriber/Client subscribes to receive a Product Message categorized with a product type and subtype. It wildcard's the *ME1* and *ME2* fields of component and product name (PROD-NAME), respectively.

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.PROD.*.*.FD.OE.>` ***or***

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.PROD.FD.ORBEVT.FD.OE.PERAPTIME.>`

`FD` – Component name of product publisher

`ORBEVT` –PROD-NAME of the product

`FD` – Product Type (FD = Flight Dynamics)

`OE` – `Product` Subtype (OE = Orbital Event)

`PERAPTIME` - A subtype of `OE`. `PERAPTIME` refers to a product that contains the perigee and apogee times of the orbit.

8.12.3.2 Product Message Header

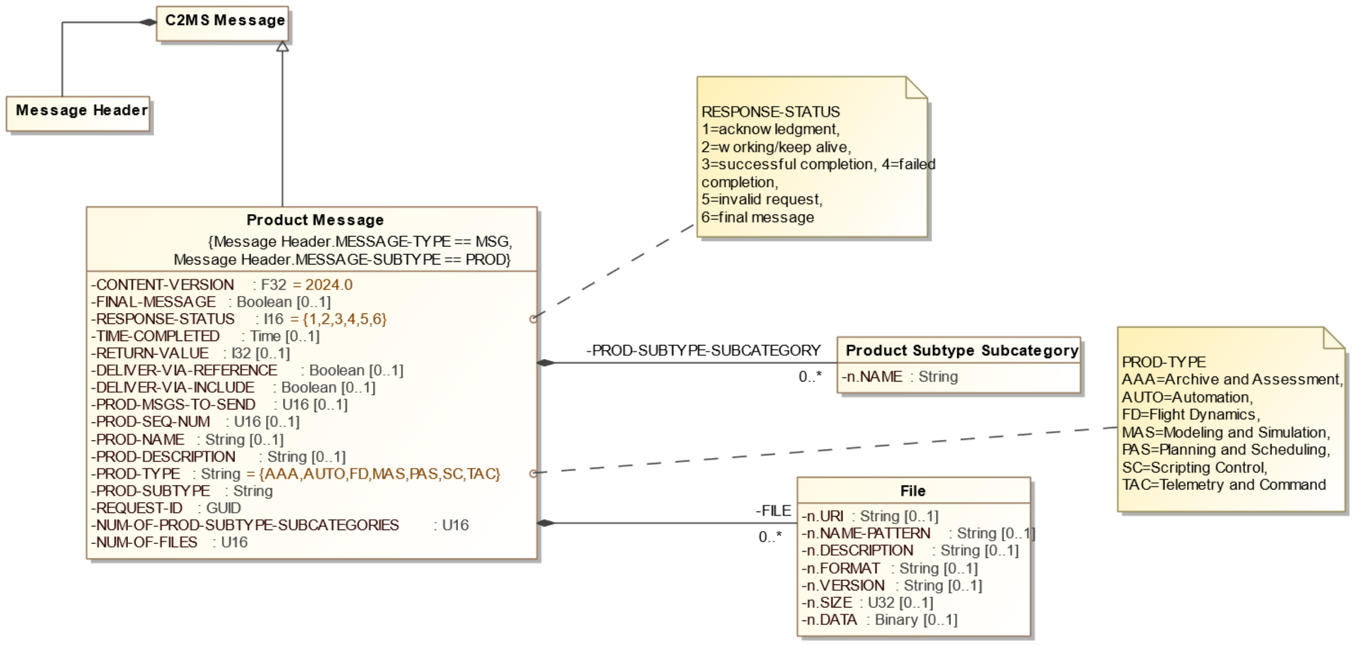
The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Product Message header.

Table 8-164. Product Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	PROD

8.12.3.3 Product Message Contents

The following figure shows a UML object diagram of the Product Message with its required and optional fields.



C2MS Message
Message Header
Product Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == PROD}
-CONTENT-VERSION : F32 = 2024.0
-FINAL-MESSAGE : Boolean [0..1]
-RESPONSE-STATUS : I16 = {1,2,3,4,5,6}
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-DELIVER-VIA-REFERENCE : Boolean [0..1]
-DELIVER-VIA-INCLUDE : Boolean [0..1]
-PROD-MSGS-TO-SEND : U16 [0..1]
-PROD-SEQ-NUM : U16 [0..1]
-PROD-NAME : String [0..1]
-PROD-DESCRIPTION : String [0..1]
-PROD-TYPE : String = {AAA,AUTO,FD,MAS,PAS,SC,TAC}
-PROD-SUBTYPE : String
-REQUEST-ID : GUID
-NUM-OF-PROD-SUBTYPE-SUBCATEGORIES : U16
-NUM-OF-FILES : U16
RESPONSE-STATUS
1=acknowledgment,
2=working/keep alive,
3=successful completion, 4=failed completion,
5=invalid request,
6=final message
-PROD-SUBTYPE-SUBCATEGORY
0..*
Product Subtype Subcategory
-n.NAME : String
PROD-TYPE
AAA=Archive and Assessment,
AUTO=Automation,
FD=Flight Dynamics,
MAS=Modeling and Simulation,
PAS=Planning and Scheduling,
SC=Scripting Control,
TAC=Telemetry and Command
-FILE
0..*
File
-n.URI : String [0..1]
-n.NAME-PATTERN : String [0..1]
-n.DESCRIPTION : String [0..1]
-n.FORMAT : String [0..1]
-n.VERSION : String [0..1]
-n.SIZE : U32 [0..1]
-n.DATA : Binary [0..1]

Figure 8-34. Product Message Diagram

The following table describes additional field names, values, and notes for the Product Message.

Table 8-165. Product Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
FINAL-MESSAGE	Boolean	O			When true, indicates the last message in the stream.
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the Product Message that was processed. Note: Even though a Request is valid, a product may not be able to be successfully generated. In this case the following Product Message would indicate a Failed Completion. Only required for RESP
			1	Acknowledgement	
			2	Working / Keep Alive	
			3	Successful Completion	
			4	Failed Completion	
			5	Invalid Request	
			6	Final Message	
TIME-COMPLETED	Time	O			Time application created the product

Field Name	Type	Presence	Value	Description
RETURN-VALUE	I32	O		Return value or status based on the RESPONSE-STATUS. Used to provide function call status or error code in the case of failed completion.
DELIVER-VIA-REFERENCE	Boolean	O		Indicates the product is referenced by a URI. A product can be included in a message, referenced by a URI, or both.
DELIVER-VIA-INCLUDE	Boolean	O		Indicates the product is included in this message. A product can be included in a message, referenced by a URI, or both.
PROD-MSGS-TO-SEND	U16	O	0+	Indicates the number of Product Messages that will be published to satisfy the PROD REQ
PROD-SEQ-NUM	U16	O	1+	Indicates which message this is in the sequence of Product Messages that constitutes a product
PROD-NAME	String	O		Name of the product
PROD-DESCRIPTION	String	O		Description of the product in text or xml
PROD-TYPE	String	R	**Value** — **Description** AAA — Archive and Analysis AUTO — Automation FD — Flight Dynamics MAS — Modeling and Simulation PAS — Planning and Scheduling SC — Scripting Control TAC — Telemetry and Command	Category of product. Could be echo of the PROD-TYPE field from the Product Request message
PROD-SUBTYPE	String	R	Product type and subtype being requested. (See Table A-2, Product Categories).	Subcategory of the product. Could be echo of the PROD-SUBTYPE field of the Product Request Message
REQUEST-ID	GUID	R		This field's value is to be the same as the REQUEST-ID in the associated REQ message.
NUM-OF-PROD-SUBTYPE-SUBCATEGORIES	U16	R	1+	Number of further delineations / categories beyond the product subtype. Also, used as msg subject elements *ME5, ME6*, etc. in Product Message.
PROD-SUBTYPE-SUBCATEGORY.n.NAME	String	D		First subcategory of the product subtype. (Subject elements *ME5, ME6*, etc.) – "n" starts at "1". This field is required for each PROD-SUBTYPE-SUBCATEGORY specified by NUM-OF-PROD-SUBTYPE-SUBCATEGORIES.
NUM-OF-FILES	U16	R	0+	Indicates the number of files included in this response message.
FILE.n.URI	String	O		URI specifying the location where the file of the product is stored – "n" starts at "1".
FILE.n.NAME-PATTERN	String	O		Describes the file name
FILE.n.DESCRIPTION	String	O		Description of the file in text or xml

Field Name	Type	Presence	Value	Description
FILE.n.FORMAT	String	O		For application use. This field describes the file format as agreed upon between the producer and consumer of this message.
FILE.n.VERSION	String	O		Identifies the version ID of the file
FILE.n.SIZE	U32	O	KB	Size of the included file
FILE.n.DATA	Binary	O		The file content

8.13 Simple Service Messages

C2MS intends to rework this set of messages in a future release, deprecating the current set and replacing them with a completely new set of messages. This is for the purpose of moving away from some constructs such as the dual-use of the DESTINATION-COMPONENT field as well as the dual-use of the Simple Service Request Message as either a service request or a completely unrelated publish. At present, these messages are retained as-is in C2MS for backward compatibility. Meanwhile, C2MS encourages limiting the use of Simple Service Messages to early-entry service providing components that have not yet had an opportunity to define their own component-specific interfaces.

