The Consultative Committee for Space Data Systems

MO SERVICES AND SOIS ELECTRONIC DATASHEETS

CCSDS RECORD

CCSDS 870.10-Y-1

YELLOW BOOK

December 2021

The Consultative Committee for Space Data Systems

MO SERVICES AND SOIS ELECTRONIC DATASHEETS

CCSDS RECORD

CCSDS 870.10-Y-1

YELLOW BOOK

December 2021

AUTHORITY

Issue:	CCSDS Record, Issue 1
Date:	December 2021
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS). The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4).

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
Email: secretariat@mailman.ccsds.org

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Record is therefore subject to CCSDS document management and change control procedures, which are defined in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be sent to the CCSDS Secretariat at the email address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (BELSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- China Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Hellenic Space Agency (HSA)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Netherlands Space Office (NSO)/The Netherlands.
- Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 870.10-Y-1	MO Services and SOIS Electronic Datasheets, CCSDS Record, Issue 1	December 2021	Original issue

CONTENTS

CONTENTS

Figure Page

Table

1 INTRODUCTION

1.1 RATIONALE

Founded in 1982 by the major space agencies of the world, the CCSDS is a multinational forum for the development of communications and data systems standards for space data systems, with the goal of enhancing governmental and commercial interoperability and cross support while also reducing risk, development time, and project costs. Within CCSDS, two areas and their working groups have been tasked with looking at different areas of Application Layer interoperability, specifically:

- Mission Operations and Information Management Services (MOIMS), covering the interfaces between the ground mission control, planning and scheduling systems, and the spacecraft;
- Spacecraft Onboard Interface Services (SOIS), covering Application Layer services, devices, subnets, and the interfaces between the spacecraft and onboard electronic devices.

These two areas of standardization are focused separately, on the ground and up to the spacecraft (MOIMS) and on spacecraft only (SOIS). As part of describing the integration of these two domains, we need to describe how the MOIMS services interface to the spacecraft environment and also the possibility of interfacing some of the MOIMS applications layer services into the onboard environment. Since SOIS describes the spacecraft devices and their configuration, there is also the possibility of the SOIS descriptions informing the MOIMS about the spacecraft as an aid during development and operations preparation. Three possible cases of MOIMS integration with the real-time onboard environment were developed following recent discussions between SOIS and MOIMS:

a) Case 1: traditional case with SOIS covering the interfaces between spacecraft and onboard devices, while MOIMS is only on ground. Interfaces between flight and ground are ‘traditional’ Telemetry, Tracking, and Commanding (TT&C) with mapping from MOIMS to TT&C done on the ground.

b) Case 2: intermediate integrated case with MOIMS service interfaces extended across the space link to a Proxy interface that maps to traditional TT&C, hard and ‘soft’, real-time, onboard, control architecture. SOIS continues to cover the same interfaces as before.

c) Case 3: onboard integrated case with MOIMS-based mission control and monitor services and frameworks migrated on board and adapted to the real-time environment. The same kinds of hard real-time, RTOS, and SOIS spacecraft and onboard sub-network services are used. Some devices may even integrate directly with the MOIMS service framework.

In all of these different cases, it is essential to recognize that responsibility for robustness, reliability, redundancy, fault protection, and the real-time management of precious power, thermal, data bandwidth, data storage, and GNC/pointing resources, to say nothing of

autonomous fault protection, fault isolation, and robust onboard fault recovery remains with the real-time operating system and spacecraft real-time software. All of these topics are central to designing and developing real spacecraft. The migration of any MOIMS services on board must be developed to be consistent with this real-time environment, and any such functions must be implemented and tested to stringent quality standards (for example, NASA Class A, B, or C in reference [11]). These constraints may be somewhat relaxed only if the MOIMS MO functions are implemented in a separate co-processor that cannot interfere with real-time spacecraft operations.

This division of responsibility for Case 1 can be illustrated by an example whereby a hypothetical client institution designs, builds, and is involved in the operation[1] of a simple onboard instrument on both ESA and NASA spacecraft.

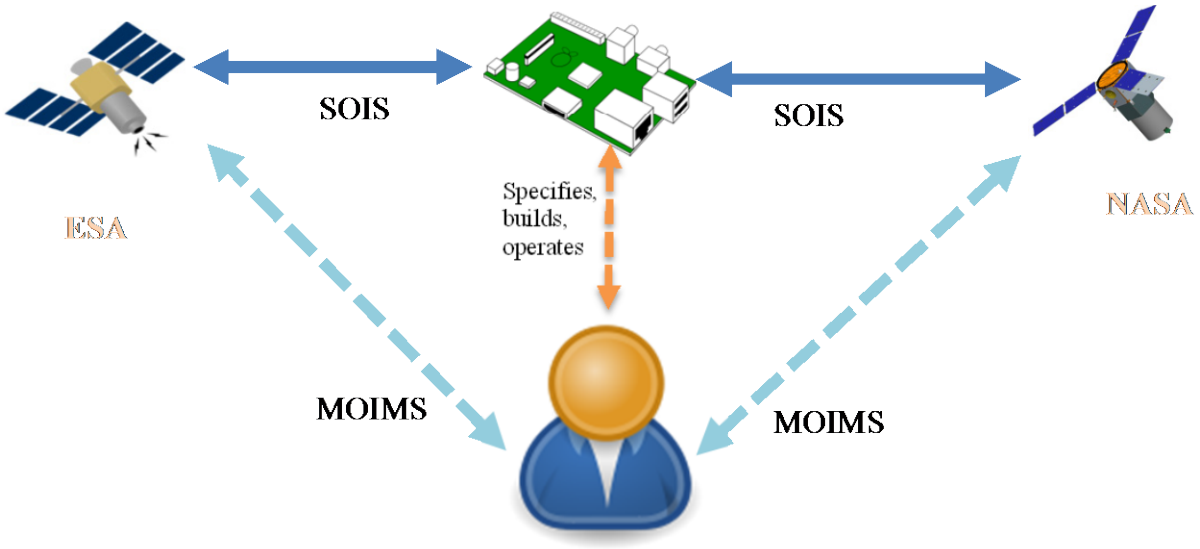

Figure 1-1: Two Hypothetical Missions Using CCSDS Standards

In such a case:

– SOIS provides services for communicating the information the client institution supplies to the spacecraft prime contractor in order to describe the overall platform and to describe the interfaces and functionality of the instrument or device with regard to the onboard platform;

– MOIMS provides services for communicating the information the client institution supplies to the spacecraft prime contractor, and further to the operator (agency), in order to operate the spacecraft and the instrument during the mission lifecycle.

If the available CCSDS standards are applied in all cases, the result can be minimal new work for the client, for the prime, and for the operator. This assumes that two copies of

[1] This involvement could potentially take the form of real-time commanding, requests for planning the scheduling of an activity, or be entirely delegated to the agency. The institute in question may or may not be part of either agency.

similar devices fly on these different spacecraft, built by different prime contractors, under the responsibility of, and operated by, different agencies, but using similar software. The same potentials for savings apply in more complex cases, in which the client, designer, manufacturer, and operator are not the same, or when there are multiples of each.

However, for SOIS and MOIMS, both sets of standards are new. This report is aimed at investigating how these aspects of a mission using both sets of standards would interact, with a view to ensuring those interactions are well-defined, well-understood, and unproblematic. This report also provides a limited exploration of three different approaches to MOIMS and SOIS integration that may be adopted by different missions. Of course, there are many possible combinations of the SOIS and MOIMS features, and these three cases are just intended to explore a representative range of possible integrations.

Following this introduction, this report:

- provides a brief overview of both sets of standards;
- performs an analysis of the relationships between the two standards;
- describes how the two standards could interoperate on a mission, as in the above example;
- explores the 3 cases in some greater depth;
- and provides some recommendations and conclusions.

1.2 REFERENCES

The following publications are referenced in this document. At the time of publication, the editions indicated were valid. All publications are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the publications indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.[2]

[1] *Spacecraft Onboard Interface Services—XML Specification for Electronic Data Sheets*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 876.0-B-1. Washington, D.C.: CCSDS, April 2019.

[2] *Mission Operations Services Concept*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 520.0-G-3. Washington, D.C.: CCSDS, December 2010.

[2] This report has not been updated to reflect any architectural changes in SOIS or MOIMS Areas since the original (unpublished) 2017 report.

[3] *Mission Operations Message Abstraction Layer*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 521.0-B-2. Washington, D.C.: CCSDS, March 2013.

[4] *Mission Operations Monitor & Control Services*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 522.1-B-1. Washington, D.C.: CCSDS, October 2017.

[5] *XML Telemetric and Command Exchange—Version 1.2*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 660.0-B-2. Washington, D.C.: CCSDS, February 2020.

[6] *Spacecraft Onboard Interface Services—Specification for Dictionary of Terms for Electronic Data Sheets*. Issue 2. Draft Recommendation for Space Data System Practices (Red Book), CCSDS 876.1-R-2. Washington, D.C.: CCSDS, June 2016.

[7] *Space Engineering—Telemetry and Telecommand Packet Utilization*. ECSS-E-70-41C. Noordwijk, The Netherlands: ECSS Secretariat, April 2016.

[8] *Reference Architecture for Space Data Systems*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 311.0-M-1. Washington, D.C.: CCSDS, September 2008.

[9] *Space Communications Cross Support—Architecture Requirements Document*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 901.1-M-1. Washington, D.C.: CCSDS, May 2015.

[10] *Mission Operations—MAL Space Packet Transport Binding and Binary Encoding*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 524.1-B-1. Washington, D.C.: CCSDS, August 2015.

[11] *NASA Software Engineering Requirements*. NASA Procedural Requirements, NPR 7150.2C. https://nodis3.gsfc.nasa.gov/npg_img/N_PR_7150_002C_/N_PR_7150_002C_.pdf

[12] Daniel P. Siewiorek and Priya Narasimhan. "Fault-Tolerant Architectures for Space and Avionics Applications." [2005]. https://www.cs.unc.edu/~anderson/teach/comp790/papers/Siewiorek_Fault_Tol.pdf

2 INTRODUCTION TO SOIS EDS

2.1 OVERVIEW

Electronic Data Sheets (EDS) (see reference [1]) is a concept to allow capturing relevant information about electronic equipment, components, devices, interfaces, behaviors, and deployments, in a format that can be electronically manipulated. This should capture the relevant aspects of a device, not just enable an efficient exchange of information (easing its maintainability, enforcing consistency, etc.), but also enable the development process of related software to be supported by the use of model-based software engineering techniques.

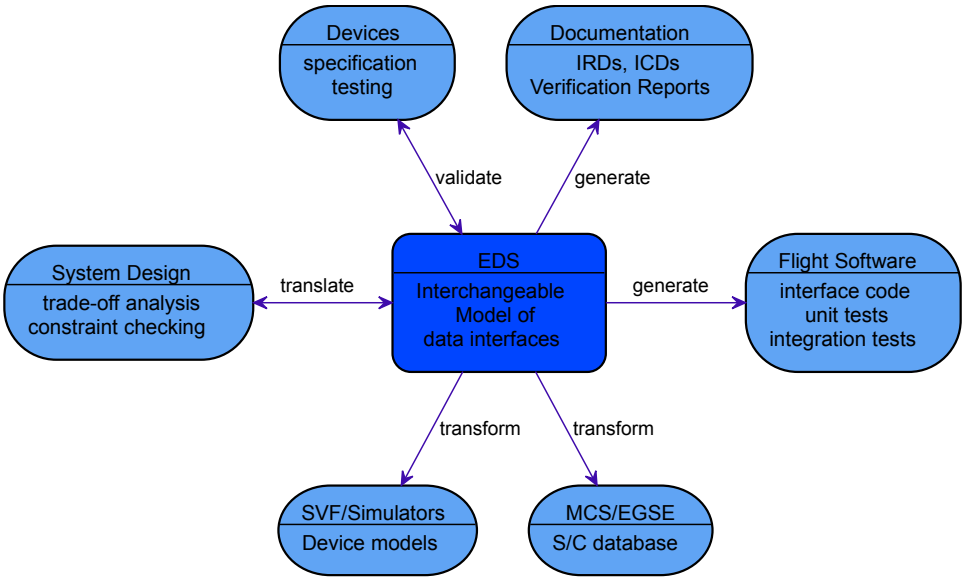

Figure 2-1: SOIS EDS Concept

In the course of the mission lifecycle, different parts of the overall system (which includes both space and ground) will need to represent, or interact with, an onboard device, including:

- tools used for the design and validation of the device itself;
- system design and analysis tools modelling a system using the device, for example, to check bus bandwidth and schedulability;
- tools used for the design and implementation of the Flight Software (FSW), which executes on the central onboard computer, and is in charge of communication with, and autonomous operation of, the device;
- mission control and Electronic Ground Support Equipment (EGSE) systems, in cases in which the device contributes to some portion of the spacecraft TM/TC definition;
- software validation facilities and operational simulators that model device behavior in order to validate the interaction of the FSW and the device;
- tools that generate portions of the system documentation.

This wide range of usages means that no one tool could plausibly meet them all, hence the need for a standardized data format that can operate across multiple tools. The SOIS Recommended Standard for Electronic Data Sheets (SEDS) takes the form of an eXtensible Markup Language (XML) schema designed for *tool interchange*, that is, exchanging device data between two or more software systems. These Electronic Data Sheets use a standardized Dictionary of Terms (DoT) to describe devices, their interfaces, and their behaviors. This DoT is extensible to permit new terms to be added and existing ones to be specialized as needed.

