Overlap Assessment between OMG C2MS and CCSDS MO

Revision 1.0, 2018-06-22

Prepared by Stefan Gärtner, DLR,
on behalf of CCSDS SM&C Working Group members of CNES, DLR and ESA

Contents

1. Introduction

This document assesses possible overlaps between the proposed OMG C2MS standard [1] and the set of published CCSDS MO standards [2], [3], [4], [5], [6], [7] and proposed CCSDS MO standards [8], [9]. Overlaps are identified on several levels of abstraction – overall scope of the standards, functionality defined in either of the standards, down to overlap in technical details. Notable differences and caveats are pointed out where necessary.

Although this document tries to explain overlaps of C2MS and MO in an understandable way, it is out of scope of this document to provide an introduction to either of the standards. In order to gain more in-depth knowledge reading of [1] and [2] is recommended.

Throughout the document abbreviations C2MS and MCMS for [1] will be used interchangeably.

2. References

[1] OMG, Proposed Mission Control Message Specification (MCMS) RFC.

[2] CCSDS, 520.0-G-3 Mission Operations Services Concept, 2010.

[3] CCSDS, 521.0-B-2 Mission Operations Message Abstraction Layer, 2013.

[4] CCSDS, 521.1-B-1 Mission Operations Common Object Model, 2014.

[5] CCSDS, 522.1-B-1 Mission Operations Monitor & Control Services, 2017.

[6] CCSDS, 524.1-B-1 Mission Operations - MAL Space Packet Transport Binding and Binary Encoding, 2015.

[7] CCSDS, 524.2-B-1 Mission Operations - Message Abstraction Layer Binding to TCP/IP Transport and Split Binary Encoding, 2017.

[8] CCSDS, 522.0 Mission Operations Common Services.

[9] CCSDS, 522.2 Mission Operations Mission Data Product Distribution Services.

[10] CCSDS, 523.1-M-0 Mission Operations Message Abstraction Layer - Java API, 2013.

[11] CCSDS, 523.2 Mission Operations - C++ API.

[12] CCSDS, 524.3 Mission Operations - Message Abstraction Layer Binding to HTTP Transport and XML Encoding.

[13] CCSDS, 524.4 Mission Operations - Message Abstraction Layer Binding to ZeroMQ Message Transport Protocol.

3. Overlap regarding scope

C2MS objectives and benefits [1, p. xiv]:

> *"The objective of this MCMS (Mission 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."*

Scope of C2MS [1, p. 1]:

> *"This document, the Mission Control Message Specification (MCMS), 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."

MO objectives and benefits [2, p. 2–3]:

> *"Standardisation of a Mission Operations Service Framework offers a number of potential benefits for the development, deployment and maintenance of mission operations infrastructure:*
>
> - *increased interoperability between agencies, at the level of spacecraft, payloads, or ground-segment infrastructure components;*
> - *standardisation of infrastructure interfaces, even within agencies, leading to re-use between missions and the ability to establish common multi-mission infrastructure;*
> - *standardisation of operational interfaces for spacecraft from different manufacturers;*
>
> *[…]"*

Scope of MO [2, p. 2–5]:

> *"The Mission Operations Service Framework is concerned with end-to-end interaction between mission operations application software, wherever it may reside within the space system."*

And, more specifically, scope of MO MAL [2, p. 2–12]:

> *"The Message Abstraction Layer provides a standard messaging layer between the Consumer and Provider sides of the service framework. This, together with standardized bindings between the MO Service Framework layers, ensures that different implementations of the service framework can interoperate across the service interfaces, providing the underlying communications protocol stack is equivalent on both sides of the interface. The layer provides the following fundamental aspects:*
>
> - *A specification of the fundamental data types, enumerations and structures;*
> - *A definition of the rules for combining data types and structures;*
> - *Generic Messaging Interaction Patterns that define the allowed sequence of message exchange;*
> - *Fundamental concepts such as security and Quality of Service (QoS)."*