In many service-oriented designs, a one-to-one message exchange will occur between the consumer and producer of a service. That is, the consumer will make a request of the producer of the product or service, and the producer will, in turn, reply with a response. Many such message exchanges are possible using a complementary pair of Request and Response type messages. The pair of Simple Service Messages is a general mechanism for the invocation of a service from one component to another. Software components wishing to expose or make certain services available to other components can utilize this general mechanism for that purpose.

The Simple Service Messages are similar in nature and function to the Directive Messages. Where the Directive Request Message will use a keyword or text string to request some functionality of a component, the Simple Service Request Message will make a service request by name, number, and/or operation. It will also pass in any parameters that are required. The Simple Service Messages do not provide for files to be included in the messages though a block of data can be returned in the Simple Service Response message. Also, it is expected that the number of services offered by a component will be small enough that name or number can easily identify them.

The intention of the Simple Service Messages is to allow a component to offer a small number of simple services to other components and the system in general. They are not meant to provide a comprehensive service framework. Simple Service Messages are not intended to provide permanent access to service capabilities, but rather, to provide a way for onboarding new service capabilities quickly. The process of maturing service-providing components should include establishing their own component-specific interfaces, leaving behind the use of Simple Service Messages. Services are expected to be provided locally and not across domains, thereby allowing (or requiring) for all provided services to be uniquely named within the immediate service area. Thus, any component could request any service, by name, offered by another component, if authorized, within the local domain. Because a service name may be used as a subject name element (*ME1*) it must follow the same syntax as elements in a message subject.

8.13.1 Simple Service Request Message

A Simple Service Request Message is a service request that is issued to one application from another. A service request may also be input from a user through a GUI or command line, or as part of the internal logic of a component. Services could also be grouped together in a procedure (or proc), an executable schedule, or other such orchestration techniques. As components become less coupled, they may tend to offer or provide more services and thus enable more rapid software development with orchestrated modules. The Simple Service Request Message will request the invocation of a single service from a single component.

8.13.1.1 Simple Service Request Message Subjects

In most request/response message exchange patterns, the *ME1* element is used to identify the requestor and responder of the request. With the Simple Service messages, the *ME1* element may also be used to identify the service. That is, the unique name of the service (following the syntax of the message subject name elements) is inserted into element *ME1*. When the name of the service is inserted into the *ME1* element, the message subject elements TYPE, SUBTYPE, and *ME1* would appear as follows:

… REQ.SERV.[SERVICENAME]
and
… RESP.SERV.[SERVICENAME] …

The above syntax shows the message subject for a service request message and service response message. In service terminology, the requestor is known as the consumer of the service and the responder is known as the producer or provider. This message subject syntax allows the consumer to request a service without knowledge of the name of the component providing the service. The provider of the service should respond in kind using the service name in the *ME1* element. As should be obvious, the producer must subscribe not only to messages subjects using *ME1* as a component name, but also to message subjects using *ME1* as a service name. *If a single producer provides a large number of services, it will require an equally large number of message subject subscriptions to manage, and this convention may not be desirable.*

Services are expected to be provided locally and not across domains, thereby allowing for all provided services to be uniquely named within the immediate service area. If the service cannot be uniquely named within the service area, then the *ME1* and *ME2* elements can be employed together to uniquely identify all services, similar to the mission-satellite and message type-subtype pairings. The *ME2* element will serve as the general subject matter or group, and the *ME1* element will identify the service, uniquely named, within that group.

Since a service name may be used as a subject name element it must follow the same syntax as elements in a message subject. See 6.2.2 Format of C2MS Subject Names.

In order to distinguish service names from component names, naming conventions may be established. For example, all services could be named as “S_NNNN”, and/or all components could be named “C_NNNN”.

Table 8-166. Simple Service Request Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	REQ	SERV	[DESTINATION-COMPONENT or SERVICE-NAME]	[service group]	[operati on]			
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	SERV	FD	DAY304	FD			
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	REQ	SERV	FD	MAN55	FD			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	REQ	SERV	FD	*	FD			

Table 8-167. Properties of the Miscellaneous Elements for the Simple Service Request Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field in Msg, if applicable**
ME1	Required	Component name of producer, or name of the service	DESTINATION-COMPONENT from header; or SERVICE-NAME
ME2	Optional	Functional area, subject matter, or group to which the service belongs	SERVICE-GROUP
ME3	Optional	Name of the operation within the service	OPERATION-NAME
ME4 ...	Not used		

The *ME2* and *ME3* elements serve as a two-element pair to uniquely identify all services, similar to the mission-satellite and message type-subtype pair. Should the name of the service not be unique, the *ME2* element can be utilized to avoid ambiguity.

Examples:

Two components, APP4 and APP1, interact with the Simple Service Request Message. APP4 sends a Simple Service Request Message to APP1.

APP4 subject to send a Simple Service Request for a particular service to APP1:

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.APP1`

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.SERVICEA` (using the service name convention in *ME1*)

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.SERVICEA.GROUPB` (using the optional service name convention of *ME1* and *ME2*)

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.SERVICEA.GROUPB.OPERATION1` (using all the *ME* elements)

APP1 message subject subscription to receive a request for a particular service using the service name convention in *ME1 (*assumes all services are uniquely named):

`C2MS.*.*.*.*.*.*.*.SERVICEA.>`

APP1 message subject subscription to receive a request for a particular service when service names are not unique, using the *ME1* and *ME2* elements.

`C2MS.*.*.*.*.*.*.*.SERVICEA.GROUPB.>`

APP1 message subject subscription to receive a request for a particular operation within a service by using all the subject elements.

`C2MS.*.*.*.*.*.*.*.SERVICEA.GROUPB.OPERATION1`

APP1 message subject subscription to receive requests for any of its provided services when the service naming convention is not used:

`C2MS.*.*.*.*.*.REQ.SERV.APP1`

APP1 subjects to receive all requests for any of its services:

`C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.APP1.>`

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.SERVICEA.>
C2MS.*.*.*.*.*.REQ.SERV.SERVICEB.>
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.REQ.SERV.SERVICEC.>
```
and so on for each service.

8.13.1.2 Simple Service Request Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Simple Service Request Message header.

Table 8-168. Simple Service Request Message Header Field Values

Header Field Name	Value
MESSAGE-TYPE	REQ
MESSAGE-SUBTYPE	SERV

8.13.1.3 Simple Service Request Message Contents

The following figure shows a UML object diagram of the Simple Service Request Message with its required and optional fields.

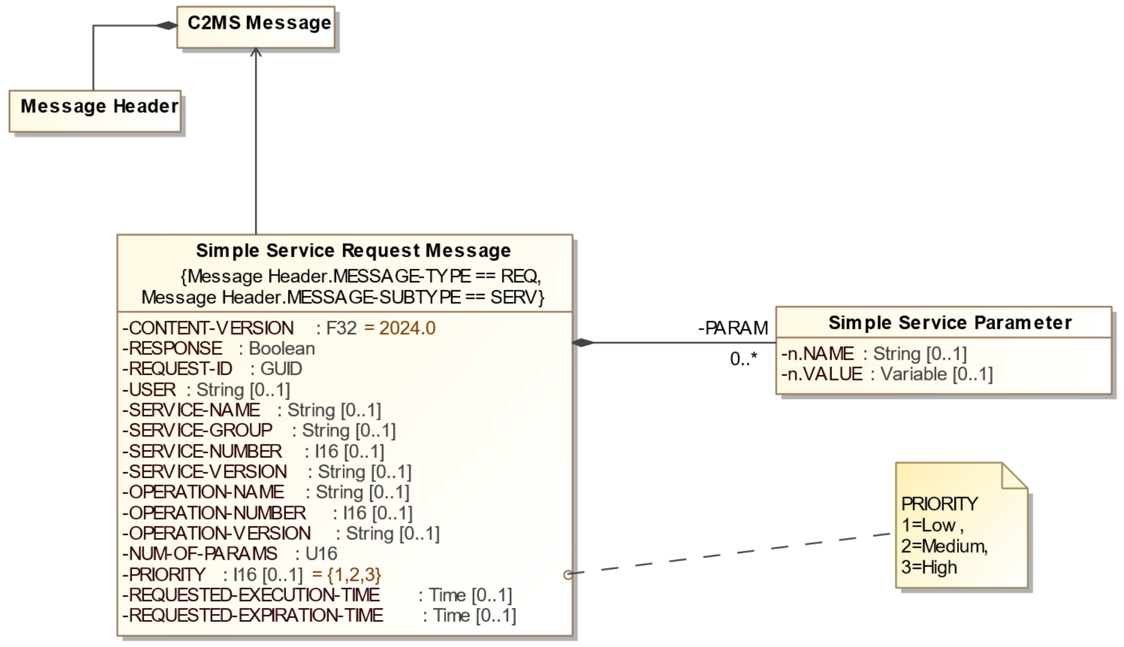

C2MS Message
Message Header
Simple Service Request Message
{Message Header.MESSAGE-TYPE == REQ,
Message Header.MESSAGE-SUBTYPE == SERV}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE : Boolean
-REQUEST-ID : GUID
-USER : String [0..1]
-SERVICE-NAME : String [0..1]
-SERVICE-GROUP : String [0..1]
-SERVICE-NUMBER : I16 [0..1]
-SERVICE-VERSION : String [0..1]
-OPERATION-NAME : String [0..1]
-OPERATION-NUMBER : I16 [0..1]
-OPERATION-VERSION : String [0..1]
-NUM-OF-PARAMS : U16
-PRIORITY : I16 [0..1] = {1,2,3}
-REQUESTED-EXECUTION-TIME : Time [0..1]
-REQUESTED-EXPIRATION-TIME : Time [0..1]
-PARAM
0..*
Simple Service Parameter
-n.NAME : String [0..1]
-n.VALUE : Variable [0..1]
PRIORITY
1=Low ,
2=Medium,
3=High

Figure 8-35. Simple Service Request Message Diagram

The following table describes additional field names, values, and notes for the Simple Service Request Message.

Table 8-169. Simple Service Request Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2019	Version number for this message content description
RESPONSE	Boolean	R		Indicates if a response is required
REQUEST-ID	GUID	R		ID to identify the request message
USER	String	O		Which user/workposition/proc/schedule the message is coming from
SERVICE-NAME	String	O		Name of the service offered by a component. Note that if the SERVICE-NAME is being used in the Message Subject (ME1), then the SERVICE-NAME field must conform to the specialized String type, Subject Token String.
SERVICE-GROUP	String	O		Functional area, subject matter, or group to which the service belongs. Note that if the SERVICE-GROUP is being used in the Message Subject (ME2), then the SERVICE-GROUP field must conform to the specialized String type, Subject Token String.
SERVICE-NUMBER	I16	O		Number of the service
SERVICE-VERSION	String	O		Version of the service
OPERATION-NAME	String	D		Name of the operation within the specified service. An operation name or number may be specified, but not both.
OPERATION-NUMBER	I16	D		Number of the operation within the specified service. An operation name or number may be specified, but not both.
OPERATION-VERSION	String	O		Version of the operation within the specified service
NUM-OF-PARAMS	U16	R		Number of parameters included within the service request
PARAM.n.NAME	String	O		Name of the parameter - “n” starts at “1”.
PARAM.n.VALUE	Variable	O		Value of the parameter; Component must ascertain the data type before accessing the value (e.g., with a function call)
PRIORITY	I16	O	**Value** **Description** 1 Low 2 Medium 3 High	Indicates processing priority, if applicable
REQUESTED-EXECUTION-TIME	Time	O		Absolute or relative time can apply
REQUESTED-EXPIRATION-TIME	Time	O		Absolute or relative time can apply

Services may evolve over time and services may offer a number of variations or operations for a particular service. A service may be identified by name or number. The requestor may choose either option. If there are a number of operations for that service, then the requestor must specify which operation, by name or number is being requested. When service and operation parameters are not specified, the default will be the first service and first operation within that service as determined by the provider.

The Simple Service Request Message can be used to:

1. Request a service
2. Provide an unsolicited service

To request a service and receive a response via the Simple Service Response Message, the requestor must mark the RESPONSE field as true.

The Simple Service Request Message can also be used to provide an unsolicited service. That is, an application can provide a set of data to other applications automatically, without them having to issue a request. The service provider simply populates the "Parameters" fields of the Simple Service Request Message with the data to be published. The application can also identify the service in the "Service Information" fields. Additionally, the provider of the unsolicited service will publish the message with the name of the service in the *ME1* element of the message subject. Applications wanting to avail themselves of the service will subscribe to the message subject for that service.

As an example, at the conclusion of a pass, an application will automatically provide a description of the pass that includes the start and stop times, the collection point, and the satellite. This data can be inserted into the "Parameter" fields along with the service name, say, 'PASSDESC", in the "SERVICE-NAME" field. Also, the "RESPONSE" field is set to false. Then the message is published with 'PASSDESC" in the *ME1* element of the message subject. Applications wishing to receive this pass description data automatically can easily subscribe to this message and receive all pass descriptions whenever they are published.

Note that for unsolicited services, the Simple Service Request Message will be sent with a "Publish" message exchange pattern.

8.13.2 Simple Service Response Message

A Simple Service Response Message is sent by an application in response to a Simple Service Request Message. The Simple Service Response Message will provide acknowledgment of the Simple Service Request Message, a status of the action completed, and any data to be returned. A series of Simple Service Response Messages may be required in the case where the processing of the action is lengthy. An example of this would be a complex mathematical calculation using a large volume of data. In this event, an interim or interactive "working" type message would be issued to let the original application know that the action is still being processed. Please see Section 6.4 C2MS Messages: Their Characteristics and Interactions for a general discussion on these types of messages.

8.13.2.1 Simple Service Response Message Subjects

Table 8-170. Simple Service Response Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	RESP	SERV	[DESTINATION-COMPONENT or service name]	[RESPONS E-STATUS]				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	SERV	FD	DAY304				
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	RESP	SERV	FD	MAN55				
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	RESP	SERV	FD	*				

Table 8-171. Properties of the *Miscellaneous Elements* for the Simple Service Response Message

Miscellaneous Element	**Required / Optional**	**Description**	**Field Origination in Msg, if applicable**
ME1	Required	Component name of consumer or name of service	DESTINATION-COMPONENT from header, or "SERVICE-NAME" from content of Request msg
ME2	Required	Status type supplied by Responder	"RESPONSE-STATUS" from content of Response message

Examples:
Two components, APP4 and APP1, interact with the Simple Service Response message. APP1 sends a Simple Service Response Message to APP4.

APP1 subject to send the Simple Service Response to APP4:
```