2.2 TERMINOLOGY

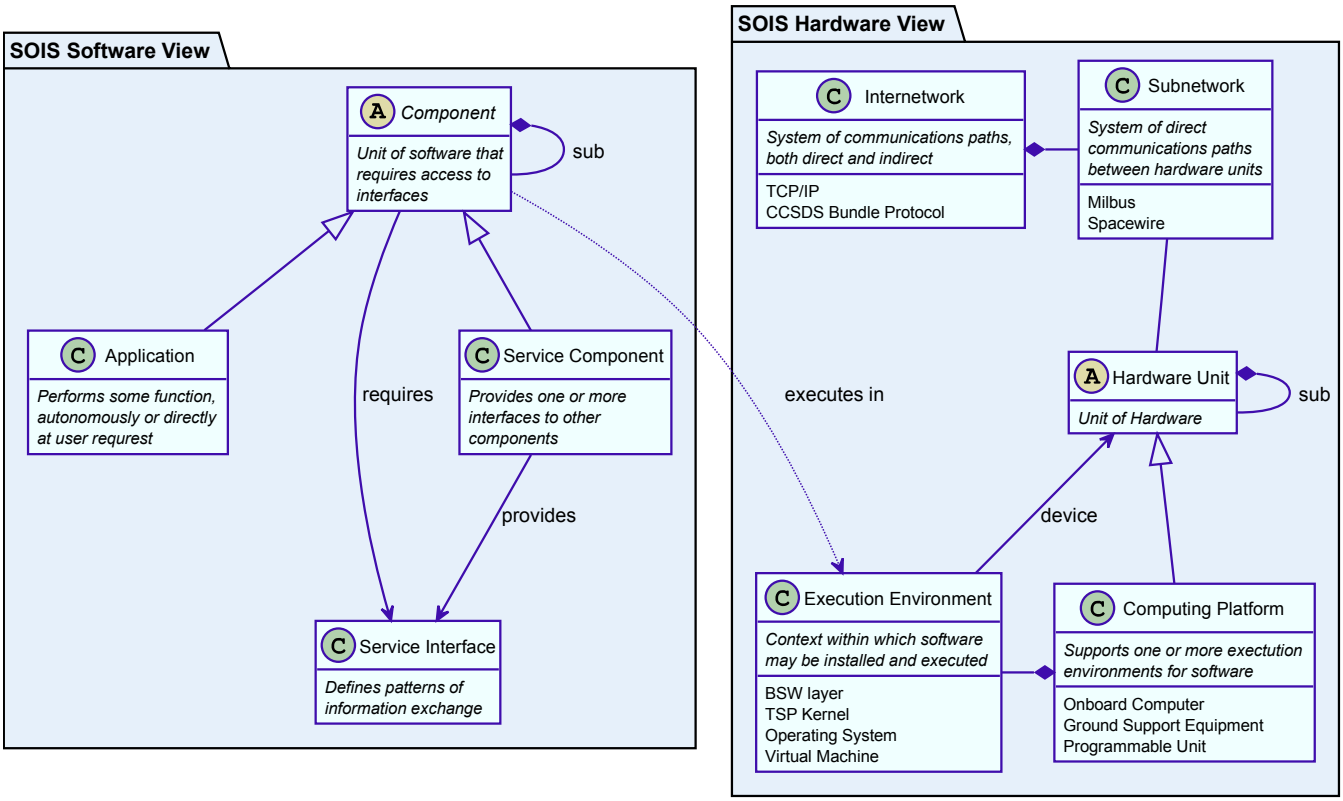

Figure 2-2: SOIS Terminology Used

SOIS uses the following terms to describe hardware and software elements of a mission:

Interface: The set of interactions performed by an object for participation with another object for some purpose, along with constraints on how they can occur.

Service Interface: Patterns of information exchange.

Component: A unit of software that requires access to interfaces and executes in an execution environment.

Application: A component that performs some function, autonomously or directly at user request.

Service Component: A component that provides one or more service interfaces to other components.

System: Area of analysis that can be described in terms of:

- components that communicate via internal service interfaces;
- external interfaces to one or more devices, each of which belonging to a single subnetwork.

Internetwork: System of end-to-end Network Layer communication paths, either direct or indirect, such as TCP/IP or CCSDS Bundle Protocol.

Subnetwork: System of direct communication paths between hardware units such as MILBus or SpaceWire. It is the Data Link Layer part of an internetwork.

Device: One or more units of hardware that could run embedded software or be supported by software running on the main Onboard Computer (OBC).

Hardware Unit: A unit of hardware that is part of a device.

Computing Platform: Supports one or more execution environments such as OBC, ground segment computer, and programmable unit for software and is integrated within a hardware unit.

Execution Environment: Context in a computing platform within which software may be installed and executed including Basic Software (BSW) layer, Time and Space Partitioned (TSP) kernel, Operating System (OS) or Virtual Machine (VM).

By definition above, device includes:

- hardware devices;
- smart devices running embedded software that might be at a specific version and result in changes to the datasheet;
- devices supported by associated software running on the main OBC (as in the case of Integrated Modular Avionics (IMA));
- devices directly attached to the main OBC board, and those attached via a data link such as MILBus;
- sensors, actuators, and instruments.

However, device does not include:

- platforms into which devices are integrated, as opposed to vice-versa;

- trivial components integrated into a device as part of the process of manufacture, such as screws and wires;
- software components or functions in general (other than in the case of Integrated Modular Avionics (IMA)).

Some common example of devices are: Star Trackers, Gyroscopes, Mass Memory Units (MMUs), Power Control and Distribution Units (PCDUs), Remote Terminal/Interface Units (RTU/RIU), Reconfiguration Modules (RMs), Telemetry, Telecommand, Reconfiguration, and Safeguard memory (TTRS), and all manner of science instruments.

One goal of SOIS EDS is to define how to take an external entity, such as a device, and make it accessible from within a system. This is done by describing a set of **device services**. These are accessible on that same basis as any other service and support all functions of the device by communicating with it using lower-level mechanisms.

2.3 SOIS REFERENCE ARCHITECTURE

The SOIS Reference Architecture describes how SOIS standards and recommendations fit together to specify and implement the communications between an *Onboard System* (i.e., OBC and FSW), and other onboard *Devices*.

It is worth noting that SOIS has not attempted to standardize the onboard, real-time, environment. These systems are many and varied, ranging from stripped-down, built for purpose, real-time executives to real-time, time and space partitioned, full featured, potentially virtualized operating systems. However, all of the SOIS services and devices, in fact, all software on board any of these spacecraft, are assumed to operate in such a real-time environment, with software built and tested to stringent quality standards (for example, NASA Class A or B in reference [11]) These real-time systems have the responsibility of providing a robust, fault-tolerant, and cybersecure environment that manages the operation of the spacecraft, manages use of precious resources such as power, propulsion, thermal, pointing control, data, and bus utilization, and hosts the various science instruments and other devices.

These real-time systems will not always be explicitly shown in all of these diagrams, but it should be understood that any references to onboard systems, OBCs, and/or FSW always assume the presence of a real-time environment that has these characteristics. And this is distinct from the typical terrestrial system deployments. Table 2-1, copied here from reference [12], demonstrates the differences among these environments.

Table 2-1: Comparison of System Features in Different Operational Environments

Operational Environment	**Commercial**	**Space**	**Avionics**
Mission duration	Years	Years	Hours
Maintenance Intervention	Manual	Remote	After mission
Outage response time	Hours	Days (Cruise phase)	Milliseconds
Resources -Power - Spare parts	Unlimited Unlimited	Minimal None	Medium After mission
Fault-Tolerant Approach			
Fault avoidance and fault intolerance	Burn-in	Radiation-hardened components	Shake, rattle, roll
		Design diversity	Design diversity
		Safe system	
Fault tolerance		Component-level redundancy	
		Subsystem-level redundancy	Subsystem-level redundancy
	Multi-computer	Multi-computer	Multi-computer
	Retry	Retry	
	Firewalls		Firewalls
	Software patches	Software reload	

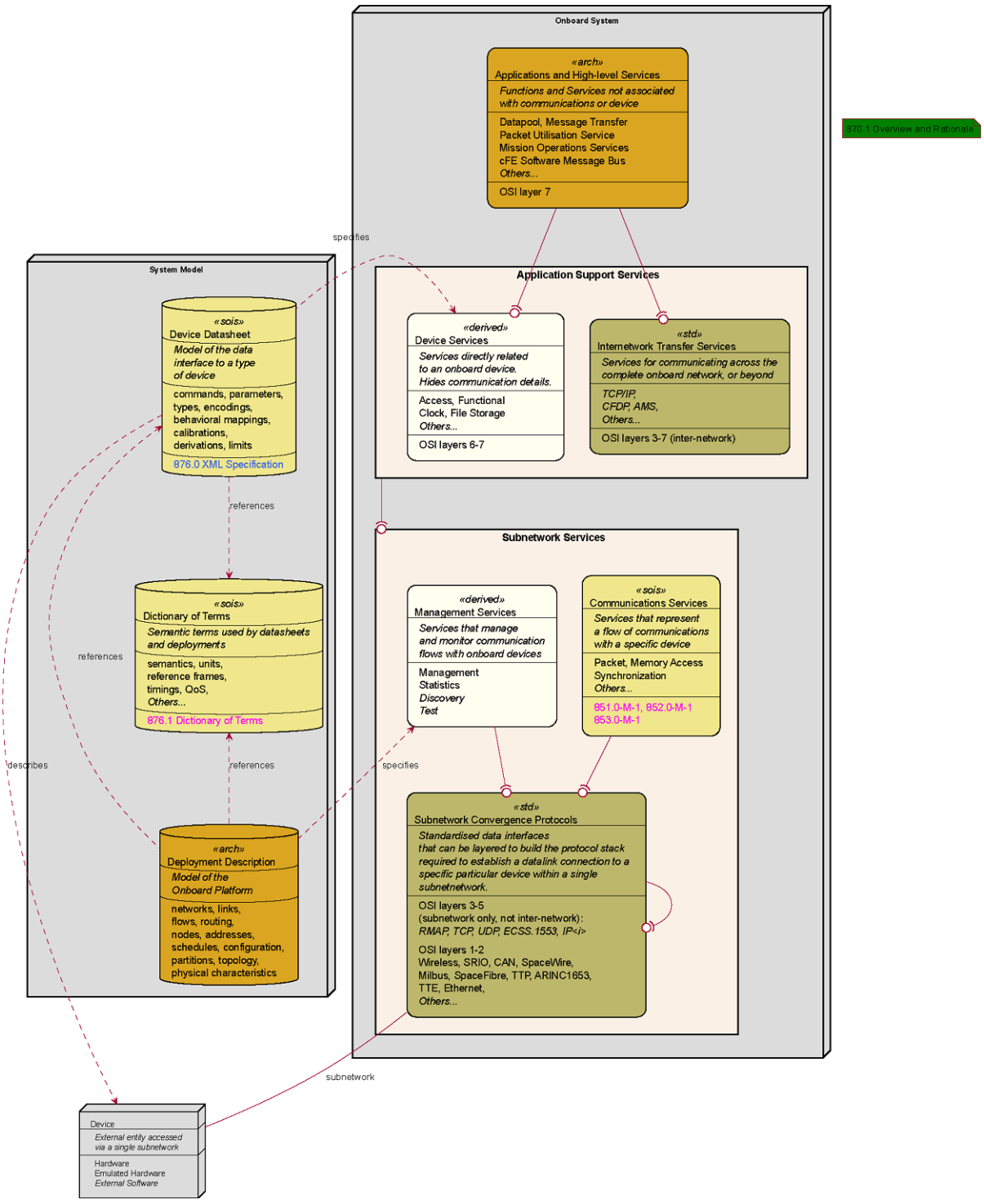

Figure 2-3: SOIS Reference Architecture

In the above diagram:

– <<sois>> indicates elements defined by SOIS standards; these have the standard in question specified;

  - <<std>> indicates elements defined by other standards;
  - <<arch>> indicates architecture or mission-specific elements;[3]
  - <<derived>> indicates run-time elements that can be manually or automatically derived from the indicated specifying elements.

In this architecture, the Onboard System is considered as two layers:

- *Application Layer*, where application and services components communicate according to defined service interfaces. This includes the *Device Services*, which make the functionality of a device available at this layer, and are logically derived from the datasheet for that device. Also at this layer are *Internetwork Transfer Services* and protocols such as CCSDS File Delivery Protocol (CFDP) that allow communication across different subnetworks. Mission operations services are included in the application layer and have traditionally been used to expose interfaces to mission operators. In these cases, the services are written to Flight Software standards and use flight software development processes.
- *Subnetwork Layer*, where binary data flows between endpoints according to a stack of communications protocols across one or more subnetworks. The services exposed include both the standardized *SOIS Communication Services* and architecture-specific *Management Services*, which are logically derived from the parts of the System Model that specify how everything is connected together. Mission operations services at the Application Layer would interface to the Management Services to support mission operations.
- Not shown in this diagram is the space-link interface to ground, as that is outside the scope of SOIS. This space link may be handled:
  - at application level;
  - by the Internetwork Transfer Services;
  - as a dedicated subnetwork with its own space-link subnetwork convergence protocols.

The SOIS EDS for a device (a Device Datasheet) forms part of the overall System Model, describing the details of one particular model of onboard device. This includes how the services it provides at the Application Layer can be implemented in terms of subnetwork-layer services.

In this SOIS model, MO Services are not shown but could interface to the architecture-specific *Mission Operations Services*. Selected MO Services may be implemented to Class A/B standards so as to be suitable as part of the onboard Application Layer, as well as on the ground, and this is addressed in Section 4. If they are on board, they will typically be at the *Applications and High-Level Services* above SOIS applications.

[3] From the point of view of the SOIS reference architecture, such elements may follow any standards, or none.

Some of the later sections envision MO compliant plug and play devices, not unlike a USB camera or disk drive, that provide a common service interface. An extension of the EDS to allow the interfaces of such devices to be described has yet to be defined.