As can be seen from the citations and as will become clear from the following assessment of functional overlaps, the scope of both C2MS and MO have strong overlaps: Both standards strive to provide means to integrate typical satellite ground-segment applications common in satellite mission operations. Both standards approach this goal by standardizing the communication interfaces between systems, leveraging a Platform Independent Model (PIM) [1, p. 1], [2, p. 2–10]. The resulting benefits are the same in both cases: "plug-and-play" of components from different vendors or agencies, standardized external interfaces with possibility for internal use and re-use, and more that can be looked up in either of the standards.

It should be noted that CCSDS MO provides a whole framework consisting of service descriptions (e.g. [5], [8], [9]), a Platform Independent Model called MAL [3], an abstract API formulated using request and

indication primitives, concrete APIs for several implementation languages (e.g. [10], [11]), and concrete transport bindings (e.g. [6], [7], [12], [13]). The proposed C2MS standard provides message descriptions and a Platform Independent Model, but neither an abstract or concrete API nor concrete on-the-wire representations of the messages. The API is declared out of scope by C2MS, although at least one proprietary concrete API (GMSEC API) seems to exist. The on-the-wire message representation is not explicitly declared out of scope, but it can be inferred that this would be an implementation detail of the GMSEC API's message bus implementation. As such, C2MS does not provide any interoperability between two parties implementing the standard. Lack of an on-the-wire format prevents on-the-wire interoperability. Lack of an API prevents compatibility of any software components.

Comparison with MO thus makes most sense if C2MS is compared with MAL and its service descriptions. APIs and transport bindings are largely out of scope for C2MS and will not be compared to the MO counterparts.

C2MS currently restricts itself to ground-to-ground interfaces, whereas MO tries to be as deployment-independent as possible and explicitly addresses not only ground-to-ground, but also space-to-ground and space-to-space interfaces. A transport mapping to space-to-ground is available in [6]. The C2MS restriction mainly comes from the fact that a broker-based message bus is expected to be used. This technology is usually not available over the space link. However, due to the lack of a concrete on-the-wire format, this is not a fundamental restriction of C2MS. Still, employing C2MS over the space link would probably require a redesign of many message formats in order to efficiently use the limited band-width.

4. Overlap regarding functionality

The following table lists each message type defined by C2MS and identifies overlaps and equivalents with MO services. Generally, MO offers additional functionality on top of that (e.g. a MC::Statistics or MC::Check service, or more functionality for each of the mentioned services). As this document is primarily concerned with overlaps the added functionality is out of scope and only mentioned where deemed necessary. On the other hand, added functionality by C2MS over MO will be mentioned explicitly.

Usually, there is no direct one-to-one correspondence between a C2MS message and an MO service or operation, therefore the closest mapping or the idiomatic way of performing the same or similar function is mentioned. MO provides some rather generic services that are not only meant to be used by the end-user, but also by some other service. Most notably, these are the COM services (Event, Archive, Activity) [4]. Each service making use of those usually specifies usage specializations (e.g. concrete message body contents). C2MS has some similar message types, e.g. the Log message type that is used to convey different types of information belonging to different functional domains. C2MS specifies all uses of message types in one place, whereas MO groups together functional domains in services. This is the effect of the difference between the service-oriented paradigm of MO and the message-oriented paradigm of C2MS. However, the provided functionalities can be compared with each other nonetheless.