        C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.SERV.APP4.1   or
        C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.SERV.SERVICEA.1
```

APP4 subscribes to receive its own Simple Service Response Messages:
```
        C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.RESP.SERV.APP4.>   or
        C2MS.*.*.*.*.*.RESP.SERV.APP4.>
```

8.13.2.2 Simple Service Response Message Header

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the Simple Service Response Message header.

Table 8-172. Simple Service Response Message Header Field Values

Header Field Name	**Value**
MESSAGE-TYPE	RESP
MESSAGE-SUBTYPE	SERV

8.13.2.3 Simple Service Response Message Contents

The following figure shows a UML object diagram of the Simple Service Response Message with its required and optional fields.

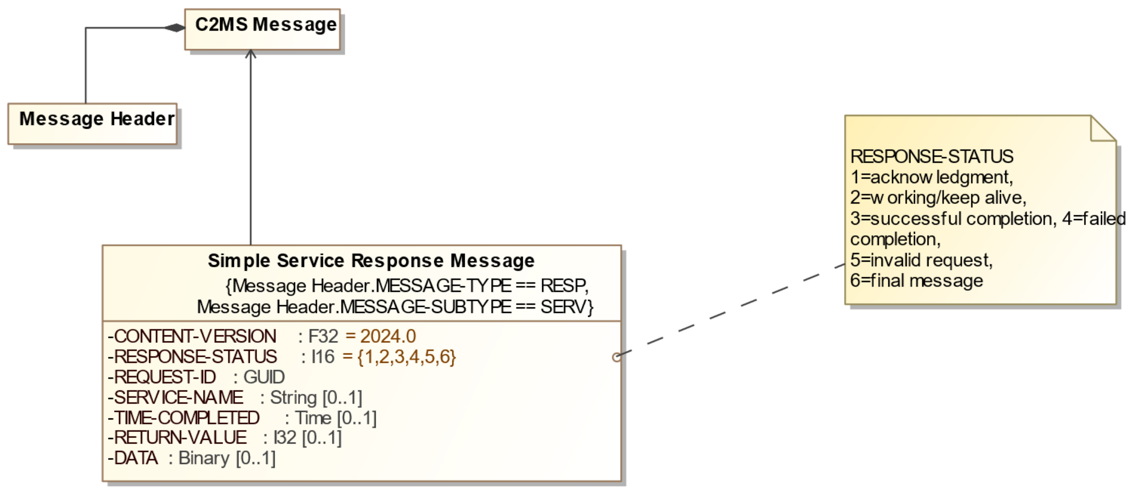

C2MS Message
Message Header
Simple Service Response Message
{Message Header.MESSAGE-TYPE == RESP,
Message Header.MESSAGE-SUBTYPE == SERV}
-CONTENT-VERSION : F32 = 2024.0
-RESPONSE-STATUS : I16 = {1,2,3,4,5,6}
-REQUEST-ID : GUID
-SERVICE-NAME : String [0..1]
-TIME-COMPLETED : Time [0..1]
-RETURN-VALUE : I32 [0..1]
-DATA : Binary [0..1]
RESPONSE-STATUS
1=acknowledgment,
2=working/keep alive,
3=successful completion, 4=failed completion,
5=invalid request,
6=final message

Figure 8-36. Simple Service Response Message Diagram

The following table describes additional field names, values, and notes for the Simple Service Response Message.

Table 8-173. Simple Service Response Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2024		Version number for this message content description
RESPONSE-STATUS	I16	R	**Value**	**Description**	Identifies the status of the Simple Service being processed
			1	Acknowledgement	
			2	Working/keep alive	
			3	Successful completion	
			4	Failed completion	
			5	Invalid Request	
			6	Final Message	
REQUEST-ID	GUID	R			This field's value is to be the same as the REQUEST-ID in the associated REQ message.
SERVICE-NAME	String	O			Name of the service offered by a component. Note that if the SERVICE-NAME is being used in the Message Subject (ME1), then the SERVICE-NAME field must conform to the specialized String type, Subject Token String.
TIME-COMPLETED	Time	O			Time application completed processing the Simple Service
RETURN-VALUE	I32	O			Return value or status based on the RESPONSE-STATUS. Provides additional status information particular to the request.
DATA	Binary	O			Additional data that may accompany the response

8.14 Navigation Data Messages

Within the CCSDS recommended standards there are definitions for various telemetry messages and formats (frames and packets). The C2MS includes a corresponding message set to encapsulate the CCSDS telemetry message definitions. (Additionally, C2MS includes other types of non-CCSDS telemetry messages.) See Section 8.7 Real-time Telemetry Data Messages.

Currently, CCSDS has defined a set of six navigation data messages for attitude, orbit, and tracking data. It is expected that these types of definitions will grow in number. The C2MS will follow a similar course of action with the navigation data messages as it has with the telemetry messages. As a result, the C2MS will be able to provide support for the existing set of navigation data messages and be extensible in anticipation of accommodating additional navigation data messages.

There are three basic types of CCSDS Navigation Data Messages (NDM).

- Attitude Data Message (ADM)
- Orbit Data Message (ODM)
- Tracking Data Message (TDM)

ADMs are used to convey spacecraft attitude information. These can include:

- Attitude Parameter Message (APM) – Consists of instantaneous attitude state and optional attitude maneuvers.
- Attitude Ephemeris Message (AEM) – Consists of a history/forecast of the attitude of the object that can be interpolated to ascertain the attitude of the object at other times.

ODMs are used to convey trajectory information. These can include:

- Orbit Parameter Message (OPM) – Consists of a single state vector at a given time that represents the trajectory of the object.
- Orbit Mean-Elements Message (OMM) – Consists of a single object at a specified epoch expressed in mean Keplerian elements.
- Orbit Ephemeris Message (OEM) – Consists of a history/forecast of state vectors that can be interpolated to ascertain the trajectory of the object at other times.

TDMs are used to convey a variety of tracking data used in the orbit determination process.

For example:

- Doppler and range radiometrics in a variety of tracking modes
- Very-long-baseline Interferometry (VLBI) data, antenna pointing angles, etc.

The CCSDS navigation data messages are summarized in the following table.

Table 8-174. CCSDS Navigation Data Messages

Message	Contents	Purpose	CCSDS Document	CCSDS Doc. No.
Attitude Parameter Message (APM)	Attitude state for single object at single epoch	Suitable for exchanges that 1) are automated and/or have human interaction; 2) do not require high fidelity dynamic modeling	Attitude Data Messages	504.0-B-1 (05/2008)
Attitude Ephemeris Message (AEM)	Attitude state for single object at multiple epochs	Suitable for exchanges that 1) automated; 2) require high fidelity dynamic modeling		
Orbit Parameter Message (OPM)	Position and velocity of a single object at a specified epoch	Suitable for exchanges that 1) are automated and/or have human interaction; 2) do not require high fidelity dynamic modeling	Orbit Data Messages	502.0-B-2 (11/2009)
Orbit Mean-Elements Message (OMM)	Specifies orbital characteristics of a single object at a specified epoch	Suitable for exchanges that 1) are automated and/or have human interaction; 2) do not require high fidelity dynamic modeling		
Orbit Ephemeris Message (OEM)	Position and velocity of a single object at multiple epochs within a single time range	Suitable for exchanges that 1) automated; 2) require high fidelity dynamic modeling		
Tracking Data Message	Tracking data for one or more tracking participants at multiple epochs within a specific time range	Convey a variety of tracking data used in the orbit determination process in a single message	Tracking Data Message	503.0.B-1 (11/2007)

CCSDS Navigation messages can be exchanged in one of two formats: keyword value notation (KVN) and XML, the “eXtensible Markup Language”. XML is a better format for specifying the ASCII-based data in the messages.

The schema for these types of messages can be found at the CCSDS web site public.ccsds.org in the document “XML Specification for Navigation Data Messages (CCSDS 505.0.B-1, 12/2010). Additionally, other non-CCSDS navigation data messages may be exchanged. These messages could be in ASCII format, binary, or even a raw tracking data stream.

Attitude data messages are used to transfer spacecraft attitude information between cooperating entities. They can be used for preflight planning and tracking, tracking and attitude operations, attitude propagations and predictions.

Orbit data messages are used to transfer spacecraft orbit information between cooperating entities. They can be used for preflight planning and tracking, scheduling tracking support, orbit propagation, orbit reconstruction, collision avoidance analysis, and maneuver planning and assessment.

Tracking Data Messages are used to exchange spacecraft tracking data between space agencies, between centers within a space agency, between systems within a center, and other types of interfaces. Some examples of the data within a Tracking Data Message are uplink frequencies, range, Doppler, antenna angles, clock parameters, and meteorological data.