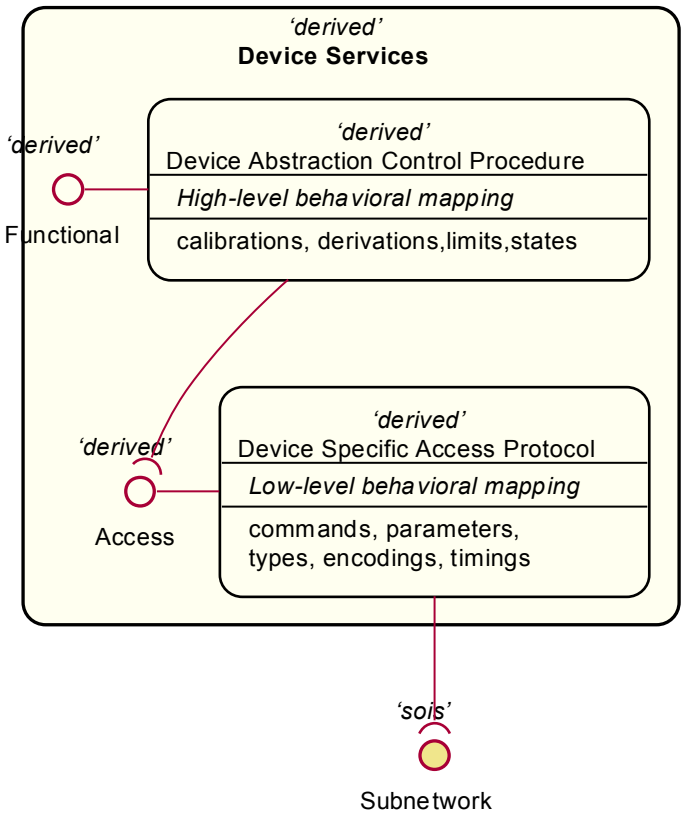

Figure 2-4: Device Services Details

All Device Datasheets use the EDS schema and the Dictionary of Terms (DoT) to define the contents and interpretation of messages exchanged between applications and the device across the subnetwork layer. It also may define the state machines describing message exchange protocols and device states. By specifying the device data interface in terms of this abstract model, it becomes possible to determine the correctness and completeness of a device datasheet in isolation from the actual FSW that will be used to communicate with the device in any particular case.[4] This validated datasheet can then be used as an input to the development and testing of those systems that interact with the device (e.g., the spacecraft FSW, system engineering database, checkout systems, etc.). This could, in fact, include MO Services that access device and spacecraft knowledge in the form of EDS datasheets.

For flexibility, the Device Services are conventionally defined in two parts:

[4] For example, this can be done by using the datasheet to process logs taken during hardware testing, or ideally, by doing such testing using a tool with EDS support.

- a Device Specific Access Protocol (DSAP), connecting to the device via a subnetwork interface and exposing an access interface, which defines the lowest-level access to all raw decoded data transmitted to and from a particular class of device;
- a Device Abstraction Control Procedure (DACP), connecting to the DSAP interface and exposing a Functional interface, which provides higher-level access to calibrated values or derived parameters, and restrictions on how the device can be operated based on its current state.

Both of these service interfaces are device-specific because different devices support different sets of data. These are split to allow missions the option of supporting either one, or both.[5]

In the typical case, there will be a single, distinct component providing each interface, and the component implementing the higher-level interface will be defined in terms of the lower-level one. The lowest-level component will require one or more subnetwork-level interfaces. However, interfaces in a datasheet can be defined in multiple parts and collected together using inheritance. This allows a part of the interface of a device to be standardized, or pre-specified, while other parts are manufacturer additions or customizations.[6] SOIS does not define or require this approach, but SOIS Data Sheets include support for this use case. It would be up to the mission or other standards, such as those in MOIMS SM&C, as to whether the data sheets would be defined using this common services approach.

A key characteristic of SEDS interfaces is that, while they support two-way data exchange, they are partitioned into:

- parameters:[7] messages coming from the device, plus those two-way exchanges whose sole purpose is to get or set a discrete value on a device;
- commands: messages sent to a device, plus two-way exchanges with any purpose other than reading or writing a single parameter.

In some cases, the classification of a particular exchange as a parameter or command with a single argument is debatable; however, this conceptual split, which was taken from XML Telemetric & Command Exchange (XTCE) (reference [5]), reflects the way Mission Control Systems (MCSes) and operations teams work.

Mapping the device-specific interfaces defined in the datasheet to actual Application Programming Interfaces (APIs) or messaging interfaces used by a specific FSW architecture is explicitly not the concern of a datasheet. This is intentional since the same device datasheet can be used when the same hardware device is used for missions that have different

[5] It is common for there to be no requirement to perform calibration on board. In such cases the FSW uses only the access-level interface, while the datasheet still contains calibration data for the sake of ground systems, simulators, etc.
[6] For example, for a GPS device there might be a standardised interface for positional data, supplemented by device-specific diagnostic and recovery interfaces.
[7] SEDS parameters are commonly aggregates of primitive values; as such they arguably more closely resemble packets than the individual parameters of typical datapool-based software architectures.

software architectures. Instead, the information required to map to any specific architecture is the concern of the code generation toolchain that interprets the EDS, which would either be architecture specific or highly configurable. The result of this approach is code that directly uses the required coding standards and the native synchronization and communication APIs of the target architecture, not an additional universal wrapper layer.

All provided and required interfaces are explicitly defined within a datasheet; there is no privileged treatment or special-casing for standardized interfaces. The datasheet construct used to define interfaces can be used to specify both high-level functional interfaces[8] and low-level binary interfaces containing data encoded in a specific way, as is commonly produced by device hardware. Such subnetwork interfaces can be directly mapped to specific logical data links supported by the communications service of subnetwork layer. This mapping is typically static, that is, done at system design time and recorded in a Deployment Description.[9]

As a consequence of the above, SEDS interfaces descriptions are able to not only specify a new interface (a capability shared by many other similar component systems), but to capture an existing interface. This includes cases in which that interface was designed and implemented without knowledge of the SEDS or SOIS. In other words, SEDS allows description of an existing, well-tested, and certified legacy device and wrapping it as an interface of a compliant component within a system. This avoids the costs and risks associated with producing custom variants of a device containing support for the technology stack of a particular agency.

The SOIS EDS approach can be thought of as Datasheet Mediated Interoperability. The EDSes do not directly participate in the operational interfaces, but they have a strong role in the specification of and interpretation of the details of actual device, sub-net, and component interfaces, as well as in the specification of component configurations.

[8] This can take the form of an API, or a messaging interface following known systematic encoding rules.
[9] In some cases, the mapping may be modifiable though the management services; the consequences of doing so should be carefully analysed in the scope of the mission architecture, technology, and requirements.

2.4 SIMPLE EXAMPLE OF DATASHEET USE

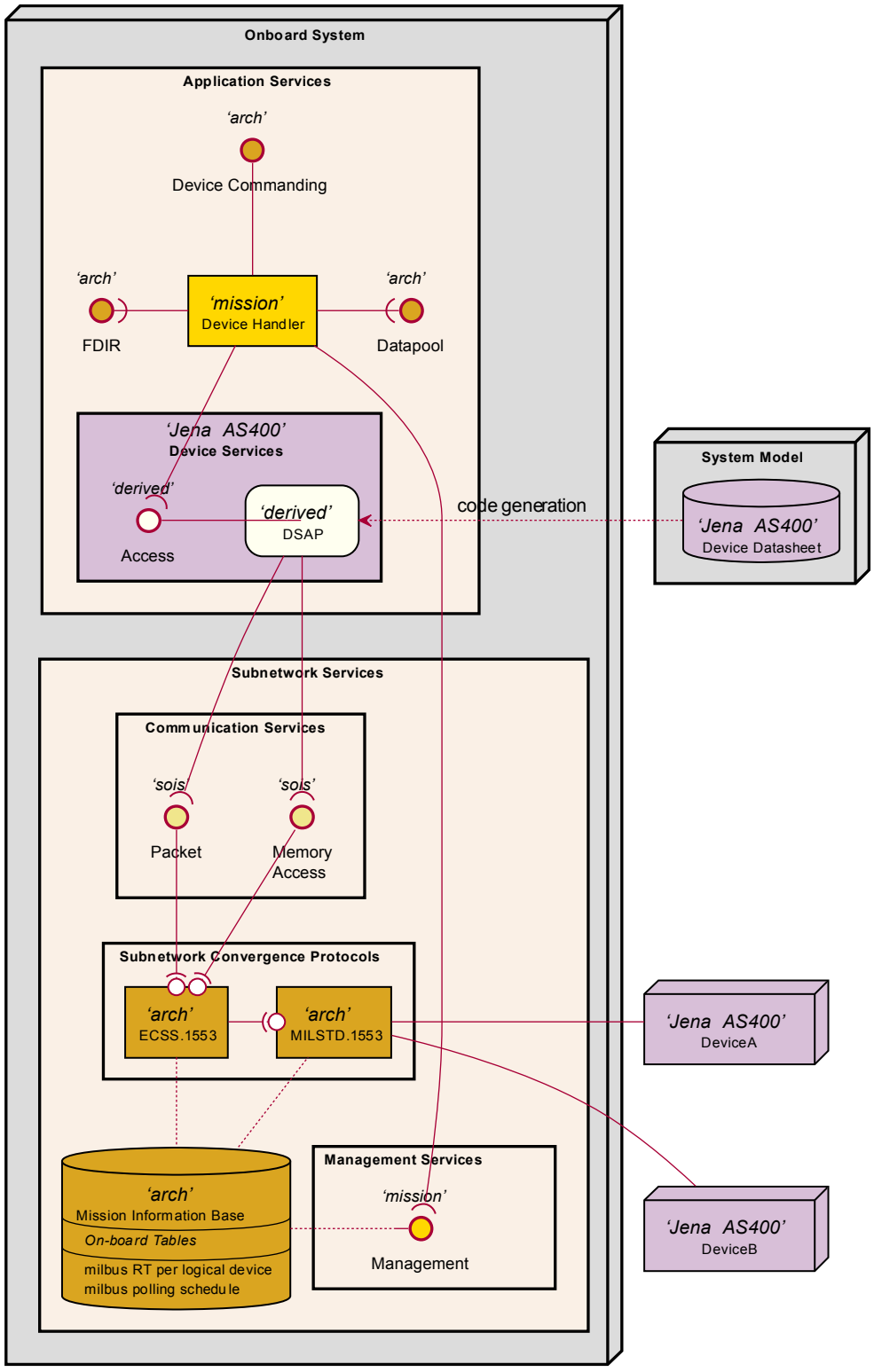

Figure 2-5: Sample Realization of SOIS Reference Architecture

The above diagram provides an example of how the SOIS Reference Architecture could be realized to integrate and manage a particular model of onboard device, in this case the Jena AS400 Star Tracker, of which prime and redundant instances are available as Remote Terminals (RTs) on a MIL-STD-1553 bus, of which the Onboard System is the bus controller. In it:

- The device datasheet is used as an input for code generation of the **DSAP**, which in this case is a simple set of functions that perform encoding, classification, and decoding of device data.
- A mission-specific Device Handler uses the corresponding Access Interface (i.e., calls those functions) to implement:
  - commanding of the device, either from the ground or onboard autonomy functions;
  - reading of telemetry from the device and storing it in a data pool, from which it can be reported to the ground and accessed by onboard autonomy functions;
  - triggering Fault Detection, Isolation, and Recovery (FDIR) logic on failures.
- The Memory Access Interface is understood to map directly to polling of a specific Sub-Address (SA) to read fixed-size, high-priority data, so the code generator creating the DSAP implementation translates calls to that interface in the datasheet to the corresponding calls to this architecture-specific implementation.
- The Packet Interface is understood to use the ECSS.1553 protocol to asynchronously transfer blocks of data (using multiple SAs per cycle), so the code generator creating the DSAP implementation translates calls to that interface in the datasheet to the corresponding calls to this architecture-specific implementation.

In this simple example, there is no explicit Deployment Description; the corresponding data is available as a set of hand-written onboard tables recording:

- the RT address of each instance of the device;
- the polling schedule for the device.

These are used in the implementation of each of the protocols and adjusted via the Management Interface (e.g., to adjust the polling schedule based on spacecraft mode).

2.5 OTHER CCSDS SERVICES AND THE OSI MODEL

All CCSDS standards follow the OSI model, shown in figure 2-6, in which logical communication between peer entities at each layer, shown horizontally, is implemented by messages going down the stack to the lowest layer, across the physical medium, and up the stack on the other side.

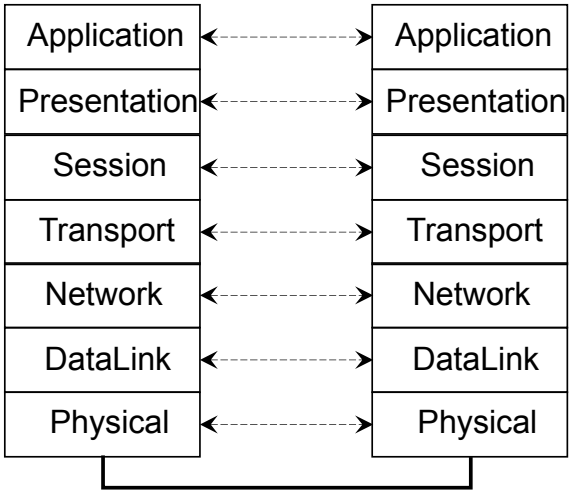

Figure 2-6: OSI Model

The relationship between SOIS and other areas of standardization within CCSDS is well established:

- CCSDS space link standards specify communications between spacecraft and ground, or pairs of spacecraft, for OSI layers physical(1) and data link(2).
- Cross Support Services (CSS) standards allow interoperability of ground stations by standardizing their interface with the Operations Control Centre, for all OSI layers.
- Space Internetworking Service (SIS) standards govern application-to-application communication on board a single spacecraft, communications among multiple spacecraft, and communications between space-based applications and their counterparts on Earth and/or other planetary bodies, for OSI layers network(3) through application(7).

Table 2-2: OSI Layering of SOIS Reference Architecture

<table>
<tr><th>Layer</th><th>Function</th><th>Per-Device</th><th>Cross-subnetwork</th><th>Per-Subnetwork</th></tr>
<tr><td>7</td><td>Application</td><td rowspan="2">Device Services</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>6</td><td>Presentation</td></tr>
<tr><td colspan="5">Communications Service Interfaces</td></tr>
<tr><td>5</td><td>Session</td><td rowspan="5"></td><td rowspan="2">Internetwork Transfer Services</td><td rowspan="3">Subnetwork Convergence Protocols</td></tr>
<tr><td>4</td><td>Transport</td></tr>
<tr><td>3</td><td>Network</td><td rowspan="2"></td></tr>
<tr><td>2</td><td>Data Link</td><td rowspan="2"></td></tr>
<tr><td>1</td><td>Physical</td><td></td></tr>
</table>

Looking at the overall SOIS reference architecture, each component that does communication-related processing can be statically assigned to a range of OSI functional layers, as shown in table 2-2, above.