C2MS Message Subtype	C2MS Message Type	MO Service Equivalent
Log	Message	No standard Log service defined, but maps well to COM::Event service, especially because certain occurrence types have correspondence in certain standardized uses of the COM::Event service (e.g. telemetry limit violation -> MC::Check service, configuration change -> Common::Configuration service, ...) In case of command verification occurrence types the direct correspondence is contained in the MC::Action service. The MC::Action services defines how to use the COM::Activity service (which in turn makes use of the COM::Event service) for the same use case.
Archive Message Retrieval	Request	COM::Archive retrieve, query, and count operations. COM::Archive additionally defines standardized ways to store data. In C2MS each component decides on its own what it stores and has no way of storing data in a more central archive. With MO both use cases are covered. MO does not directly support archive extraction by expression, but each service instance is free to define additional query operations. This is not much different than C2MS where extraction by expression is only possible if additional information from the concrete component is provided (such as the storage format).
	Response	The archive message retrieval response data of C2MS is an opaque binary blob and it is not immediately clear how the requested messages are represented inside this blob. COM::Archive on the other hand always retrieves typed COM objects whose structure can be looked up using the provided information. For arbitrary product retrieval (on a higher level than COM objects) the Mission Data Product Distribution Services are provided by MO.
Directive	Request	No one-to-one correspondence with MO. Typically, requesting a service from an MO component means invoking a specific operation. Using the service definition one can say that each operation provides its own message type. Upon reception of this message type the service operation is performed and more messages are produced according to the interaction pattern of the operation. A generic Directive message is not needed because all possible service operations are well-defined and as such their messages, too. Still, if deemed necessary a service definition taking actions according to the parse result of a free-form text is trivial to specify. This approach relies on extra information, however, that is better captured in machine-readable service definition. Alternatively, for a more dynamic approach that roughly maps to the Directive message the MC::Action service can be used with DIRECTIVE-KEYWORD mapping to an ActionIdentity and DIRECTIVE-STRING to ActionInstanceDetails. ActionInstanceDetails may be much more complex than a simple String and allow a more structured Directive request that does not need to rely on String parsing.
	Response	Typically for an MO service request the response depends on the

		interaction pattern and is tailored to the concrete operation to be performed. Alternatively, if the dynamic MC::Action service is used, the submitAction operation provides the acceptance acknowledgment or error condition. Execution tracking is possible by subscribing to the events mandated by the COM::Activity service that is required to be used by the MC::Action service. No additional data can be returned by this operation.
Component-to-Component Transfer (C2CX)	Configuration Status Message	Solicited configuration status can be obtained by invoking COM::Configuration.getCurrent operation on the component in question. Unsolicited configuration status (the closest to a Configuration Status Message) is obtained by subscribing to ConfigurationSwitched COM events, which are created by the component in question on each configuration change. Group association is currently not provided by MO, however, considering the sparse documentation of C2MS on this topic, it seems feasible to map this to the Domain concept of MO.
	Control Message	Maps to COM::Configuration.activate. A pre-defined set of component modes or configurations can be selected or a parameterized mode or configuration, depending on the component to control. The controlled component is notified using a ConfigurationSwitch event on the COM::Event service.
	Device Message	No specific MO service defined. It maps to MC::Parameter.monitorValue or MC::Aggregation.monitorValue as device parameter reporting is very similar to telemetry reporting.
	Heartbeat Message	No specific MO service defined. It maps to MC::Parameter.monitorValue, if properly configured.
	Resource Message	No specific MO service defined. It maps to MC::Parameter.monitorValue or MC::Aggregation.monitorValue, if properly configured.
Real-Time Telemetry Data	Message for CCSDS Packet	No specific MO service defined. The idiomatic way is usage of MC::Aggregation.monitorValue containing typed parameters. Depending on the use case other alternatives exist, e.g. definition of a CCSDS packet service or delivery of CCSDS packets as MAL::Blob parameters using MC::Aggregation.
	Message for CCSDS Frame	No specific MO service defined. The idiomatic way is usage of MC::Aggregation.monitorValue containing typed parameters. Depending on the use case other alternatives exist, e.g. definition of a CCSDS frame service or delivery of CCSDS frames as MAL::Blob parameters using MC::Aggregation.
	Message for TDM Frame	No specific MO service defined. The idiomatic way is usage of MC::Aggregation.monitorValue containing typed parameters. Depending on the use case other alternatives exist, e.g. definition of a TDM frame service or delivery of TDM frames as MAL::Blob parameters using MC::Aggregation.
	Message for Processed Telemetry	No specific MO service defined. It would also be unclear how to do that because a parameter (or mnemonic) value might span more than one CCSDS frame. A more suitable level for a service of this kind