Note:

- For Navigation Data messages subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages.
- For Navigation Data message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.1 Attitude Parameter Message

The content of the C2MS Attitude Parameter Message contains a CCSDS APM (or another format) and is used for transferring APMs. Attitude information within the Attitude Parameter Message contains the state of a single object at a specified epoch. The APM provides information for use in modeling finite maneuvers. Solar radiation pressure can be modeled when accompanied with an Orbit Parameter Message.

The NDM-TYPE field of the Attitude Parameter Message Contents is set to CCSDS-APM or APM for non-CCSDS formats. The NDM-SUBTYPE is a string that describes one of the many types of APMs. The mode of the APM can be either real-time (RT), replay (RPY), simulation (SIM), or test (TEST). The ACTIVITY-ID is used in conjunction with the mission and vehicle to identify the specific activity that is occurring during the mission. The FORMAT field indicates whether the APM is in XML, keyword value notation, raw, or binary format.

8.14.1.1 Attitude Parameter Message Subjects

For Attitude Parameter Message subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages

8.14.1.2 Attitude Parameter Message Header

For Attitude Parameter Message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.1.3 Attitude Parameter Message Contents

The following figure shows a UML object diagram of the Attitude Parameter Message with its required and optional fields.

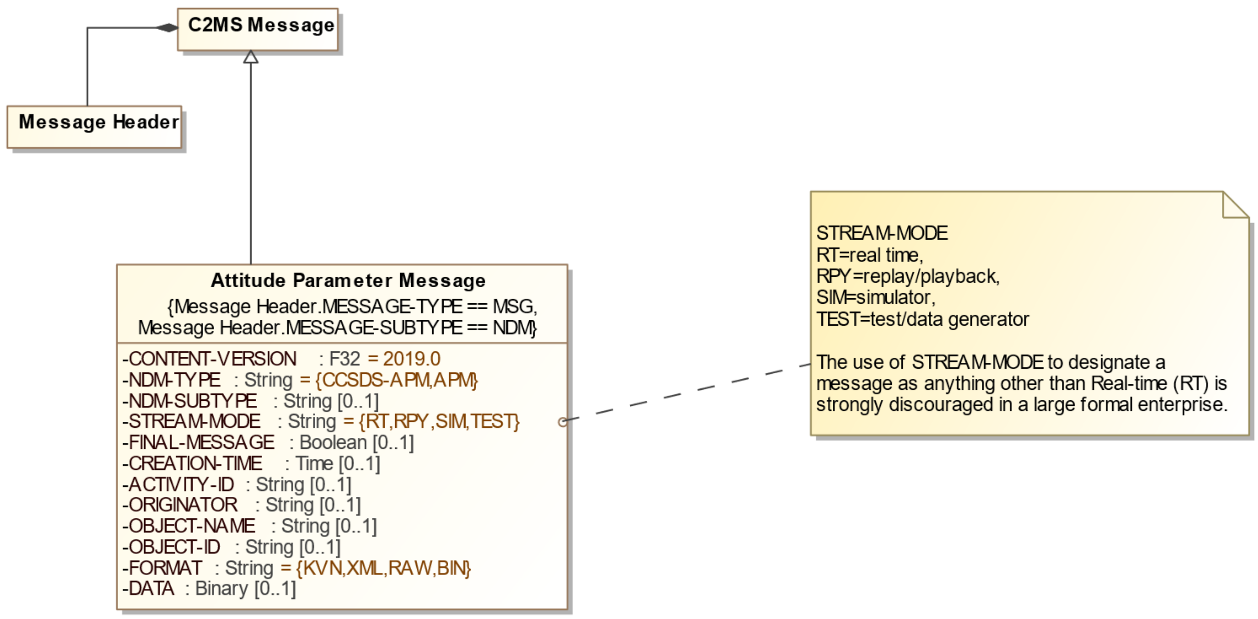

C2MS Message
Message Header
Attitude Parameter Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == NDM}
-CONTENT-VERSION : F32 = 2019.0
-NDM-TYPE : String = {CCSDS-APM,APM}
-NDM-SUBTYPE : String [0..1]
-STREAM-MODE : String = {RT,RPY,SIM,TEST}
-FINAL-MESSAGE : Boolean [0..1]
-CREATION-TIME : Time [0..1]
-ACTIVITY-ID : String [0..1]
-ORIGINATOR : String [0..1]
-OBJECT-NAME : String [0..1]
-OBJECT-ID : String [0..1]
-FORMAT : String = {KVN,XML,RAW,BIN}
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-37. Attitude Parameter Message Diagram

The following table describes additional field names, values, and notes for the Attitude Parameter Message.

Table 8-175. Attitude Parameter Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2019	Version number for this message content description
NDM-TYPE	String	R	[CCSDS-APM, APM]	Message contains an Attitude Parameter Message in CCSDS or another format
NDM-SUBTYPE	String	O	[miscellaneous]	Descriptor of the type / kind of the contents of the APM. E.g. Attitude state info, Euler angle rates, or spacecraft parameters
STREAM-MODE	String	R	**Value** | **Description** RT | Real-time RPY | Replay SIM | Simulator TEST | Test/Data Generator	Identifies the mode of the stream of messages as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
FINAL-MESSAGE	Boolean	O		When true (and known, especially for replay data), indicates the last message in the stream.
CREATION-TIME	Time	O		Time the Navigation Data Message was created.
ACTIVITY-ID	String	O		Specifies the activity occurring within the mission
ORIGINATOR	String	O		Creating agency. E.g. GSFC-FDF, GSOC, JPL, JAXA etc.
OBJECT-NAME	String	O		Spacecraft name
OBJECT-ID	String	O		Spacecraft identifier
FORMAT	String	R	[KVN, XML, RAW, BIN]	Format of the DATA field KVN: Keyword = Value Notation, XML: eXtensible Markup Language, Raw, or Bin (binary)
DATA	Binary	O		Attitude Parameter Message contents – see Table 8-174 for reference to CCSDS format

8.14.2 Attitude Ephemeris Message

The content of the C2MS Attitude Ephemeris Message is a CCSDS AEM (or another format) and is used for transferring AEMs. Attitude information within the Attitude Parameter Message contains the state of a single object at multiple epochs within a specified time range. The AEM provides information for use in dynamic modeling of various kinds of torques such as solar pressure, magnetics, and atmospheric torques.

The NDM-TYPE field of the Attitude Ephemeris Message Contents is set to CCSDS-AEM or AEM for non-CCSDS formats. The NDM-SUBTYPE is a string that describes one of the many types of AEMs. The mode of the AEM can be either real-time (RT), replay (RPY), simulation (SIM), or test (TEST). The ACTIVITY-ID is used in conjunction with the mission and vehicle to identify the specific activity that is occurring during the mission. The FORMAT field indicates whether the AEM is in XML, keyword value notation, raw, or binary format.

8.14.2.1 Attitude Ephemeris Message Subjects

For Attitude Ephemeris Message subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages.

8.14.2.2 Attitude Ephemeris Message Header

For Attitude Ephemeris Message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.2.3 Attitude Ephemeris Message Contents

The following figure shows a UML object diagram of the Attitude Ephemeris Message with its required and optional fields.

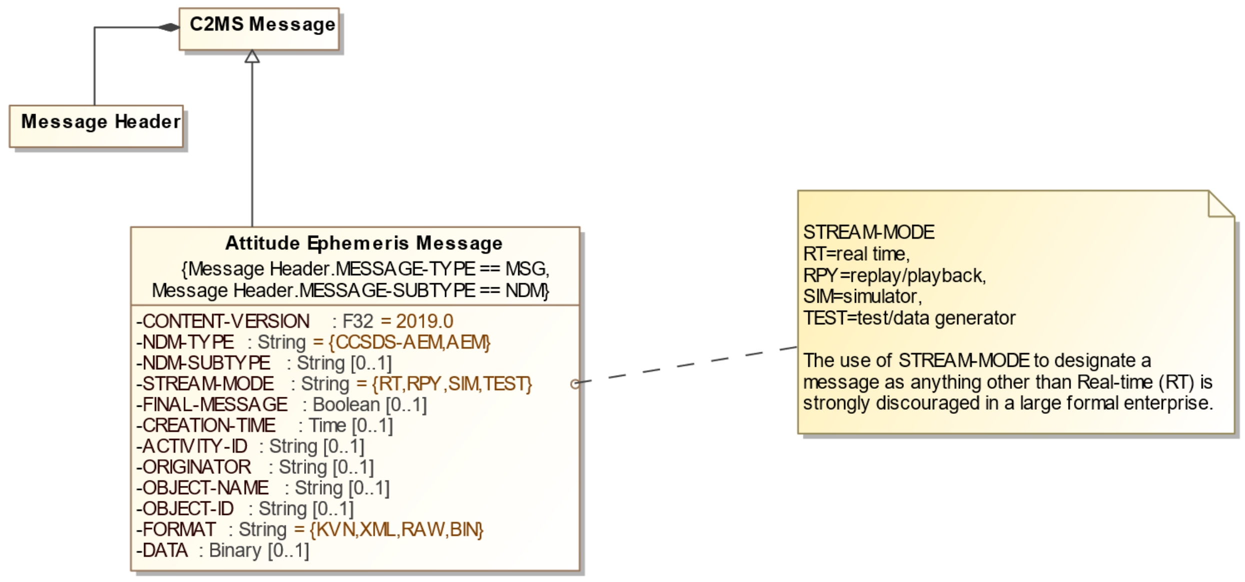

C2MS Message
Message Header
Attitude Ephemeris Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == NDM}
-CONTENT-VERSION : F32 = 2019.0
-NDM-TYPE : String = {CCSDS-AEM,AEM}
-NDM-SUBTYPE : String [0..1]
-STREAM-MODE : String = {RT,RPY,SIM,TEST}
-FINAL-MESSAGE : Boolean [0..1]
-CREATION-TIME : Time [0..1]
-ACTIVITY-ID : String [0..1]
-ORIGINATOR : String [0..1]
-OBJECT-NAME : String [0..1]
-OBJECT-ID : String [0..1]
-FORMAT : String = {KVN,XML,RAW,BIN}
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-38. Attitude Ephemeris Message Diagram

The following table describes additional field names, values, and notes for the Attitude Ephemeris Message.

Table 8-176. Attitude Ephemeris Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2019		Version number for this message content description
NDM-TYPE	String	R	[CCSDS-AEM, AEM]		Message contains an Attitude Ephemeris Message in CCSDS or another format
NDM-SUBTYPE	String	O	[miscellaneous]		Descriptor of the type / kind of the contents of the AEM. E.g. Quaternion values, spin data, and Euler elements
STREAM-MODE	String	R	**Value**	**Description**	Identifies the mode of the stream of messages as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a
			RT	Real-time	
			RPY	Replay	

Field Name	Type	Presence	Value		Description
			SIM	Simulator	message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
CREATION-TIME	Time	O			Time the Navigation Data Message was created.
ACTIVITY-ID	String	O			Specifies the activity occurring within the mission
ORIGINATOR	String	O			Creating agency. E.g. GSFC-FDF, GSOC, JPL, JAXA etc.
OBJECT-NAME	String	O			Spacecraft name
OBJECT-ID	String	O			Spacecraft identifier
FORMAT	String	R	[KVN, XML, RAW, BIN]		Format of the DATA field KVN: Keyword = Value Notation, XML: eXtensible Markup Language, Raw, or Bin (binary)
DATA	Binary	O			Attitude Ephemeris Message contents– see Table 8-174 for reference to CCSDS format

8.14.3 Orbit Parameter Message

The content of the C2MS Orbit Parameter Message contains a CCSDS OPM (or another format) and is used for transferring OPMs. Orbit information within the Orbit Parameter Message contains position and velocity information about a single object for a specific epoch. The OPM provides information for use in modeling maneuvers, atmospheric drag, and other predictive calculations.