- A SOIS EDS specifies the interface to a device at the presentation layer (6).[10] Consequently, while the encoding is fixed, it can be layered over any lower-level protocols by specifying the details of the transport technology used. This is done by defining subnetwork terms for each subnetwork interface referenced by the device. This means that the processing performed by the Device Services component specified by the datasheet will be at OSI layers 7 and 6.

- The Communications Service Interfaces (i.e., PS, MAS, and SYNC) sit immediately below the datasheet, carrying encoded data, and so are below level 6.

- The Subnetwork Convergence Protocols do all the processing necessary to deliver blocks of data to and from a known endpoint *within a single known subnetwork*. Depending on the subnetwork in question, this may involve processing at any OSI layer[11] from 5 to 2. In some cases, the subnetwork does not actually require that level of processing. That can be modelled by treating the corresponding layering function as a null or identity transform. This leads to a uniform treatment of all subnetworks.[12]

- The Internetwork Transfer Services do all the processing necessary to deliver a block of data to and from a known endpoint *within any accessible subnetwork*. This corresponds to OSI layers 4 to 3 and may include features of layer 5.

[10] As previously discussed, this is required for capturing the interface of an existing hardware device, as opposed to providing a specification to which a hardware device should be constructed.
[11] Some existing protocols, such as TCP/IP and SPACELINK packets, do not cleanly separate the processing of all OSI layers; however, in all known cases, they fit within the range of layers assigned to each component.
[12] A specific software implementation is of course free to optimise this case.

3 INTRODUCTION TO MO SERVICES

3.1 OVERVIEW

CCSDS Mission Operations (MO) (reference [2]) is a set of standard end-to-end services based on a Service Oriented Architecture (SOA) intended to be used for mission operations of space assets.

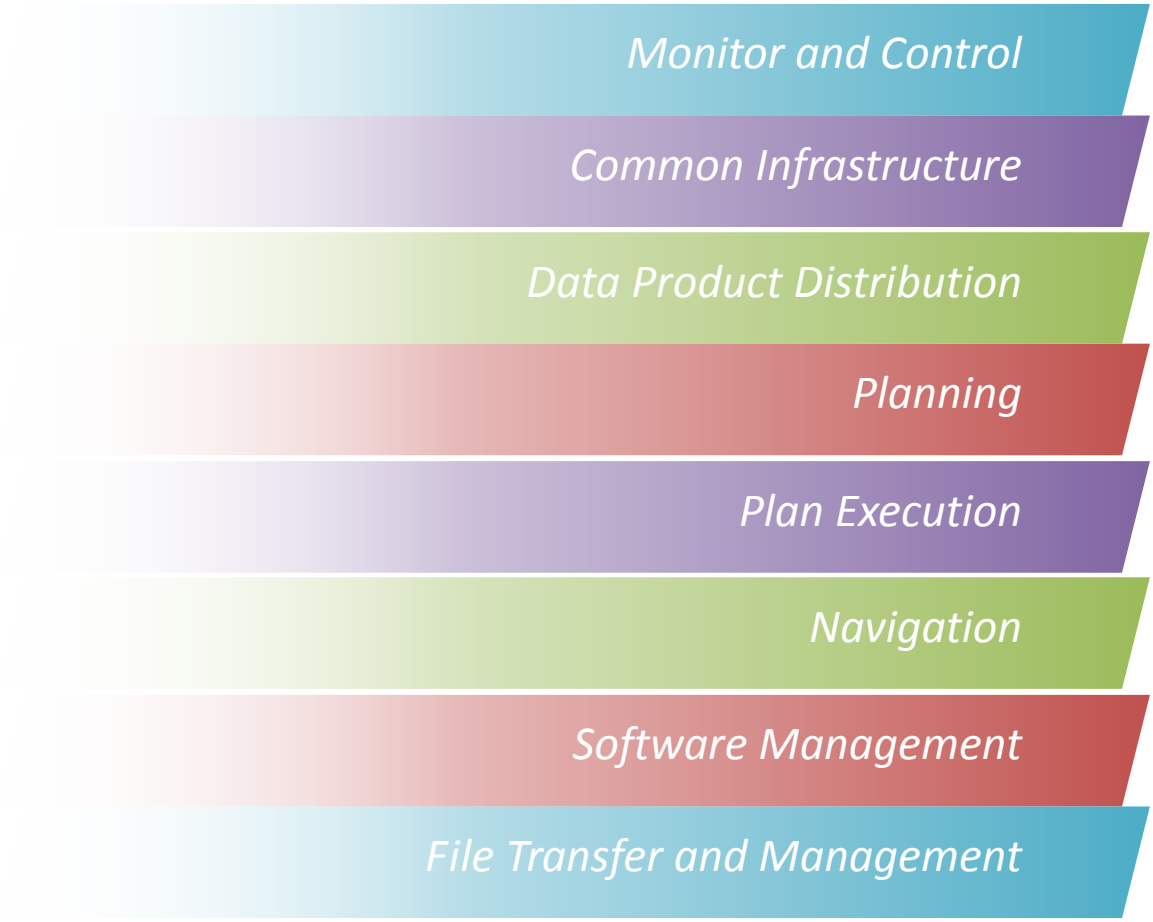

Figure 3-1: CCSDS MO Scope

CCSDS ***Mission Operations*** define specifications for a set of standard operations services (reference [2]) for the spacecraft operations.

To support the development of standardized services, MOIMS has started by defining an abstract open architecture and framework that is:

– independent from implementation, message encoding, and communication technology;

– able to integrate new and legacy systems of different organizations;

– designed to support the long lifetimes of space missions;

– based on an SOA;

– allows defining new bespoke services for a mission-specific need.

Figure 3-2: Details of an MO Service

Each MO Service, whether standardized or bespoke, is defined by a set of operations that the provider of the service makes available to be used by the service consumer. Each operation is defined from a template specified by an interaction pattern; one of send/submit/request/invoke/progress/pubsub. Each such pattern has a list of the messages that are exchanged between service provider and consumer to implement the operation.

The MO concept is supported by the MO framework, which, in an abstract manner, allows the specification of an MO Service, its operations, its data model, and a few related generic facilities, such as archiving. These abstract definitions must be transformed using a real, concrete, data representation and data transport binding in order to be implemented in a real system.

At the core of the framework is the Message Abstraction Layer (MAL) (reference [3]), which defines a standard XML notation for service and data specifications. These abstract specifications then get transformed into the appropriate message encoding and transport technology that are specific to the target deployment and used at implementation time. This approach allows the use of the most appropriate encoding/transport for each deployment, for instance, XML over HTTP for a service deployed on the ground, Binary over Space Packet Protocol (SPP) for a service deployed on the space-to-ground link, and Binary over SOIS subnet for a service deployed on board.

Figure 3-3 is a graphical representation of this transformation from abstract services to a deployable implementation framework.

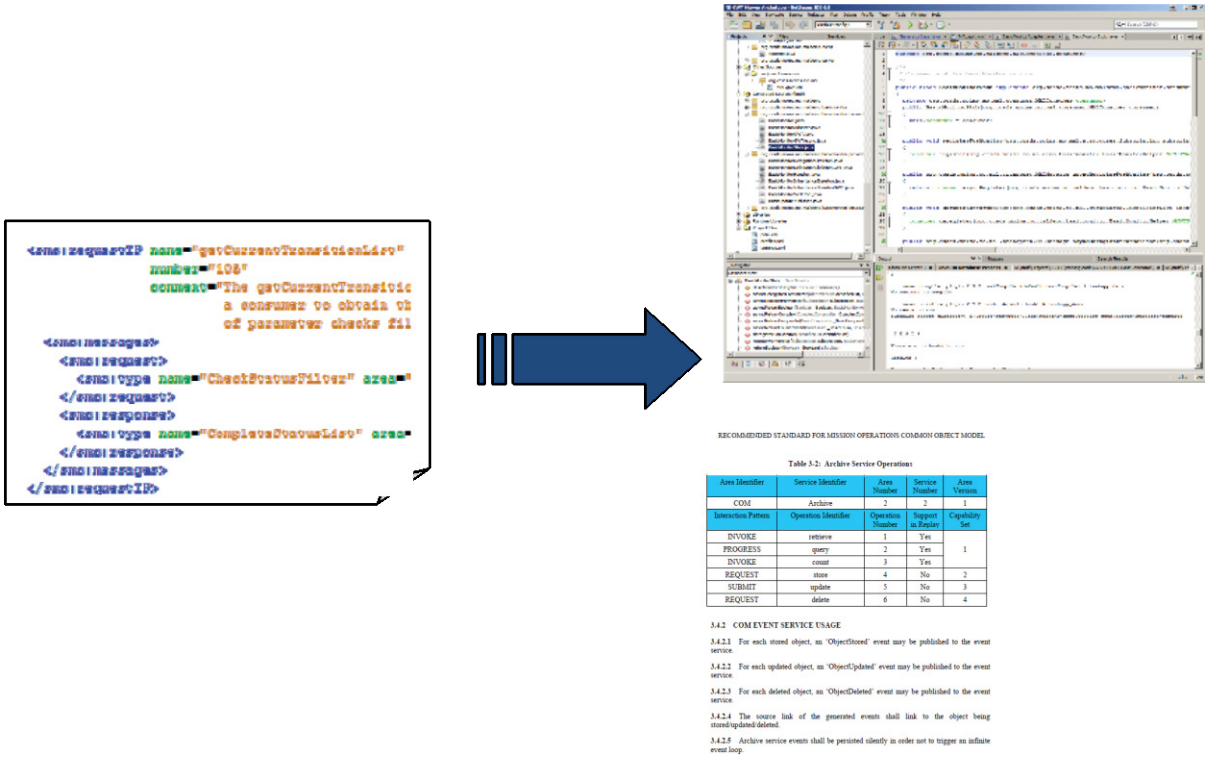

Figure 3-3: Transformation of MAL into Technology-Dependent Interface Specifications

3.2 MO AND OTHER CCSDS SERVICES

In the OSI model, an MO Service, as defined in MAL, is at the Application Layer (7). Consequently, it can be layered over any lower-level protocols by specifying the details of the encoding and transport technology used. The abstract MAL must be bound to a data representation and a transport binding in order to provide a deployable service, but different bindings will typically not directly interoperate. The MAL defines a process for different choices of bindings to be connected by using a protocol matching bridge.

Regardless of which of the three deployment cases are adopted (see Section 5 for details), all of the communications between MO terrestrial services and the spacecraft and SOIS onboard devices will utilize other CCSDS services for space link communications, cross support access and control of ground station communications assets, space internetworking services (where employed), and other services like navigation, radiometric, and security that are defined elsewhere in CCSDS. (For details of how all of those other CCSDS services and protocols work, see *Space Communications Cross Support—Architecture Requirements Document*, reference [9].)

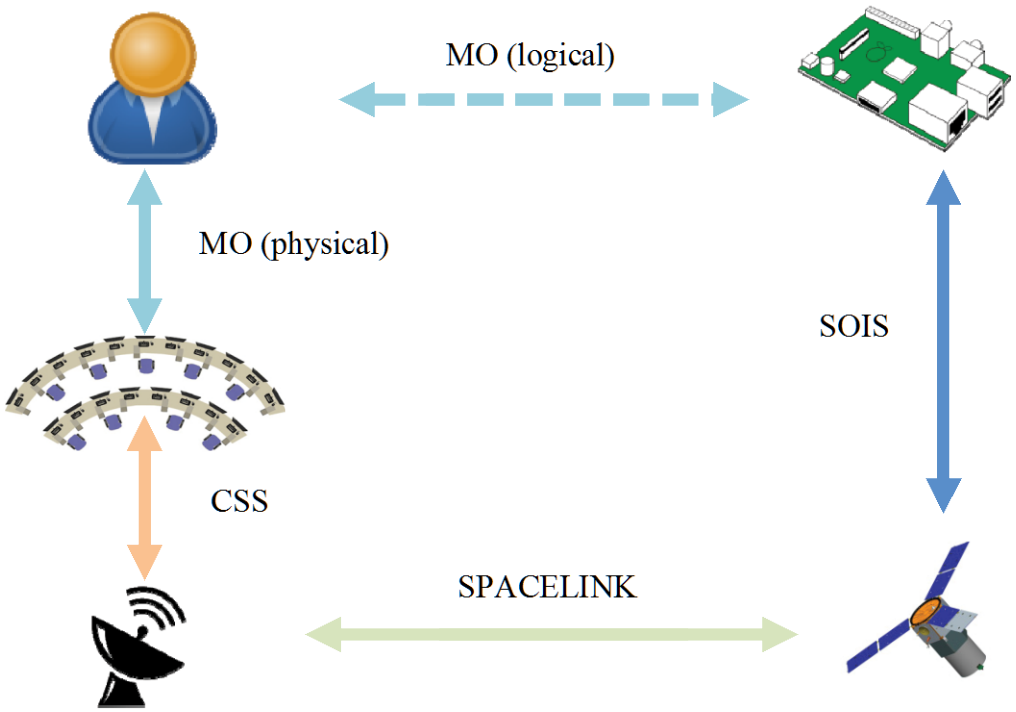

Figure 3-4: MO, SOIS, and Other CCSDS Services

Figure 3-4 outlines how this works:

– The client institute uses an MO Service to monitor or control a device. Logically, the corresponding set of messages flow to and from the device.

– However, those messages actually pass through each of:

  • a suitable MO transport (e.g., HTTP) to get to the control center;

  • where they are turned into protocols suitable for transmission over a space data link (SDL);

  • CSS protocols are used to get the SDL frames to the ground station;

  • the SDL protocols, error correcting codes, and suitable modulations are used to communicate over the RF space link to the satellite;

  • the original MO Service requests, in whatever local form is required on board, are extracted; and

  • SOIS protocols are used to send them to the device.

– At each stage, the messages may be either translated into, or layered inside, the protocols used in the next step.[13]

[13] For example, the control center typically generates SPACELINK TC transfer frames that are then encapsulated into appropriate CSS SLE messages.