	Frame	would be on CCSDS packet level. In this case one can map to MC::Aggregation.monitorValue with predefined AggregationDefinitions that are generated per CCSDS packet using the spacecraft database. MC::Aggregation also provides operations to manage definition and generation of parameter aggregations, which is not provided by C2MS.
Replay Telemetry Data	Request, Response	These messages roughly map to MDPD::MDPD.requestProduct for historic product retrieval and MDPD::MDPD.monitorProduct for future product retrieval. It is also possible to directly query the COM::Archive service for stored AggregationValueInstance or ParameterValueInstance objects. In future, a Replay Session Management and Control service can provide the possibility to allow the same services and operations for telemetry data (i.e. MC::Parameter and MC::Aggregation) to be re-used in a replay session. Every MAL message always contains information about the session (any identifier) and session type (live, replay, or simulation).
Real-Time Mnemonic Value	Request, Response	These two messages map to MAL PUBSUB interaction pattern-specific REGISTER and DEREGISTER messages of MC::Parameter.monitorValue for continuous delivery. For "oneshot" requests they would either map to MC::Parameter.getValue or MC::Aggregation.getValue.
	Data Message	Maps to the PUBLISH/NOTIFY message of MC::Aggregation.monitorValue.
Archive Mnemonic Value	Request, Response, Data Message	The same mapping considerations as for Replay Telemetry Data apply.
Satellite Command	Request	Maps to MC::Action.submitAction operation.
	Response	Maps to events emitted by the COM::Activity service, whose use is mandated by the MC::Action.submitAction operation.
Product	Request, Response	Maps to Mission Data Product Distribution services, specifically to MDPD::MDPD.requestProduct operation.
	Product Message	If this message would be generated following a Request, Response exchange, it maps to the UPDATE messages of the MDPD::MDPD.requestProduct PROGRESS operation. If this message is used for unsolicited product distribution, it maps to MDPD::MDPDP.monitorProduct. In addition to inline or reference delivery, MDPD can also be used for push delivery, actively pushing the product to a third party.
Navigation Data	Attitude Parameter Message	No corresponding MO services defined yet, but planned in form of Navigation services.
	Attitude Ephemeris Message	
	Orbit Parameter Message	

Orbit Mean-Elements Message
Orbit Ephemeris Message
Tracking Data Message

5. Overlap regarding technical details

5.1. Use of Messages

As already mentioned C2MS employs a message-oriented paradigm, whereas MO uses a service-oriented paradigm. These paradigms mainly affect how functionality is presented in the documents and how functionality is grouped together. In fact, both standards mediate data exchange through the use of well-defined messages and their equally well-defined exchange patterns.

C2MS' use of messages is the main interface for the user due to lack of a standardized API. MO uses messages merely as a transport mechanism. Still, each message is well-defined. For interacting parties it is not discernible how messages were generated, i.e. by the RPC-like API of MAL or directly by an application. In this sense MAL can be used in the same way as C2MS.

Messages in both standards consist of a message header and a message body. C2MS messages additionally employ a subject name. The only technical differences that arise from the different paradigms are different message header fields in order to represent the grouping: In C2MS each message is defined by combination of fields "Type" and "Subtype". In MO "Type" would roughly map to "Interaction Type" and "Interaction Stage" and "Subtype" would decompose in "Area", "Area Version" and "Service Number".