The NDM-TYPE field of the Orbit Parameter Message Contents is set to CCSDS-OPM or OPM for non-CCSDS formats. The NDM-SUBTYPE is a string that describes one of the many types of OPMs. The mode of the OPM can be either real-time (RT), replay (RPY), simulation (SIM), or test (TEST). The ACTIVITY-ID is used in conjunction with the mission and vehicle to identify the specific activity that is occurring during the mission. The FORMAT field indicates whether the OPM is in XML, keyword value notation, and raw, or binary format.

8.14.3.1 Orbit Parameter Message Subjects

For Orbit Parameter Message subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages.

8.14.3.2 Orbit Parameter Message Header

For Orbit Parameter Message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.3.3 Orbit Parameter Message Contents

The following figure shows a UML object diagram of the Orbit Parameter Message with its required and optional fields.

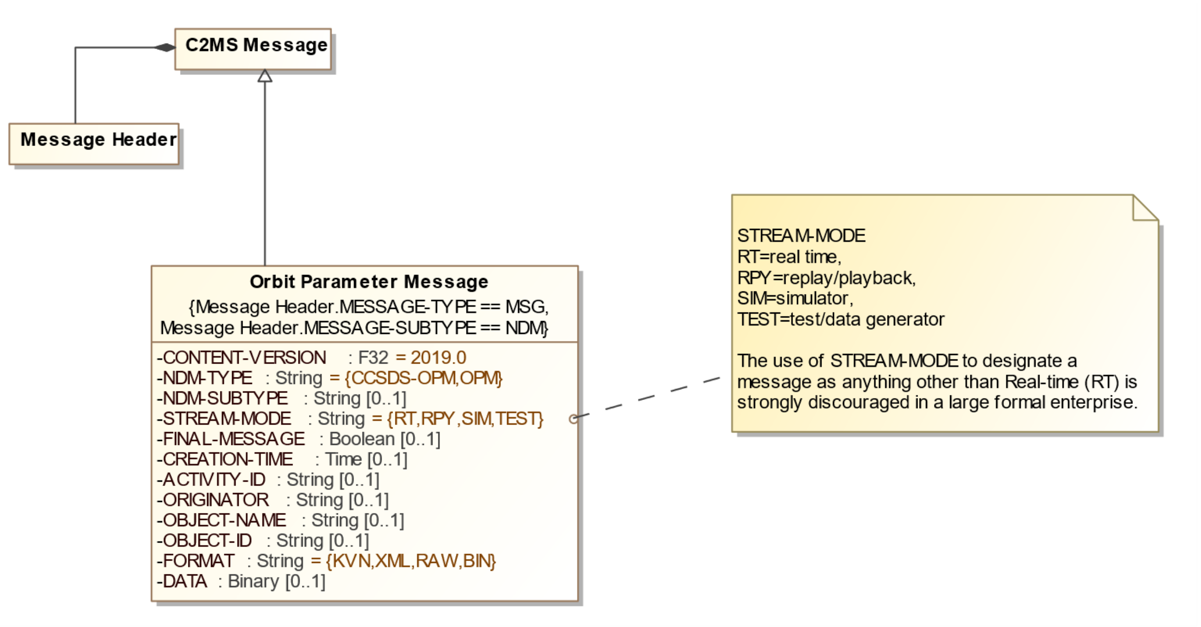

C2MS Message
Message Header
Orbit Parameter Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == NDM}
-CONTENT-VERSION : F32 = 2019.0
-NDM-TYPE : String = {CCSDS-OPM,OPM}
-NDM-SUBTYPE : String [0..1]
-STREAM-MODE : String = {RT,RPY,SIM,TEST}
-FINAL-MESSAGE : Boolean [0..1]
-CREATION-TIME : Time [0..1]
-ACTIVITY-ID : String [0..1]
-ORIGINATOR : String [0..1]
-OBJECT-NAME : String [0..1]
-OBJECT-ID : String [0..1]
-FORMAT : String = {KVN,XML,RAW,BIN}
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-39. Orbit Parameter Message Diagram

The following table describes additional field names, values, and notes for the Orbit Parameter Message.

Table 8-177. Orbit Parameter Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2019	Version number for this message content description
NDM-TYPE	String	R	[CCSDS-OPM, OPM]	Message contains an Orbit Parameter Message in CCSDS or another format
NDM-SUBTYPE	String	O	[miscellaneous]	Descriptor of the type / kind of the contents of the OPM. E.g., state vector, Keplerian elements, maneuvers, matrix
STREAM-MODE	String	R	**Value** / **Description**: RT – Real-time; RPY – Replay; SIM – Simulator; TEST – Test/Data Generator	Identifies the mode of the stream of messages as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
FINAL-MESSAGE	Boolean	O		When true (and known, especially for replay data), indicates the last message in the stream.
CREATION-TIME	Time	O		Time the Navigation Data Message was created.
ACTIVITY-ID	String	O		Specifies the activity occurring within the mission
ORIGINATOR	String	O		Creating agency. E.g. GSFC-FDF, GSOC, JPL, JAXA etc.
OBJECT-NAME	String	O		Spacecraft name
OBJECT-ID	String	O		Spacecraft identifier

Field Name	Type	Presence	Value	Description
FORMAT	String	R	[KVN, XML, RAW, BIN]	Format of the DATA field KVN: Keyword = Value Notation, XML: eXtensible Markup Language, Raw, or Bin (binary)
DATA	Binary	O		Orbit Parameter Message contents–see Table 8-174 for reference to CCSDS format

8.14.4 Orbit Mean-Elements Message

Orbital Mean-Elements data messages are used to transfer spacecraft orbital characteristics between cooperating entities. The information can be used to determine future contact parameters between ground and space assets.

The content of the C2MS Orbital Mean-Elements Message contains a CCSDS OMM (or another format) and is used for transferring OMMs. Orbital state information within the Orbital Mean-Elements Message can contain mean Keplerian elements, spacecraft parameters, and two-line element sets of a single object for a specific epoch. The OMM provides information for use in directing antennas and planning contacts with satellites.

The NDM-TYPE field of the Orbit Parameter Message Contents is set to CCSDS-OMM or OMM for non-CCSDS formats. The NDM-SUBTYPE is a string that describes one of the types of OMMs. The mode of the OMM can be either real-time (RT), replay (RPY), simulation (SIM), or test (TEST). The ACTIVITY-ID is used in conjunction with the mission and vehicle to identify the specific activity that is occurring during the mission. The FORMAT field indicates whether the OMM is in XML or keyword value notation.

8.14.4.1 Orbit Mean-Elements Message Subjects

For Orbit Mean-Elements Message subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages.

8.14.4.2 Orbit Mean-Elements Message Header

For Orbit Mean-Elements Message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.4.3 Orbit Mean-Elements Message Contents

The following figure shows a UML object diagram of the Orbit Mean-Elements Message with its required and optional fields.

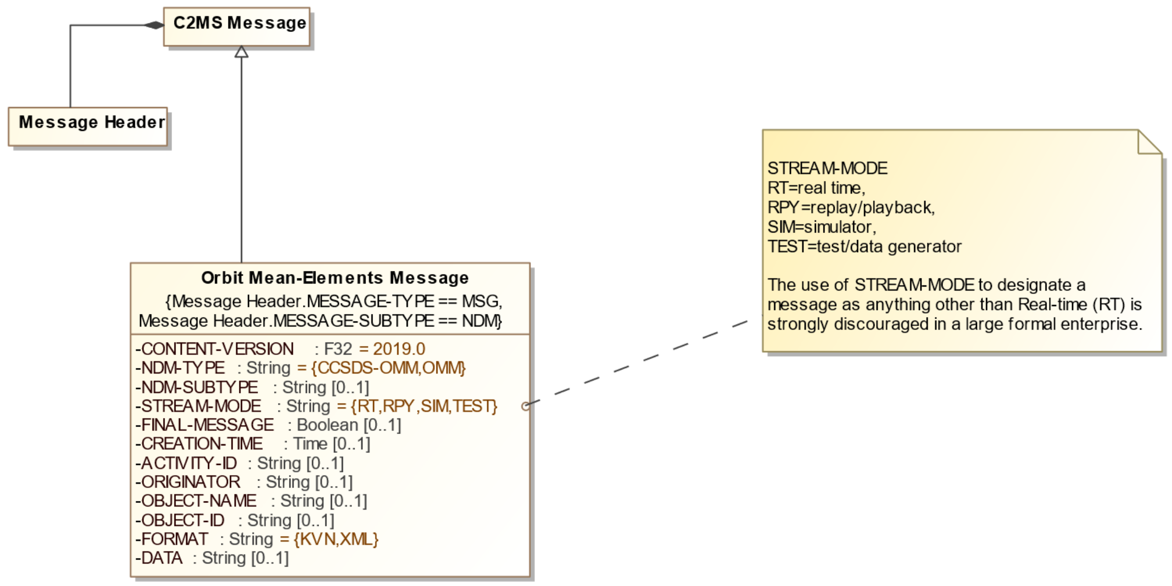

C2MS Message
Message Header
Orbit Mean-Elements Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == NDM}
-CONTENT-VERSION : F32 = 2019.0
-NDM-TYPE : String = {CCSDS-OMM,OMM}
-NDM-SUBTYPE : String [0..1]
-STREAM-MODE : String = {RT,RPY,SIM,TEST}
-FINAL-MESSAGE : Boolean [0..1]
-CREATION-TIME : Time [0..1]
-ACTIVITY-ID : String [0..1]
-ORIGINATOR : String [0..1]
-OBJECT-NAME : String [0..1]
-OBJECT-ID : String [0..1]
-FORMAT : String = {KVN,XML}
-DATA : String [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-40. Orbit Mean-Elements Message Diagram

The following table describes additional field names, values, and notes for the Orbit Mean-Elements Message.

Table 8-178. Orbital Mean-Elements Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2019		Version number for this message content description
NDM-TYPE	String	R	[CCSDS-OMM, OMM]		Message contains an Orbit Mean-Elements Message in CCSDS or another format
NDM-SUBTYPE	String	O	[miscellaneous]		Descriptor of the type / kind of the contents of the OMM. E.g. two-line element set, Keplerian elements, covariance matrix, or user defined
STREAM-MODE	String	R	**Value**	**Description**	Identifies the mode of the stream of messages as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
CREATION-TIME	Time	O			Time the Navigation Data Message was created.
ACTIVITY-ID	String	O			Specifies the activity occurring within the mission
ORIGINATOR	String	O			Creating agency. E.g. GSFC-FDF, GSOC, JPL, JAXA etc.
OBJECT-NAME	String	O			Spacecraft name

Field Name	Type	Presence	Value	Description
OBJECT-ID	String	O		Spacecraft identifier
FORMAT	String	R	[KVN, XML]	Format of the DATA field KVN: Keyword = Value Notation XML: eXtensible Markup Language
DATA	String	O		Orbit Mean-Elements Message contents– see Table 8-174 for reference to CCSDS format

8.14.5 Orbit Ephemeris Message

The content of the C2MS Orbit Ephemeris Message contains a CCSDS (or other format) OEM and is used for transferring OEMs. Orbit information within the Orbit Ephemeris Message contains position and velocity information about a single object at multiple epochs within a specific time range. The OEM provides information for use in modeling maneuvers, representing orbits, and other predictive calculations.

The NDM-TYPE field of the Orbit Ephemeris Message Contents is set to CCSDS-OEM or OEM (for non-CCSDS formats). The NDM-SUBTYPE is a string that describes one of the many types of OEMs. The mode of the OEM can be either real-time (RT), replay (RPY), simulation (SIM), or test (TEST). The ACTIVITY-ID is used in conjunction with the mission and vehicle to identify the specific activity that is occurring during the mission. The FORMAT field indicates whether the OEM is in XML or keyword value notation.

8.14.5.1 Orbit Ephemeris Message Subjects

For Orbit Ephemeris Message subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages.

8.14.5.2 Orbit Ephemeris Message Header

For Orbit Ephemeris Message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.5.3 Orbit Ephemeris Message Contents

The following figure shows a UML object diagram of the Orbit Ephemeris Message with its required and optional fields.