- In many cases, the original commands may be translated into very different forms suitable for the different operating/communications environments.
- The opposite path is followed for a reply from the device to the user.

SIS protocols might also be used in the case of a relay satellite (not shown).

This top-level view of end-to-end flows naturally leaves out many of the details, which are discussed in subsequent sections.

4 ANALYSIS

4.1 AREAS OF OVERLAP BETWEEN THE STANDARDS

As presented in Sections 2 and 3, the two sets of standards have different scopes and purposes, but do have two areas of overlap:

– Interfaces, which include the protocol bindings and are the place where the behavior of an object is exposed. Objects may have one or more interfaces.

– Types, which describe and constrain the contents of the messages between two or more communicating entities.

In the case of the hypothetical mission shown in figure 1-1, if the hardware device produced by the client institute has a certain number of configurable settings and modes, the spacecraft FSW can adjust those settings according to an interface specified in the EDS datasheet for the device. Figure 4-1 shows a device interface expressed as an EDS.

3.1.2.1 PDU: TelecommandModeType

Byte Offset	Bit Range	Field Name	Type	Encoding	Fixed Value	Description
0	[0..0]	type	TelecommandTypeEnumType	UNSIGNED	Mode	
1	[0..31]	mode	ModeType	UNSIGNED		
Fixed byte length is 5						

PDU Binary Encoding for TelecommandModeType

3.1.2.2 PDU: TelecommandUserDataType

Byte Offset	Bit Range	Field Name	Type	Encoding	Fixed Value	Description
0	[0..0]	type	TelecommandTypeEnumType	UNSIGNED	UserData	
1	[0..7]	userDataLength	Octet	UNSIGNED		
2	[0..7]	userData	Octet	UNSIGNED		
Repeat previous 1 entries a total of 'userDataLength' times						
Length is variable.						

PDU Binary Encoding for TelecommandUserDataType

Figure 4-1: Binary Interface to the Device Expressed as a CCSDS EDS

NOTE – The above formatted EDS extract shows how the Protocol Data Units (PDUs) exchanged with the device are split into fields with associated encodings, types, and semantics.

The same interface could also be expressed as a MAL data structure, allowing operators or high-level software applications to configure those settings on the device. When a service interface is expressed in MAL, the encoding and layout details are intentionally delegated by MAL to concrete technology mappings. This is suitable for the intended usage of MAL, as it allows decisions on how to efficiently encode data in each deployment to be taken centrally and therefore consistently, which supports reuse from one mission to another, while the comms protocol and encoding (e.g., on board) may change, but the service spec remains the same.

However, for interfacing directly with hardware, things are different, as any such decision on how data *should* be encoded does not affect the fact that the hardware *does* encode it a particular way. The EDS directly describes the encoding that must be used at the device interface. Figure 4-2 is a sequence diagram showing the flow of different data objects among the different functions on the end-to-end path.

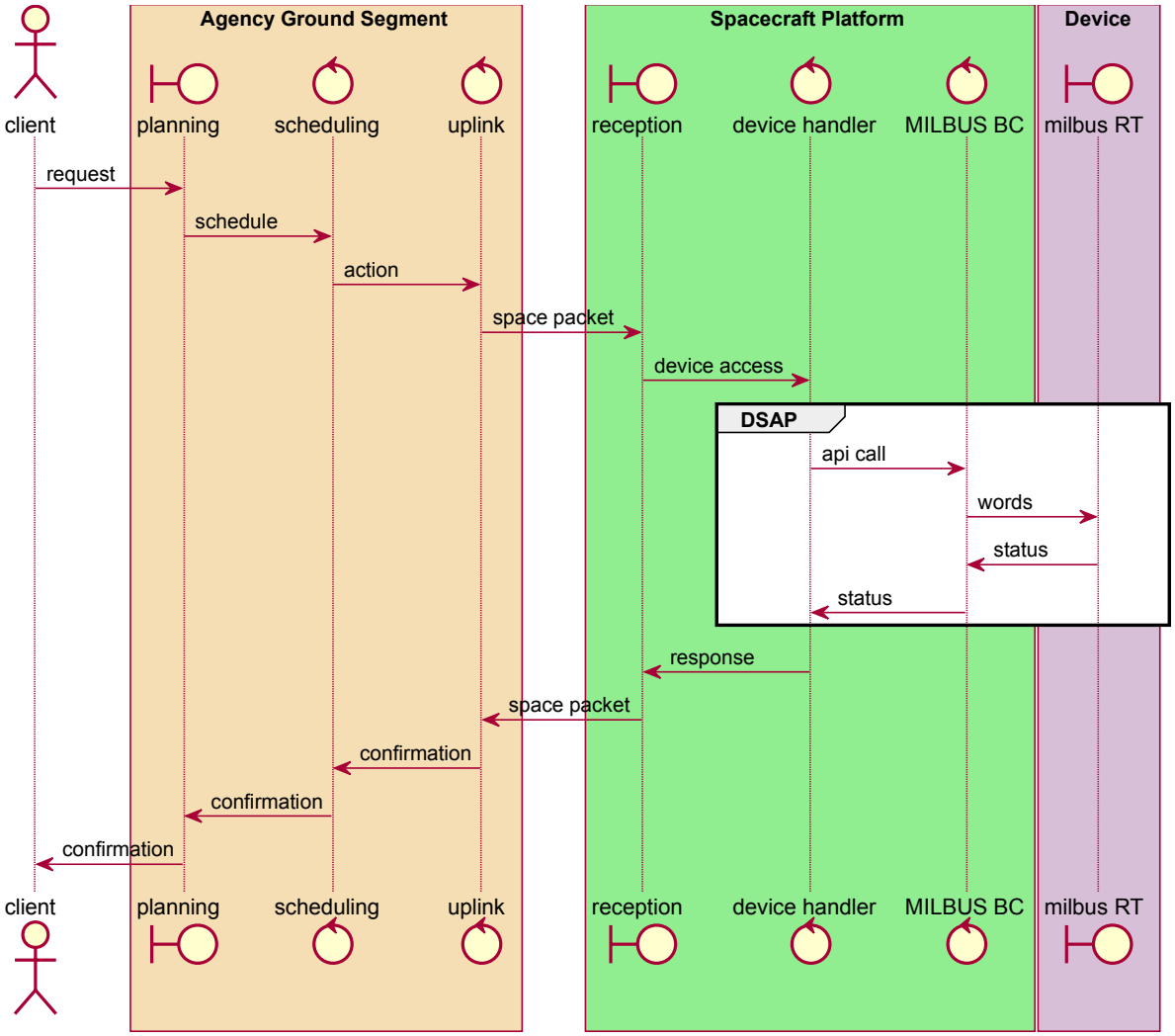

Figure 4-2: Sequence Diagram: Adjusting a Setting on a Device at the Request of the End User

The client request, or 'Action', in figure 4-2 could be 'Take a Picture'. There is no requirement at this point that users have to think in terms of 'command packet' or 'device access' data structures. It should be noted that this diagram represents both Use Case 1 and Case 2 of MOIMS integration, with the only difference being where the encoding of the MO request into/out of a space packet takes place.

In effect, a subset of the electronic device interface, as defined in the CCSDS EDS, is made available, via CCSDS MO Services, to the end user. The EDS and the Deployment

Description is, in fact, available to the MO during development of the device control software and as a part of operations preparation. This will be true whether this is a Case 1 or 2 deployment, or a Case 3 deployment in which more of the MO functionality has been developed and tested to the necessary levels to be migrated on board.

Of course, in most cases, onboard real-time functions, such as power or thermal control, or GNC, would not be directly managed by the end user, but instead by autonomous, real-time, onboard functions which themselves have configuration settings managed by the client. In such a case, the above diagram would be simply split into two parts for the communication of management settings between the end user and onboard thermal control, and real-time control settings between thermal control and the device. Such configurations would also have Case 1 and 2, or even 3, variants.

4.2 SPECIFYING INTERFACES

IEEE defines 'interface', '*To connect two or more components for the purpose of passing information from one to the other*'. The noun form, an interface, is a specification of how this is done, exactly what categories of data can be exchanged in what sequences, and ultimately the protocol stack used to communicate across the interface. At the top-most layer of an interface, all components have interfaces and behavior in both the application and functional senses. Interfaces also have bindings to a particular H/W port, technology, and protocol stack. Every protocol layer in the stack has PDUs, interactions on the wire, and behaviors at the protocol level and within protocol entities. The top-most protocol in the stack, at the binding interface, is the one that exhibits the behavior at that interface.

Between different programming languages, standards, middleware tooling, etc., there are a large variety of ways to formally specify an interface. Each such specification makes certain assumptions about what an interface is, in order to describe it. Table 4-1 summarizes these features. For the purposes of this document, these formalisms can be categorized according to the following set of properties:

- **Message Encoding**: How the data in the messages passing across the interface is represented in terms of octets and bits. It can be:
  - Implicit: left to a tool to work out according to a set of defined rules;
  - Explicit: specified as part of the interface;
  - Optional: a choice of either of the above.
- **Cardinality**: The number of components connected. Can be 1:1, 1:Many, or Many:Many.
- **Directions**: From which of the ends of the interface message groups can be initiated. Can be one-way or two-way.
- **Message Grouping**: Whether the messages are always entirely standalone or can be implicitly grouped together by some underlying mechanism. It can be:
  - None: each message is standalone;
  - Paired: each message can have a single reply;
  - Patterned: messages can be organized into arbitrarily large groups according to a set of predefined interaction patterns.

Table 4-1: Interface Features

Formalism	Terminology	Encoding	Cardinality	Directions	Message Grouping
C family[14]	set of functions	implicit[15]	1:many	one-way	paired[16]
PUS[17]	service	explicit	many:many	two-way	none
RASDS[18]	port	explicit	many:many	two-way	none specified
EDS	interface	optional	unspecified	two-way	paired
MAL	service	implicit	1:1[19]	one-way	patterned

4.3 DETAILED COMPARATIVE ANALYSIS

Figure 4-3 shows a graphical comparison of the different kinds of structures that appear in the XML schema for EDS and MAL. Elements that are semantically similar are shown at the same layer. Areas marked with 'A' are abstract, hiding further detail.

[14] The programming language C is included because of its historical influence on other languages like C++ and Java, on middleware targeted at those languages such as CORBA, RMI, and ESA's SMP2, and also on formalisms designed largely to generate code in such languages, such as UML and SysML. Some of those have an explicit 'interface' construct corresponding to a set of functions.
[15] The compiler selects the actual layout of data in memory, according to properties of the target CPU.
[16] The return value of a function is inherently associated with the corresponding call.
[17] ESA Packet Utilization Standard, ECSS-E-ST-70-41C (reference [7]).
[18] Reference Architecture for Space Data Systems (RASDS), CCSDS 311.0-M-1 (reference [4]).
[19] Except a PubSub operation, which has 3 classes of participants, including any number of subscribers.

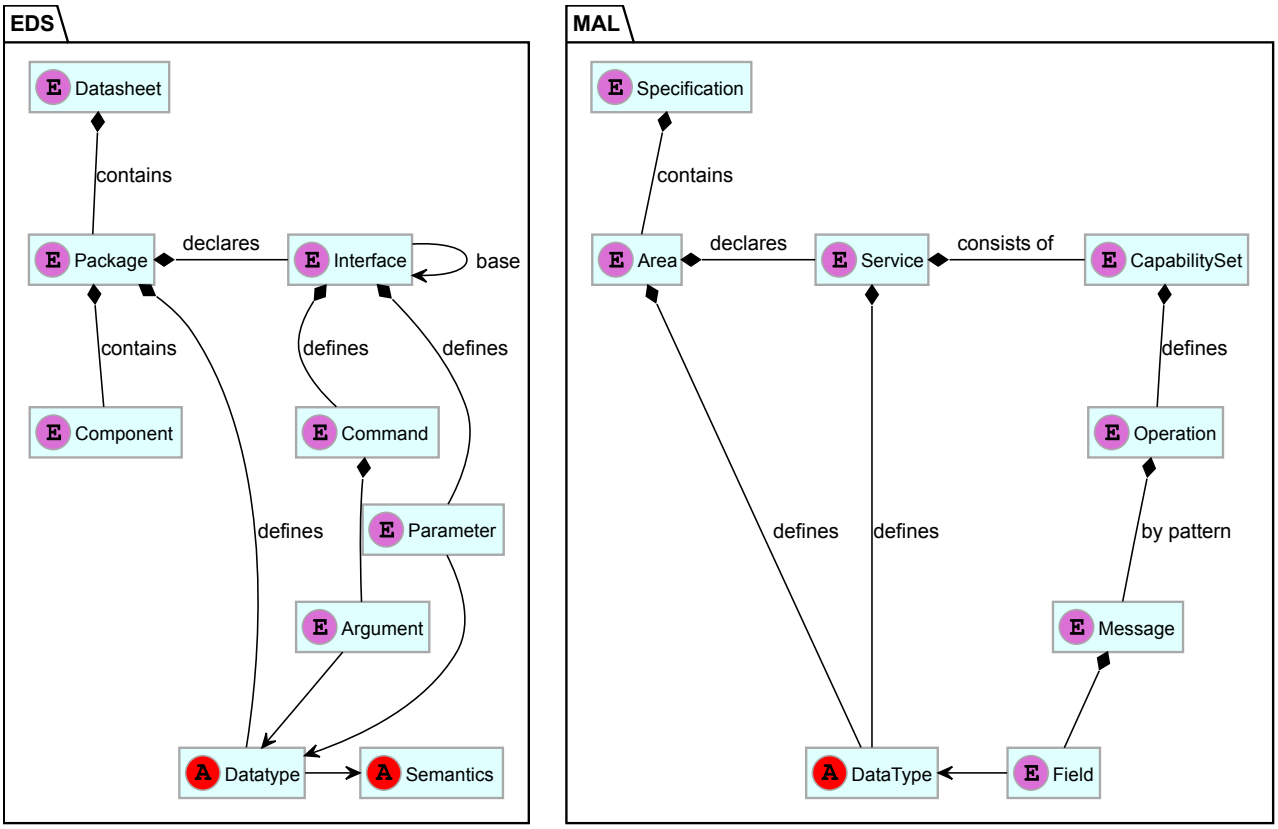

Figure 4-3: XML Structure of EDS and MAL

When an EDS is used to define an interface:

- a device has one datasheet;
- a datasheet contains several namespaces;
- namespaces define data types, declare interfaces, and contain components;
- interfaces have protocol stack binding signatures, use inheritance, and contain parameters and commands;
- commands have arguments and use interaction patterns (see figure 4-5);
- arguments and parameters have a data type and semantics, which define their meaning by referencing an associated ontology (reference [6]);
- a component can specify behavioral mappings and constraints on and between interfaces.