Elements that go into a C2MS message subject name typically are represented as ordinary message header fields in MO messages. If a message broker-based MO transport binding is employed, some of these header fields would typically be used to construct a subject name. For example C2MS elements DOMAIN1, DOMAIN2, MISSION, CONST, SAT can be mapped to "Domain" and "Session" MAL header fields, ME1 would map to "URI from" or "URI to" depending on message type and subtype.

Some C2MS message header fields (UNIQUE-ID, PUBLISH-TIME, MW-INFO, CONNECTION-ID, NODE, PROCESS-ID, USER-NAME) seem to be included to expose some of the middleware characteristics to the application layer. Their usage is not specified and thus up to the concrete middleware employed. Because MO tries to be transport-agnostic, transport characteristics are usually not provided with such a large number of header fields to the application layer. Instead, transport-specific information is typically represented in the concrete structure of the "URI from" and "URI to" fields. The concrete form of a URI has to be defined by each transport and can include such information. This allows more flexibility regarding the middleware choice.

5.2. Messaging Patterns

Both C2MS and MO rely on message exchange for communication. In both standards no arbitrary exchange of messages is allowed, but message exchange follows certain patterns. C2MS calls them Message Exchange Patterns (MEPs), MO calls them Interaction Patterns (IPs). The following table lists the correspondence between both standard's patterns. Because C2MS relies on a middleware that only provides publish-subscribe style message exchange, there is no one-to-one correspondence of the patterns. For example, the C2MS Publish pattern can either map to a MAL SEND or to a MAL PUBSUB pattern, depending on the concrete usage.

C2MS Message Exchange Pattern	Corresponding MO/MAL Interaction Pattern
Publish	SEND or PUBSUB
Request/ACK	SUBMIT
Request/Response	REQUEST
Request/ACK/Response	INVOKE
Request/ACK/Interim Status/Response	PROGRESS
Request/Interim Status/Response	similar to PROGRESS, but MAL does not allow omitting the ACK message
Request/Response/Publish	INVOKE or PROGRESS
Publish/Request/Response	PUBSUB or SEND with subsequent REQUEST
Subscription	PUBSUB

5.3. Data Types

Both C2MS and MO define a set of data types that are referenced in the message or service descriptions. The following table lists the basic C2MS field data types and their corresponding MO data types. Most have a direct one-to-one correspondence, with notable exceptions mentioned extra.

C2MS Data Type	MO/MAL Data Type	Notes
Binary	Blob	
Boolean	Boolean	
Character	Octet or UOctet	see note 1
F32	Float	
F64	Double	
Header String	Identifier or String	
I16	Short	
I32	Integer	
I64	Long	
String	String	see note 1
Time	Time or FineTime or Duration	see note 2
U16	UShort	
U32	UInteger	
U64	ULong	
Variable	Attribute	
n/a, can be represented as String	URI	

Note 1: Due to the limitation to ASCII, C2MS::String is a severely limited representation of text strings, which makes it unsuitable for usage in non-English speaking systems. Likewise, C2MS::Character cannot represent many non-English characters. MAL::String can be any Unicode string, thus it is able to represent any string of any language. For C2MS::Character there is no direct corresponding MAL type because the notion of character is very different in Unicode. If a limited range numeric value shall be expressed the closest match would be MAL::Octet or MAL::UOctet, otherwise MAL::String should be used.

Note 2: Instead of one Time data type in C2MS, MAL provides three types: Time, FineTime and Duration. MAL::Time represents absolute times up to millisecond resolution, MAL::FineTime up to picosecond resolution. C2MS::Time resolution is limited to microseconds. Relative times are represented using their own data type in MAL (MAL::Duration), different than C2MS which uses one data type for both absolute and relative times. MAL does not prescribe concrete time data type representations, in contrast to C2MS, which prescribes a string representation. A concrete representation of MAL time types is achieved in the MO framework by a transport binding.