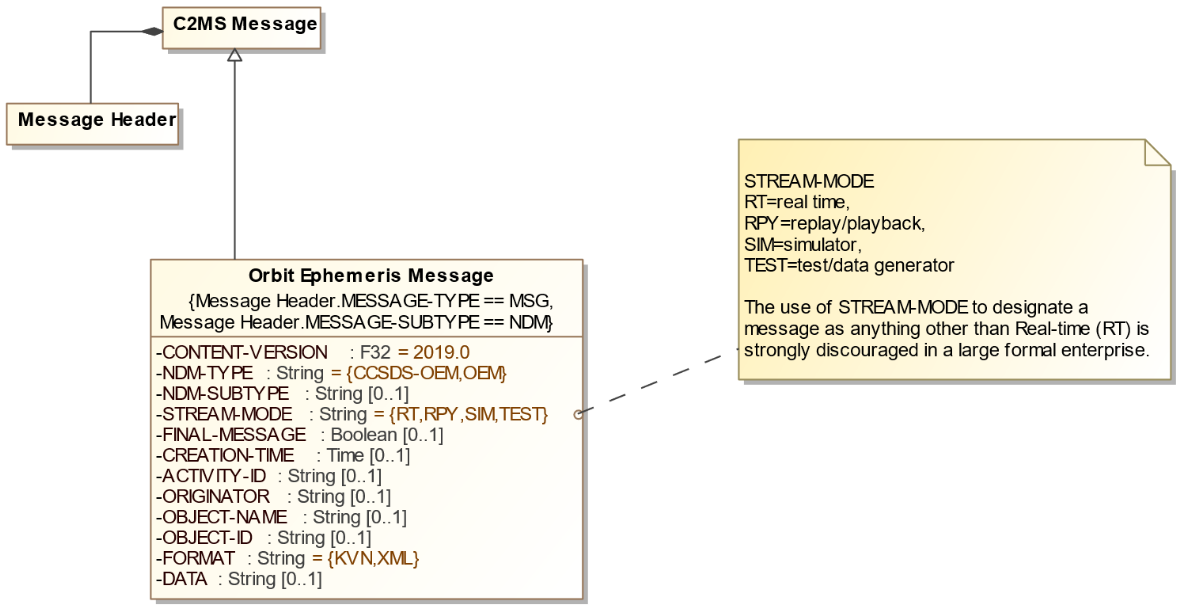

C2MS Message
Message Header
Orbit Ephemeris Message
{Message Header.MESSAGE-TYPE == MSG, Message Header.MESSAGE-SUBTYPE == NDM}
-CONTENT-VERSION : F32 = 2019.0
-NDM-TYPE : String = {CCSDS-OEM,OEM}
-NDM-SUBTYPE : String [0..1]
-STREAM-MODE : String = {RT,RPY,SIM,TEST}
-FINAL-MESSAGE : Boolean [0..1]
-CREATION-TIME : Time [0..1]
-ACTIVITY-ID : String [0..1]
-ORIGINATOR : String [0..1]
-OBJECT-NAME : String [0..1]
-OBJECT-ID : String [0..1]
-FORMAT : String = {KVN,XML}
-DATA : String [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-41. Orbit Ephemeris Message Diagram

The following table describes additional field names, values, and notes for the Orbit Ephemeris Message.

Table 8-179. Orbit Ephemeris Message Additional Information

Field Name	Type	Presence	Value		Description
CONTENT-VERSION	F32	R	2019		Version number for this message content description
NDM-TYPE	String	R	[CCSDS-OEM, OEM]		Message contains an Orbit Ephemeris Message in CCSDS or another format
NDM-SUBTYPE	String	O	[Miscellaneous]		Descriptor of the type / kind of the contents of the OEM. E.g. state vector, Keplerian elements, maneuvers, matrix
STREAM-MODE	String	R	**Value**	**Description**	Identifies the mode of the stream of messages as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
			RT	Real-time	
			RPY	Replay	
			SIM	Simulator	
			TEST	Test/Data Generator	
FINAL-MESSAGE	Boolean	O			When true (and known, especially for replay data), indicates the last message in the stream.
CREATION-TIME	Time	O			Time the Navigation Data Message was created.
ACTIVITY-ID	String	O			Specifies the activity occurring within the mission
ORIGINATOR	String	O			Creating agency. E.g. GSFC-FDF, GSOC, JPL, JAXA etc.
OBJECT-NAME	String	O			Spacecraft name

Field Name	Type	Presence	Value	Description
OBJECT-ID	String	O		Spacecraft identifier
FORMAT	String	R	[KVN, XML]	Format of the DATA field KVN: Keyword = Value Notation XML: eXtensible Markup Language
DATA	String	O		Orbit Ephemeris Message contents– see Table 8-174 for reference to CCSDS format

8.14.6 Tracking Data Message

The content of the C2MS Tracking Data Message contains a CCSDS (or other type of) TDM and is used for transferring TDMs.

The NDM-TYPE field of the Tracking Data Message Contents is set to CCSDS-TDM or TDM (for non-CCSDS formats). The NDM-SUBTYPE is a string that describes one of the many types of TDMs. Some examples are Doppler and range radiometrics in a variety of tracking modes, very-long-baseline interferometry (VLBI) data, antenna pointing angles, etc.

The ACTIVITY-ID is used in conjunction with the mission and vehicle to identify the specific activity that is occurring during the mission. The FORMAT field indicates whether the TDM is in XML, keyword value notation, raw, or binary format.

The C2MS Tracking Data Message Contents can optionally contain information about the origin, time, and quality of the data.

8.14.6.1 Tracking Data Message Subjects

For Tracking Data Message subjects, see Section 8.14.7.1 Subjects for Navigation Data Messages

8.14.6.2 Tracking Data Message Header

For Tracking Data Message header, see Section 8.14.7.2 Header for Navigation Data Messages.

8.14.6.3 Tracking Data Message Contents

The following figure shows a UML object diagram of the Tracking Data Message with its required and optional fields.

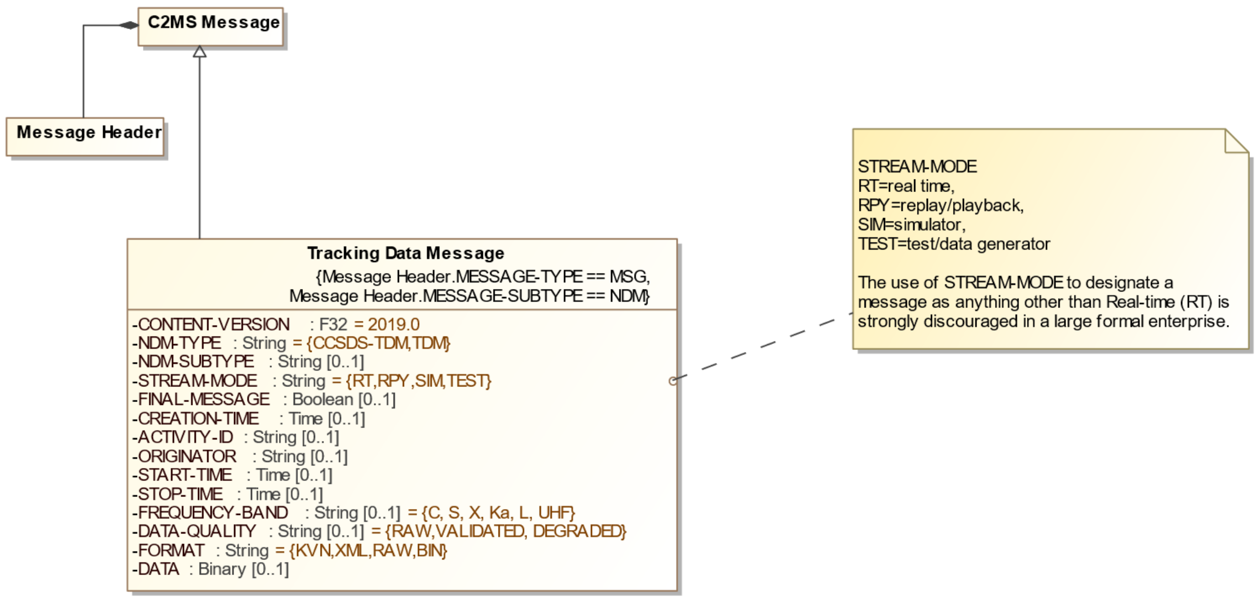

C2MS Message
Message Header
Tracking Data Message
{Message Header.MESSAGE-TYPE == MSG,
Message Header.MESSAGE-SUBTYPE == NDM}
-CONTENT-VERSION : F32 = 2019.0
-NDM-TYPE : String = {CCSDS-TDM,TDM}
-NDM-SUBTYPE : String [0..1]
-STREAM-MODE : String = {RT,RPY,SIM,TEST}
-FINAL-MESSAGE : Boolean [0..1]
-CREATION-TIME : Time [0..1]
-ACTIVITY-ID : String [0..1]
-ORIGINATOR : String [0..1]
-START-TIME : Time [0..1]
-STOP-TIME : Time [0..1]
-FREQUENCY-BAND : String [0..1] = {C, S, X, Ka, L, UHF}
-DATA-QUALITY : String [0..1] = {RAW,VALIDATED, DEGRADED}
-FORMAT : String = {KVN,XML,RAW,BIN}
-DATA : Binary [0..1]
STREAM-MODE
RT=real time,
RPY=replay/playback,
SIM=simulator,
TEST=test/data generator
The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise.

Figure 8-42. Tracking Data Message Diagram

The following table describes additional field names, values, and notes for the Tracking Data Message.

Table 8-180. Tracking Data Message Additional Information

Field Name	Type	Presence	Value	Description
CONTENT-VERSION	F32	R	2019	Version number for this message content description
NDM-TYPE	String	R	[CCSDS-TDM, TDM]	Message contains a Tracking Data Message in CCSDS or another format
NDM-SUBTYPE	String	O	[miscellaneous]	Descriptor of the type / kind of the contents of the TDM. E.g. Doppler, angle, range, one-way.
STREAM-MODE	String	R	**Value** / **Description** RT / Real-time RPY / Replay SIM / Simulator TEST / Test/Data Generator	Identifies the mode of the stream of messages as either Real-time, Replay, Simulator, or Test. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
FINAL-MESSAGE	Boolean	O		When true (and known, especially for replay data), indicates the last message in the stream.
CREATION-TIME	Time	O		Time the Navigation Data Message was created.
ACTIVITY-ID	String	O		Specifies the activity occurring within the mission

Field Name	Type	Presence	Value	Description
ORIGINATOR	String	O		Creating agency. E.g. GSFC-FDF, GSOC, JPL, JAXA etc.
START-TIME	Time	O		Start time of the tracking data
STOP-TIME	Time	O		Stop time of the tracking data
FREQUENCY-BAND	String	O	[C, S, X, Ka, L, UHF]	Frequency band for transmitted signals

DATA-QUALITY	String	O	[RAW,VALIDATED, DEGRADED]	Raw = no quality check
				Validated = checked and passed
				Degraded = checked with quality issues.

FORMAT	String	R	[KVN, XML, RAW, BIN]	Format of the DATA field KVN: Keyword = Value Notation, XML: eXtensible Markup Language, Raw, or Bin (binary)
DATA	Binary	O		Track Data Message contents– see Table 8-174 for reference to CCSDS format

8.14.7 SUBJECTS and HEADER for Navigation Data Messages

8.14.7.1 Subjects for Navigation Data Messages

Table 8-181. Navigation Data Message Subject Naming

	Subject Standard	Domain Elements		Mission Elements			Message Elements		*Miscellaneous Elements*					
Subject Element	**Speci-fication**	**DOMAIN1**	**DOMAIN2**	**MISSION**	**CONST**	**SAT**	**TYP**	**SUBTYP**	***ME1***	***ME2***	***ME3***	***ME4***	***ME5***	***ME6***
Subject Content	C2MS	[domain 1 – system specific]	[domain 2 – system specific]	[mission]	[constell ation]	[sat]	MSG	NDM	[COMPONE NT]	Stream-mode	NDM-TYPE	NDM-SUBTYPE (or mission specific)	Activity ID	
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	NDM	SATSIM	SIM	CCSDS-TDM			
Example for Publisher / Sender	C2MS	DOM1	DOM2	MSSN	CNS1	SAT1	MSG	NDM	FDF	RT	CCSDS-OPM			
Example for Subscriber / Receiver	C2MS	DOM1	DOM2	*	*	SAT1	MSG	NDM	*	RT	*			

Table 8-182. Properties of the *Miscellaneous Elements* for the Navigation Data Message

Miscellaneous Element	Required / Optional	Description	Field in Msg, if applicable
ME1	Required	Component name of Publisher	COMPONENT from header
ME2	Required	Identifies stream as real-time, playback, simulator, or test.	STREAM-MODE. The use of STREAM-MODE to designate a message as anything other than Real-time (RT) is strongly discouraged in a large formal enterprise. See STREAM-MODE Usage Caution in Section 6.3.7.
ME3	Required	Type of Navigation Data Message	NDM-TYPE
ME4	Optional	Subtype of Navigation Data Message	NDM-SUBTYPE; or mission specific
ME5	Optional	Activity ID	ACTIVITY-ID
ME6			

Examples

Example for Publisher / Sender of Navigation Data Messages:

```
C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.NDM.CENTER-FACILITY.RT.TDM

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.NDM.CENTER-FACILITY.RT.CCSDS-TDM

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.NDM.SATSIM.SIM.CCSDS-OPM

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.NDM.SATSIM.SIM.CCSDS-OEM
```

Example for Subscriber / Receiver of Navigation Data Messages:

```
C2MS.*.*.MSSN.*.*.MSG.NDM.*.SIM.CCSDS-OPM.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.NDM.*.RT.CCSDS-TDM.>

C2MS.DOM1.DOM2.MSSN.CNS1.SAT1.MSG.NDM.ANTENNA5.RT.CCSDS-OEM.EPHEM.ACTY-
ID-123
```

8.14.7.2 Header for Navigation Data Messages

The abbreviated table below shows the required values of the MESSAGE-TYPE and MESSAGE-SUBTYPE fields for the header of Navigation Data Messages.

Table 8-183. Header Field Values for Navigation Data Messages

Header Field Name	Value
MESSAGE-TYPE	MSG
MESSAGE-SUBTYPE	NDM

Appendix A – Categorization

Component Categories

Table A-1 organizes the software standard into broad classes of functionality and further delineates the classes into subclasses.

Table A-1. Software Class and Subclass Categories

CLASS	Class Abbr.	Subclass	Subclass Abbr.
Archive and Assessment	AAA	Assessment Archive Plotting, Trending and Analysis	AST ARC PTA
Automation	AUTO	Paging Rule-Action	PAGE RULE
Flight Dynamics	FD	Orbit Determination Navigation and Control Ephemeris	OD NAC EPH
Modeling, Simulation, and Front-End Processors	MAS	Analysis Front End Processor Simulator	ANL FEP SIM
Planning and Scheduling	PAS	Maneuver Planning Scheduling	MAN SCH
Script Control	SCRIPT		
Security	SEC	Verify Encryption	VER CRYPT
System	SYS	Operating Systems COTS GOTS Monitor Configuration Control and Management	OS COTS GOTS MON CFG
Telemetry and Command	TAC	Telemetry Command	TLM CMD

(Not a complete listing)

Product Categories

Table A-2 uses the same Class categories and Class abbreviations as seen in Table A-1 to break down and classify the products in a hierarchical manner.

Table A-2. Product Categories

Product Types (*ME3*)	Abbr.	Product Subtype (*ME4*)	Abbr.	Product Subtype2 (*ME5*)	Abbr.	Product Subtype3 (*ME6*)	Abbr.
Archive and Assessment	AAA	Data	DATA				
		Message	MSG				
		Plot/Graph	PLOT				
		Report	RPT				
		Statistics	STAT				
Automation	AUTO	Decision Making	DM				
		Expert	EXP				
		Paging	PAGE				
Flight Dynamics	FD	Attitude	ATT				
		Environmental/Celestial	EC				
		Instrument	INST				
		Maneuver	MAN				
		Memory	MEM				
Flight Dynamics	FD	Orbit	ORBIT	Ephemeris	EPHEM	Binary	BIN
						FreeFlyer	FF
						CCSDS Orbit Ephemeris Message	OEM
						Satellite Tool Kit	STK
						Special Perturbations	SP
Flight Dynamics	FD	Orbit	ORBIT	State	STATE	Extended Precision Vector	EPV
						GPS Navigation Data	GPSNAV
						Improved Interrange Vector	IIRV
						CCSDS Orbit Parameter Message	OPM
						Orbital Parameter Reports	OPR
						Two Line Mean Elements	TLE
						Vehicle Attitude	VEHATT
Flight Dynamics	FD	Orbit	ORBIT	Station Contact	STA	Az/El Tables	AZEL

Product Types (*ME3*)	Abbr.	Product Subtype (*ME4*)	Abbr.	Product Subtype2 (*ME5*)	Abbr.	Product Subtype3 (*ME6*)	Abbr.
						Ground Site Location	GSL
						Line Summary	LS
						Predicted Site Acquisition Table	PSAT
						Site Views	SV
						STDN Summary Predictions	STDNSP
						Tracking Data	TD
						Vehicle Visibility Check	VVC
Flight Dynamics	FD	Orbit	ORBIT	Instrument	INST	High Gain Antenna Predictions	HGAP
						Science Field of View Predictions	SFVP
						Sensor Interference Predictions	SIP
						User Antenna View	UAV
Flight Dynamics	FD	Orbit	ORBIT	Orbit/Object Relationships	OOR	Orbital Events	EV
						Shadow Times	ST
						Solar Beta Angle	SBA
						Sun/Earth Relationships	SER
Modeling and Simulation	MAS	Data Files	DATA				
		Data Streams	STREAM				
		Messages	MSG				
Planning and Scheduling	PAS	Schedule	SCH	Activity Schedule	ACT		
				Contact Schedule	CON	Air Force Satellite Control Network	AFSCN
						Deep Space Network	DSN
						Ground Network	GN
						Space Network	SN
						Universal Space Network	USN
Planning and Scheduling	PAS	Spacecraft	SC	Command	CMD	Command Sequence File	CSF

Product Types (*ME3*)	Abbr.	Product Subtype (*ME4*)	Abbr.	Product Subtype2 (*ME5*)	Abbr.	Product Subtype3 (*ME6*)	Abbr.
						Block Commands	BC
		Unmanned Aerial Vehicles	UAV				
Scripting Control	SCRIPT						
Telemetry and Command	TAC	Data Values	DATA	Selected	SET	ID of selected subset	nnn
Telemetry and Command	TAC	Level-0	RAW				
Telemetry and Command	TAC	Memory Dumps	MEM				
Telemetry and Command	TAC	Products Derived from a pass/contact	PASS	Pass Description	PD		
	TAC			Frames Lost	FL		
				Pass Summary Report	PSR		
		Spacecraft Tables	TBL				