When MAL is used to define a service:

- a specification covers several areas;
- areas define data types and services;

– services have interfaces that are bound in a given deployment via the data representation and transport bindings, but this is not explicit in the model;

– a service can define data types, and has optional capability sets, each of which defines a set of related operations;

– each operation has a sequence of messages, organized by interaction pattern (see figure 4-4);

– each message has a number of named fields;

– each field has a data type.

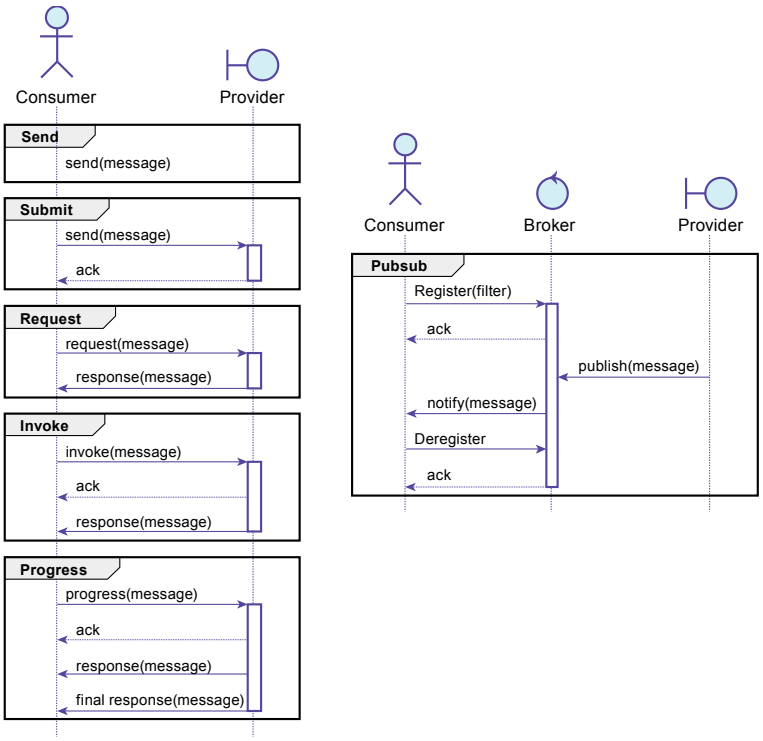

Figure 4-4: MAL Interaction Patterns

NOTE – In MO, any operation must follow one of the six supported MAL interaction patterns, governing which messages must be specified to define the operation.

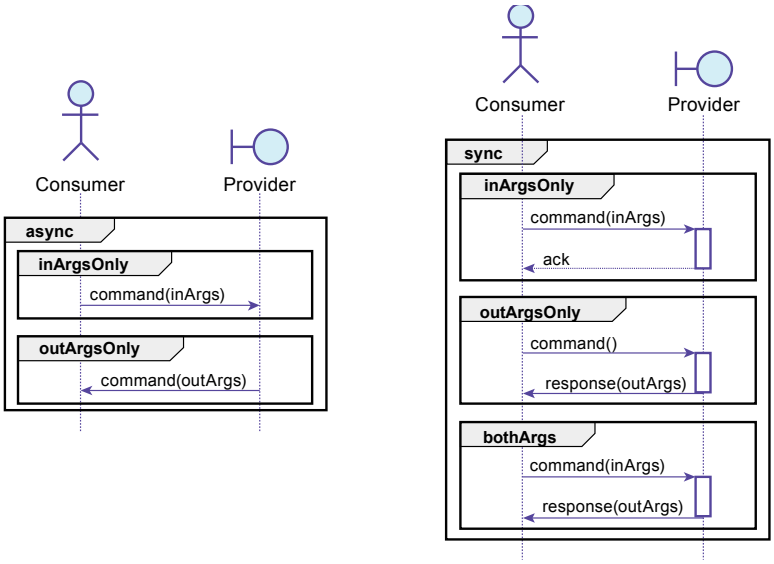

Figure 4-5: EDS Interaction Patterns

NOTE – EDS has five distinct interaction patterns for commands, based on whether the command mode is async or sync, and whether it has only input arguments, only output arguments, or both.

Four of the EDS interaction patterns map directly to the MAL patterns **Send**, **Submit**, and **Request**. The other, async + outArgsOnly, corresponds to a partial PubSub pattern with no filter or broker.

In many ways, the biggest difference between EDS and MAL interface descriptions is that EDS provides direct means for mapping to concrete realizations of interfaces and operations, whereas MAL employs abstract definitions that must be mapped in a separate step, through encoding and protocol bindings, to a concrete realization. In EDS, the data structures are bounded by actual device, component, and subnet characteristics. In MAL, the data structures are bounded by the encodings that are used. EDS also provides the means to describe software and hardware deployments and composition, deploy software on computing platforms, and produce concrete descriptions of specific deployments.

5 SOIS EDS AND MO SERVICES INTEGRATION

5.1 OVERVIEW

SOIS EDS and MO Services are two independent technologies that can be integrated together. The different possibilities of their integration are captured in this section.

Three different cases of MO integration with SOIS onboard functions are presented in detail:

a) Case 1: SOIS device interfaces, subnets, and services are deployed on board with the usual real-time flight software for GNC, M&C, C&DH, FDIR, power & thermal management, and the usual resource constraints. MO is only on ground, with typical TT&C interfaces between flight and ground. This is a traditional case without any overlapping areas of interfaces between MO and SOIS on board. SOIS EDS may be used on the ground as part of development and operations preparation. MOIMS XTCE may be used to describe the characteristics of the TT&C flows across the space link.

b) Case 2: The same SOIS services and RT FSW is deployed on board, but in this case, MO Proxy interfaces are implemented to flight SW standards and provided on board for a subset of services connecting to the usual RT FSW. This case uses MAL message exchanges over TT&C to the MAL Proxy on board. This is a transitional deployment in which MO Service interfaces are integrated with the onboard environment, but the rest of the onboard environment and services continue to operate in a normal fashion. EDS and XTCE may be used as in Case 1.

c) Case 3: The same underlying SOIS services and RT FSW on board, but more of the MOIMS MAL-based services and frameworks are implemented to flight SW standards and adapted to the RT environment and migrated on board as appropriate. Real-time spacecraft operations remain under direct control of the RTOS, but MO management and planning of these services may be migrated on board as well. MAL message exchanges are done over TT&C and over the spacecraft message bus. Some devices may also have ‘MAL native’ interfaces. This is an integrated situation in which MO interfaces are adapted and operated in real time in the onboard environment. MO Services, in this configuration, could also be implemented on a separate co-processor.

5.2 MAPPING BETWEEN SOIS EDS AND A GENERATED BESPOKE MO SERVICE

An interface specified in EDS can be mapped to MAL by the following algorithm:

a) Within the EDS datasheet:

   1) each Parameter X is replaced with the equivalent list of getX, setX, and/or updateX commands, according to the read-only and mode attributes;

   2) any types defined inline are replaced with explicit named type definitions.

b) A MAL *Specification* corresponding to the EDS *Datasheet* is generated.

c) For each EDS *Namespace* involved, a corresponding MAL *Area* is created.

d) For each EDS *Datatype* involved, a corresponding MAL *Datatype* is referenced or created.

e) A MAL *Service* description corresponding to the instantiated *Interface* specification as provided by a particular component is defined.

f) A MAL *Capability Set* for each Interface Specification involved in defining that interface is defined.

NOTE – Grouping of interfaces into capability sets depends on which interfaces are stand-alone meaningful and which ones can be grouped together.

g) A MAL *Operation* for each EDS *Command* is defined, with the interaction pattern set according to:

   1) the value of the mode attribute;

   2) the mode attributes of all arguments to the command.

h) A MAL *Message* for each slot in the selected interaction pattern is created.

i) A MAL *Field* for each input or output argument of the command, using the matching datatype is created.

The result of this will be the equivalent description of a *bespoke* MO Service. Such a service could be:

– made available to ground directly;

– consumed by an autonomous device management application that is in turn made available to higher-level management and configuration services;

– consumed by the standard MO M&CS Action and Parameter service.

5.3 USING MO M&CS ACTION AND PARAMETER SERVICES WITH EDS

This subsection describes how, as an example, the MO Action service could be directly implemented in terms of a device described by an EDS. Two cases are considered, depending on whether MO is supported on board (Case 2) or not (Case 1).

5.3.1 CASE 1 DEPLOYMENT

In Case 1, for its electronic device, the Client provides an EDS, which, in the EDS Device Access interface, describes the set of parameters and commands supported by the device. This allows establishing a logical link for MO to communicate to the Device.

To implement that logical link, MO establishes a consumer link to the Action Service of the MO ground segment using a prearranged domain id, for example, `mySc.payload.myDevice.prime`.

The MO client sends an action '`configureMode(STANDBY)`' using the MO Action Consumer API. The action service in the ground segment interfaces to the MO Adapter, which encodes those messages using a spacecraft compliant space packet encoding and sends them over the space link as a CCSDS packets inside TC space data link frames. This physically sends the space packets up to the satellite over a standard space link. The MO adapter encodes the payload PDU as specified in the client EDS. The space packet and space link encoding would benefit from using SOIS EDS specified encodings or XTCE, but have traditionally been defined in MCS unique formats.

On board, the implementation that responds at the action service layer is the Device Handler. This, when it receives the translated MAL-level message, transferred as Space Packets, uses the subnetwork interface implemented using the EDS datasheet to physically talk to the device and to send the command on the MILBus, SpaceWire, or other link and determine its success or failure. This action status data is then relayed back to the ground in space packets. The device datasheet contains all the information required to specify the behavior of a Device Handler using this model, meaning that part of the implementation can be automatically generated.

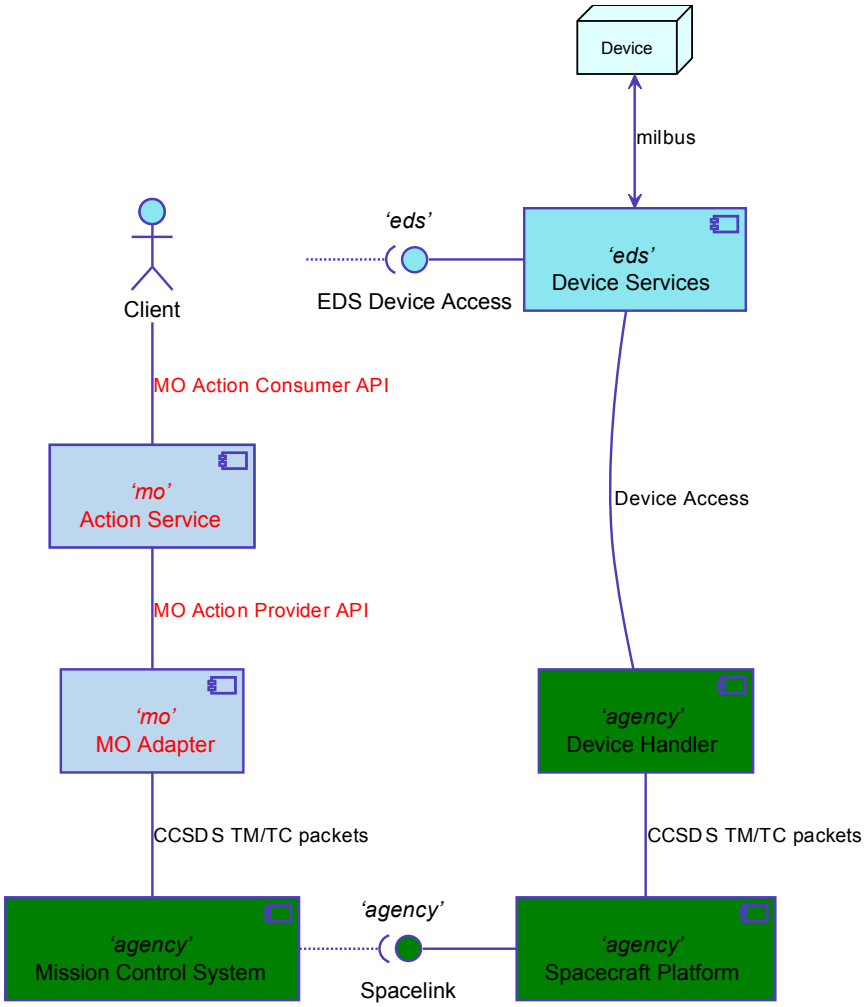

Figure 5-1: MO Action Service Implemented Using the Action Provider API On Ground (Case 1 Example)

NOTE – Usage of the parameter service would be similar.

5.3.2 CASE 2 DEPLOYMENT

For Case 2, a straightforward adaptation of the above approach works in the case in which an MO Proxy interface is supported on board. The Device Handler is implemented using the same techniques as the rest of the spacecraft platform, and supports standard TM/TC packet interfaces, as defined in a spacecraft database. The ground-side MO implementation translates MO calls and uses an MO specified encoding and a transport binding that maps them into SPP and standard TT&C protocols. On board, an MO Proxy extracts the MO calls from the TT&C transport and maps them to the legacy onboard architecture. TM/TC encoded action provider calls into the standard TM/TC Packet calls that would have been made using the onboard approach. In other words, the MO Proxy on board implements an MO Services translation, allowing communication with a legacy onboard architecture using TM/TC Packets. In this case, not only the Device Handler, but the relevant portions of the

spacecraft database and the MO Adapter, would be able to be generated from, and/or verified against, the device datasheet.