In order to compose more complex data types that are needed for meaningful data exchange C2MS and MO employ different approaches: C2MS provides UML object diagrams for each message that simply refer to the basic data types. Fields are grouped into required or optional fields and are optionally restricted to certain values. MO offers a complete type system by additionally providing abstract types like Element, Attribute and Composite. Composites are used for composing data structures. Lists and enumeration are part of MO, but are not present in C2MS. If such structures are needed in C2MS they are represented by field naming convention (lists) or restrictions of string values (enumerations). MO does not use UML diagrams to represent composed data types, but a tabular notation that is defined in [3]. MO data types can be reused across different service specifications, whereas C2MS data types are closely tied to a concrete message definition and thus cannot be reused for other messages directly.

6. Conclusion

The following table shall give a high-level overview of different areas of overlap. This table intentionally does not capture lower-level details, especially the depth to which some functionality has been specified. Generally, due to MO consisting of several standards, a broader set of functionality is offered in each area by MO. Please refer to the previous chapters for a detailed assessment. Entries marked as “proposed” are already on standardization track for an extended period of time and are all expected to be published in the near future. Entries marked as “planned” are in the charter of the CCSDS SM&C Working Group, but usually not more than a draft has been produced.

		C2MS	MO
Scope	Integration of satellite mission operations applications as main objective	yes	yes
	Ground-to-ground interfaces	yes	yes
	Space-to-ground	no, but possible	yes

	interfaces		
	Space-to-space interfaces	no, but possible	yes
Framework	Platform Independent Model	yes	yes, MAL
	API	unspecified, proprietary API exists (GMSEC API)	yes, for several programming languages
	Transport technology	unspecified, but some kind of message broker required	yes, several for ground and space
	Modeling language	UML	MAL
	Data model	messages and products are treated as data units	optional, COM
Functionality	Archiving	yes	yes
	Satellite Monitoring	yes	yes
	Satellite Control	yes	yes
	Infrastructure (Configuration, Health, …)	yes	proposed, Common
	Product Distribution	yes	proposed, MDPD
	Navigation	yes	planned
	Mission Planning	no	proposed, dedicated working group
	Automation	no	planned
	File and transfer management	no	planned
Extensible with bespoke interfaces		yes	yes

The overlap in objectives and functionality between C2MS and MO is substantial. It should be considered whether the international space community is served well by two incompatible standards serving the same purpose. C2MS at first appears to be the simpler standard, because it is contained in one medium-sized document. However, C2MS is incomplete and does not lead to interoperable systems. MO, beside from being published already, provides proven on-the-wire interoperability and gives system designers and implementers a complete stack at hand.

C2MS defines interfaces for common mission operations tasks that are mostly present in MO as well. The most notable difference where this is not the case, is all functionality regarding Navigation and Flight Dynamics. The MO interface counterparts usually offer more functionality (with the explicit possibility of implementations with restricted functionality) and either cover or are planning to cover all functional areas of C2MS and have a roadmap for more.

Both standards allow easy extension to custom interfaces and thus are not only advertised for use between agencies and vendors, but also inside a single entity, although this is not mandated. Further, none of the standards mandates any particular system architecture. How interfaces are implemented (in

one or several applications, using a service-oriented architecture or a monolith, …) is completely up to the system designer.

Instead of having two competing standards it makes sense to leverage the expertise gained from C2MS for the specification of planned MO services. Particularly Navigation and Flight Dynamics services can benefit, but also the product category catalogue can provide helpful input for the MDPD services.

7. Abbreviations

API	Application Programming Interface
C2CX	Component-to-Component Transfer
C2MS	Satellite Command & Control Message Specification
CCSDS	Consultative Committee for Space Data Systems
COM	Common Object Model
GMSEC	NASA Goddard Mission Services Evolution Center
MAL	Message Abstraction Layer
MC, M&C	Monitor and Control
MCMS	Mission Control Message Specification
MDPD	Mission Data Product Distribution
MO	Mission Operations
OMG	Object Management Group
RPC	Remote Procedure Call