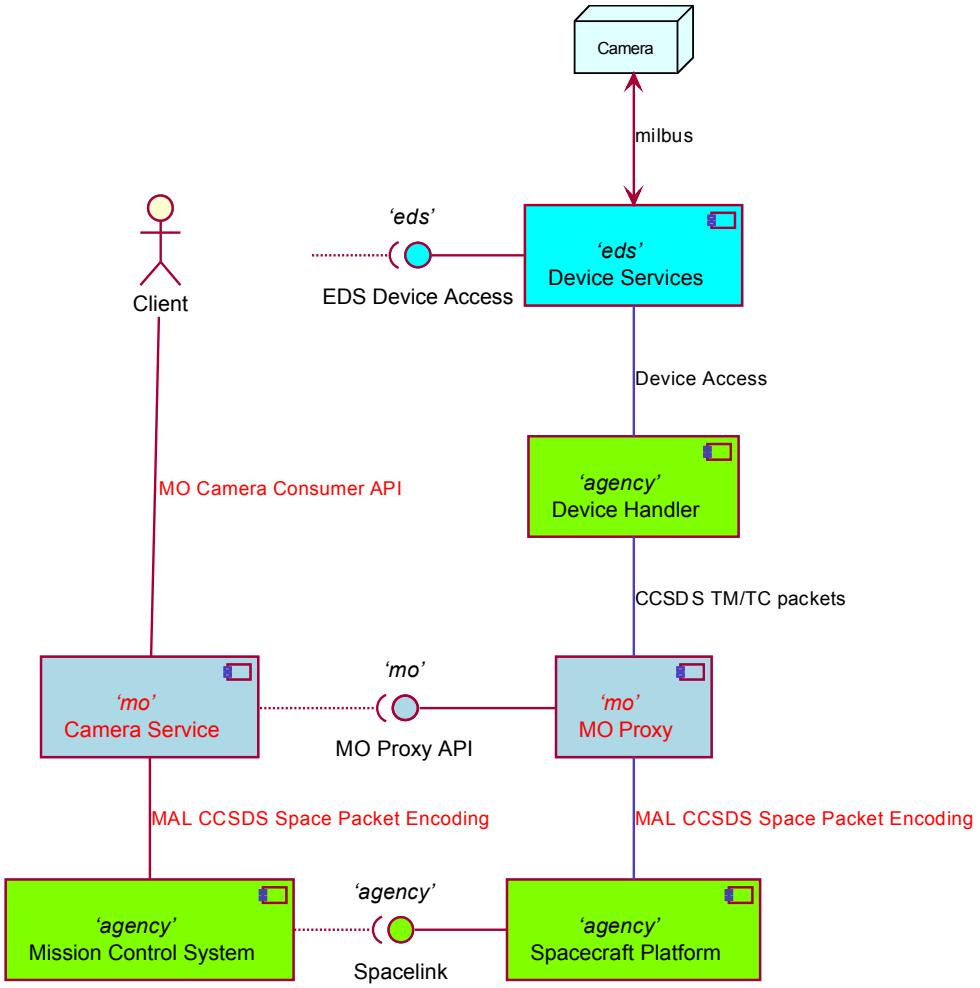

Figure 5-2: MO Action Service Implemented Using the Action Provider API On Ground (Case 2)

NOTE – Again, usage of the parameter service would be similar.

It is necessary to point out that this use of MAL CCSDS Packet Encoding across the space link (in both directions) comes with an overhead as shown in *Mission Operations—MAL Space Packet Transport Binding and Binary Encoding* (reference [10]). Tables 5-1 and 5-2, adapted from reference [10], show the space packet primary and secondary header formats used in MAL Space Packet. Figure 5-3 shows the mapping of the MAL abstract message header to the extended version of the SPP secondary header that the MAL uses.

As shown in table 5-1, the native SPP header is six (6) bytes long. As shown in table 5-2, the MAL version of the SPP secondary header is a minimum of 37 octets, not counting another 4, optional, variable-length fields that could be a minimum of 20 more octets but have a potential maximum length of thousands of bytes (each variable-length field has a 32-bit

length field). As a result, the minimum length translation of a one-byte length MAL message (if such a small MAL message existed) using the SPP and binary encoding is at least 6 times longer than a minimum-length encoding of a one-byte command in the SPP user data field.

Of course, the user data fields are seldom one byte, many traditional commanding approaches use fields of a few bytes up to 10s of bytes. MAL messages, since the data fields themselves may also include variable-length strings, will themselves be lengthy. For Cases 2 and 3, the effect of this overhead on uplink and downlink bandwidth must be considered when any such deployment being contemplated. It should also be noted that as bandwidth increases, the trade between interoperability and message overhead should be re-evaluated.

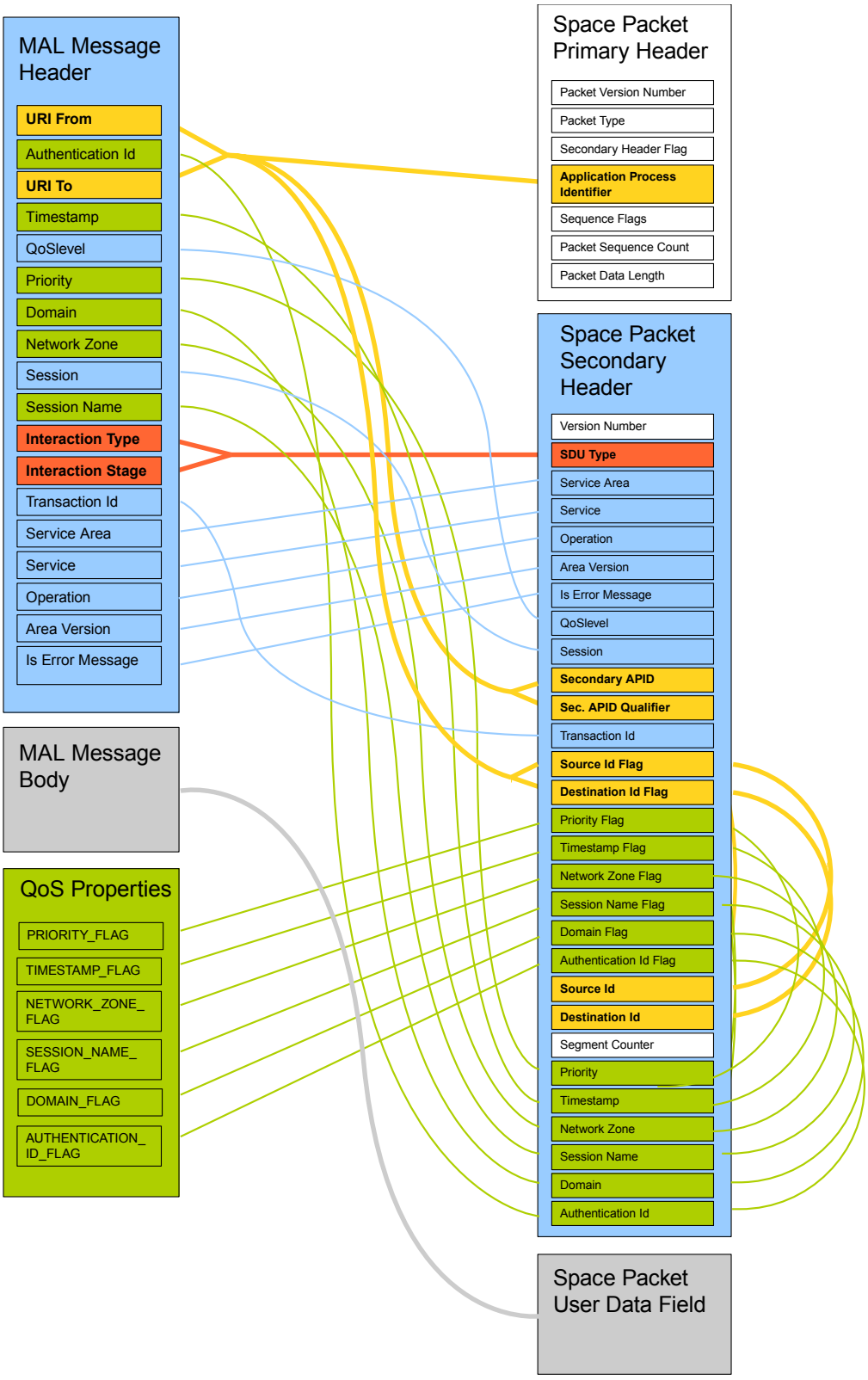

Figure 5-3: MAL Message Mapping to Space Packet

Table 5-1: Space Packet Primary Header Format

Packet Version Number	Packet Type	Secondary Header Flag	Application Process Identifier	Sequence Flags	Packet Sequence Count	Packet Data Length
Binary value (3 bits)	Binary value (1 bit)	Binary value (1 bit)	Unsigned 11-bit integer (11 bits)	Binary value (2 bits)	Unsigned 14-bit integer (14 bits)	Unsigned 16-bit integer (16 bits)
Always equal to '000'		Always equal to '1'				

Table 5-2: MAL Space Packet Secondary Header Format

Version Number	SDU Type	Service Area	Service	Operation	Area Version	Is Error Message	QoS Level	Session	Secondary APID	Secondary APID Qualifier	Transaction Id	Source Id Flag	Destination Id Flag	Priority Flag	Timestamp Flag
Binary value (3 bits)	Unsigned integer (5 bits)	Unsigned integer (16 bits)	Unsigned integer (16 bits)	Unsigned integer (16 bits)	Unsigned integer (8 bits)	Binary value (1 bit)	Unsigned integer (2 bits)	Unsigned integer (2 bits)	Unsigned integer (11 bits)	Unsigned integer (16 bits)	Signed integer (64 bits)	Binary value (1 bit)	Binary value (1 bit)	Binary value (1 bit)	Binary value (1 bit)

Network Zone Flag	Session Name Flag	Domain Flag	Authentication Id Flag	Source Id	Destination Id	Segment Counter	Priority	Timestamp	Network Zone	Session Name	Domain	Authentication Id
Binary value (1 bit)	Binary value (1 bit)	Binary value (1 bit)	Binary value (1 bit)	Unsigned integer (8 bits)	Unsigned integer (8 bits)	Unsigned integer (32 bits)	Unsigned integer (n*8 bits)	Time (n*8 bits)	Identifier (n*8 bits)	Identifier (n*8 bits)	List <Identifier> (n*8 bits)	Blob (n*8 bits)
				If 'Source Id Flag' is '1'	If 'Destination Id Flag' is '1'	If 'Sequence Flags' is '00', '01', or '10'	If 'Priority Flag' is '1'	If 'Timestamp Flag' is '1'	If 'Network Zone Flag' is '1'	If 'Session Name Flag' is '1'	If 'Domain Flag' is '1'	If 'Authentication Id Flag' is '1'

5.4 CASE 3B DEPLOYMENT: USING DEVICE-CATEGORY MO SERVICES WITH EDS

The final implementation option considered is for MO Services to be defined with semantically meaningful data for a specific category of device (example: Camera service, GPS service, etc.), with operations logically necessary for that category of device, independent of the actual vendor and model. This is a Case 3 example in which MO is integrated on board.

For this example, MO is directly supported on board with MAL-based services and framework adapted to work in the real-time environment. This would typically mean implementing the MO Services and framework to strict Class A/B standards and developing an onboard MAL encoding and transport binding suitable for the characteristics of the real-time OS and flight environment. In this case, devices might be interfaced with as in Cases 1 & 2, or devices could directly implement MAL native interfaces instead of relying on an MO compliant Device Handler to take in MAL-level messages and convert them to the SOIS onboard devices services. Figure 5-4 shows the MO Action Service used to invoke the onboard MO compliant Device Handler behavior.

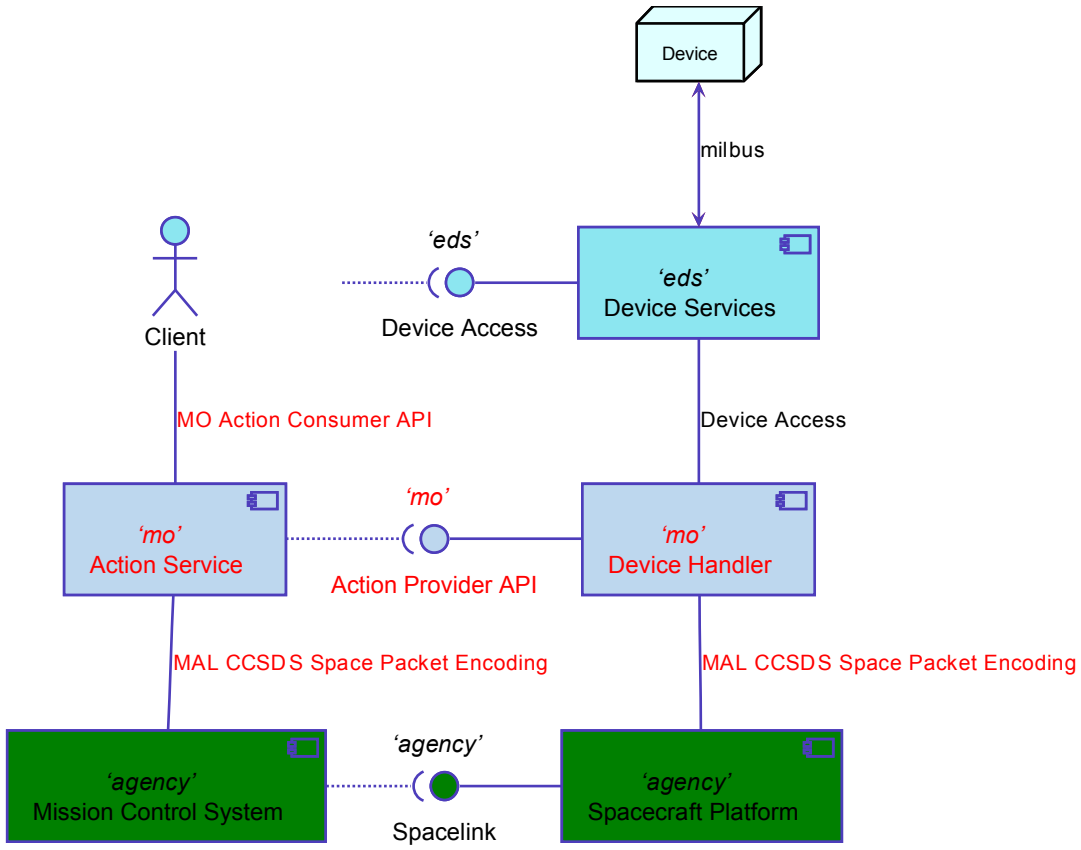

Figure 5-4: MO Action Service Implemented Using the Action Provider API On Board (Case 3)

Figure 5-5 shows a variant of this in which a bespoke, MO compliant, Camera service is used to control the behavior of a bespoke, onboard, MO compliant Device Handler. In this example, treating the camera as an MO device, it is possible to specify an MO Camera service that has semantically meaningful operations such as:

– take a picture;
– preview picture;
– zoom in;
– etc.

These operations are common to all cameras, independent of the vendor of the device (e.g., Sony, Panasonic, etc.). The exchange of semantically meaningful information from the ground to the spacecraft can be done using such an MO compliant Camera service.

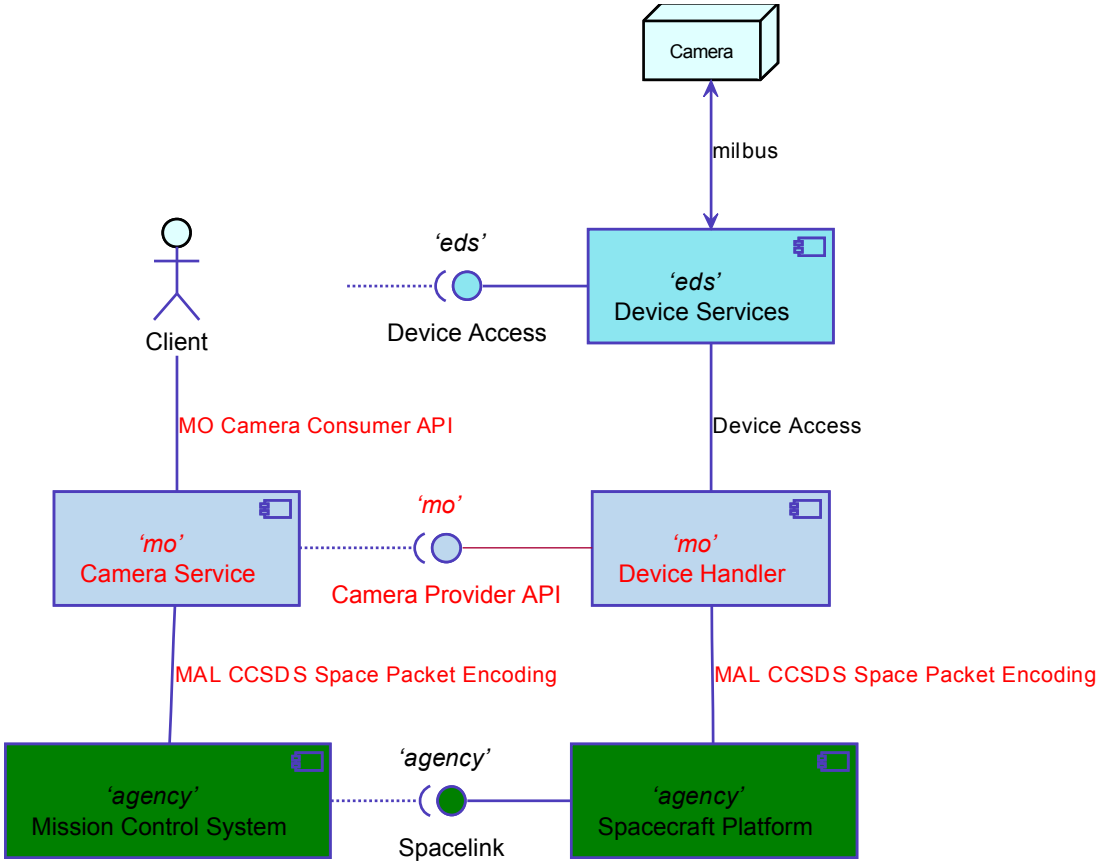

Figure 5-5: Bespoke MO Camera Service Implemented Using the Camera Provider API On Board

The implementation is very similar to that presented for a standardized MO Service, with the only difference being that instead of interacting with the generic Action service, the client would use the device-specific MO Camera service.

6 CONCLUSION

Overlaps and differences between MO and SOIS standards can best be viewed in terms of their use cases. For their respective use cases, each set of standards can bring significant value.

A SOIS EDS is intended to describe an existing software component or device without regard to implementation or architecture. A SEDS only describes the interface as built. It is up to software tooling to map that to a particular system implementation. The SEDS is the interoperability point that allows automation of device integration, and testing. With manufacturers providing a SEDS with each device and/or component, system integrators would be able to take those machine readable specifications and generate appropriate interfaces for their architectures avoiding the issues with paper specifications and Interface Control Documents (ICDs).

The scope of MO, on the other hand, is quite wide and encompasses in its broadest long-term scenario the end-to-end ground and space segment as a whole. MO standards are more focused on interoperability and code reuse, which can have significant costs benefits if adopted.

While CCSDS MOIMS and SOIS working groups have different scopes and problem spaces, there is some convergence in technical approach since both use an XML schema for their respective device and service specifications.

As described in Section 4, it should be evident that SOIS EDS and MOIMS MAL have a certain degree of overlap in terminology, especially as related to the concepts of services and interfaces.

However:

- That overlap is limited to perhaps 15-20 percent of the scope of each specification.
- The express purpose of the two sets of specifications are distinct, in which MO is about abstract definitions of services that must be mapped in a separate step, and EDS provides direct means for mapping to concrete realizations of interfaces and operations, describing software and hardware and concrete descriptions of specific deployments.
- Analyzing scenarios in which both MO Services and SOIS EDS interfaces were in use for Case 1, there is no need to translate between the two, as they operate on different levels of abstraction.
- For Case 2, there will be a need to translate directly from MO MAL messages using an onboard proxy. This has the advantage of adopting a standard service-oriented paradigm and makes minimal incursion into the onboard, real-time, and typically resource constrained environment, but at the cost of potentially significantly increasing uplink and downlink bandwidth.

- For Case 3, there is a need to have direct interface between MO Services and SOIS EDS interfaces. This has the Case 2 benefits, but also the Case 2 added costs, with an even greater impact to onboard resources.
- Just for service interfaces, translating between the two representations is, in any case, straightforward.

In considerations with respect to relation of SOIS and MO, it is important to distinguish between three use cases: ground-to-ground interfaces (Case 1), space-to-ground interfaces for typical M&CS and via a proxy on board (case 2) and a true integration of space and ground systems, including space-to-space interfaces and exposure of selected onboard interfaces to the ground (Case 3).

There is no need for any recommendation for Use Case 1 (MO on the ground only and SOIS on board). This use case has a clear border between the two set of specifications and no overlap.

In Use Case 1 there is no need to translate between the two at the service layer, as MO Services and SOIS EDS operate on different levels of abstraction. It would be the function of the MO Adaptor to map the abstracted MO interface to the concrete deployment interfaces described by an EDS in addition to including the transport interfaces.

In Use Case 2, there is little to no overlap, as this maps very well to today's current practices of having a dedicated part of onboard software dealing with M&C interfaces to the ground by means of TM/TC. In this use case, mainly the MO M&C services would add the value of cross-agency interoperability for typical operability requirements while having no major impact on the architecture of the onboard software and changes to current practices. In resource constrained environments, as it is typically the case on board a spacecraft, the implementation of the MO M&C and other proxied MO Services does not necessarily need to follow the conceptually separated layered architectures of Transport/MAL/MO Application Services. The creation of MO binding standards to typical communication protocols for S2G communication, such as SPP, would represent very little change compared to what is done today on the S2G link.

Hence the considerations about the relation and overlap of SOIS and MO are mainly for the long term vision of the 3rd use case, in which MO Services are supported natively on board for selected platform management functions (e.g., Power Management, Data Store Management) as well as for selected devices as Device Access and Control services (e.g., Star Camera service).

For Use Case 3, the following notes are to be taken into consideration:

- MO Service specifications are specified in a deployment, communication, and implementation agnostic manner and form an operational architecture with standard interaction patterns. For MO, the realization for a particular concrete space mission (a deployment) is achieved via an intermediated technology mapping to a

communication and implementation technology. MO Service specifications bring the benefits of common operational interfaces and implementation portability.

– EDS provides direct means for specifying the concrete realizations of interfaces and operations, describing software and hardware and concrete descriptions of specific deployments. The SOIS EDS provides a standard to describe the actual interfaces as built without expectations of any particular software or avionics architecture. The EDS itself is the only standard commonality point and requires software tooling to map components and devices to a chosen architecture. In Case 3, EDS would be used to map non-compliant components and services to the MO architecture. MO compliant components and services would implement any hardware-specific mappings internally.

– Both EDS and MO can make use of software tools to support the auto-generation of concrete executable code for a selected deployment. Hence the above described distinction is a conceptual difference. Practically, both EDS and a Service specification can result in auto-generated software through supporting tools.

– The current MO Service specifications do not cover the deployment considerations beyond mapping to implementation language and communication protocol. MO MAL can be extended in the future to incorporate the deployment aspects (such as hardware configurations, etc.).

For Case 2, there will be a need to translate directly from MO MAL messages using an onboard proxy. This has the advantage of adopting a standard service-oriented paradigm, with minimal incursion into the onboard, real-time, and typically resource-constrained environment, but with a disadvantage of potentially increasing uplink and downlink bandwidth.

For Case 3, there is a need to have a more direct interface between MO Services and SOIS EDS interfaces. This has the Case 2 benefits, but with an even greater impact to onboard software architecture. Also here, taking into account the typical characteristics of a resource constrained onboard environment, the conceptual layered architecture of MO framework can be merged at implementation/deployment layer to achieve efficiency.

Therefore it is recommended to perform a practical comparison of the two specifications, MAL and EDS, using the example of a concrete device, for example, a Star Camera, and focusing on Use Case 3. For this exercise, it is recommended to show end to end how the specification of the parameter and actions in an EDS relate to the MO M&C services. It shall at the same time assess how specification of Application Layer interfaces in EDS, for example, Mode Management, would compare to a corresponding MO Service specification of a Star Camera service.

The lessons learned from this analysis should be fed back into the corresponding specification development processes in order to improve areas in which either is lacking in capability or excessively complicated. For EDS, these could include:

– replacing the term ‘namespace’ with ‘area’, as that avoids confusion with XML namespaces; *This has been accepted, but with namespace replaced with package.*

– replacing the term ‘interface instance’ with ‘port’, for better compatibility with Universal Modelling Language (UML) 2.0, and avoiding the potential confusion between ‘interface definition’ and ‘interface instance’;

– replacing the ‘mode’ SYNC/ASYNC flag on parameters and commands with a Boolean value ‘oneway’, by analogy with Common Object Request Broker Architecture (CORBA); this avoids overloading the term ‘mode’, also used for arguments.

For MAL:

– It has types and data structures defined in the abstract. It gets to real syntax only when a binding is specified and several different are available, leading to potential interoperability challenges. Semantics are defined only when actual services are defined that use the MAL structures and interaction templates. At this point few of these services are fully defined.

– In order to be efficient for use in a real-time, onboard, environment-typical deployment stacks shall be specified in Magenta Books for recommending concrete bindings to SPP and sensible choices to optimize the communication throughput (e.g., fixed parts of message header are not transmitted, etc.) The Magenta Book should also outline how the conceptual layered architecture of the MO framework can be merged to meet the concerns of typically resource-constrained environments.

– For most of the resource-constrained space environments, a Case 2 style deployment is likely to be the most recommended approach, since it adds the aspects of inter-agency interoperable service-oriented architecture to the operations (standardized operations for basic functionality of M&C rather than reinventing it for each mission). At the same time, this approach has minimal impact on today’s common practices of separation of space and ground segments and architecture of the two segments.

– It should be noted, however, that with advancement of the onboard hardware and resources and in particular for more advanced spacecraft, relatively resource-rich, habitat or space station deployment would benefit from an interoperable service-based architecture of Use Case 3.

Full recommendations on the use of M&C services and how the component level Application Layer interfaces would map to corresponding MO Service specifications can only be made after the exercise. The bindings for flight have not yet been published, so they cannot be analyzed here.

ANNEX A

ABBREVIATIONS AND ACRONYMS

Term	Definition
API	application programming interface
BSW	basic software
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
CORBA	Common Object Request Broker Architecture
CSS	Cross Support Services
DACP	Device Abstraction Control Procedure
DoT	dictionary of terms
DSAP	Device Specific Access Protocol
ECSS	European Cooperation for Space Standardization
EDS	Electronic Data Sheet
EGSE	electronic ground support equipment
FDIR	fault detection, isolation, and recovery
FSW	flight software
ICD	interface control document
M&C	Monitoring and Control
M&CS	Monitoring and Control Services
MAL	Message Abstraction Layer
MCS	mission control system
MMU	mass memory unit
MO	Mission Operations
MOIMS	Mission Operations and Information Management Services
OBC	onboard computer
OS	operating system
OSI	Open Systems Interconnection
PCDU	power control and distribution unit
PDU	protocol data unit
PUS	Packet Utilization Standard
QoS	quality of service
RASDS	Reference Architecture for Space Data Systems
RIU	remote interface unit
RM	reconfiguation module
RT	remote terminal
RTU	remote terminal unit

Term	Definition
SA	sub-address
SEDS	SOIS EDS
SIS	Space Internetworking Service
SLE	Space Link Extension
SOA	service oriented architecture
SOIS	Spacecraft Onboard Interface Services
SPP	Space Packet Protocol
TC	Telecommand
TCP	Transmission Control Protocol
TM	telemetry
TSP	time and space partitioned
TT&C	telemetry, tracking, and commanding
TTRS	telemetry, telecommand, reconfiguration, and safeguard memory
UML	Universal Modelling Language
VM	virtual machine
XML	Extensible Markup Language
XTCE	XML Telemetric & Command Exchange