



CCSDS

The Consultative Committee for Space Data Systems

**MO SERVICES AND
SOIS ELECTRONIC
DATASHEETS**

DRAFT RECORD

CCSDS 870.10-Y-0

DRAFT YELLOW BOOK

May 2018

AUTHORITY

Issue:	Draft Record, Issue 0
Date:	May 2018
Location:	Washington, DC, USA

(WHEN THIS CCSDS RECORD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

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
E-mail: 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 e-mail 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 Federal Science Policy Office (BFSP0)/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.
- Chinese 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.
- 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.
- 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-0	MO Services and SOIS Electronic Datasheets, Draft Record, Issue 0	May 2018	Current draft

CONTENTS

<u>Section</u>	<u>Page</u>
1 INTRODUCTION.....	1-1
1.1 RATIONALE.....	1-1
1.2 REFERENCES	1-2
2 INTRODUCTION TO SOIS EDS.....	2-1
2.1 OVERVIEW	2-1
2.2 TERMINOLOGY	2-2
2.3 SOIS REFERENCE ARCHITECTURE.....	2-3
2.4 SIMPLE EXAMPLE OF DATASHEET USE	2-9
2.5 OTHER CCSDS SERVICES AND THE OSI MODEL.....	2-10
3 INTRODUCTION TO MO SERVICES.....	3-1
3.1 OVERVIEW	3-1
3.2 MO AND OTHER CCSDS SERVICES.....	3-3
4 ANALYSIS	4-1
4.1 AREAS OF OVERLAP BETWEEN THE STANDARDS	4-1
4.2 SPECIFYING INTERFACES	4-4
4.3 DETAILED COMPARATIVE ANALYSIS	4-5
5 SOIS EDS AND MO SERVICES INTEGRATION	5-1
5.1 OVERVIEW	5-1
5.2 MAPPING BETWEEN SOIS EDS AND A GENERATED BESPOKE MO SERVICE.....	5-1
5.3 USING MO M&CS ACTION AND PARAMETER SERVICES WITH EDS	5-3
5.4 USING DEVICE-CATEGORY MO SERVICES WITH EDS	5-5
6 RECOMMENDATIONS AND CONCLUSION	6-1
ANNEX A ABBREVIATIONS AND ACRONYMS.....	A-1
 <u>Figure</u>	
1-1 Two Hypothetical Missions Using CCSDS Standards.....	1-1
2-1 SOIS EDS Concept.....	2-1

CONTENTS (continued)

<u>Figure</u>	<u>Page</u>
2-2 Terminology Used	2-2
2-3 SOIS Reference Architecture	2-4
2-4 Device Services Details	2-6
2-5 Sample Realization of SOIS Reference Architecture	2-9
2-6 OSI Model	2-11
3-1 CCSDS MO Scope	3-1
3-2 Details of an MO Service	3-2
3-3 Transformation of MAL into Technology-Dependent Interface Specifications	3-3
3-4 MO, SOIS, and Other CCSDS Services	3-4
4-1 Binary interface to the Device Expressed as a CCSDS EDS	4-1
4-2 Sequence Diagram: Adjusting a Setting on a Device at the Request of the End User	4-2
4-3 XML Structure of EDS and MAL	4-5
4-4 MAL Interaction Patterns	4-6
4-5 EDS Interaction Patterns	4-7
5-1 MO Action Service Implemented Using the Action Provider API Onboard	5-3
5-2 MO Action Service Implemented Using the Action Provider API On Ground	5-4
5-3 Bespoke MO Camera Service Implemented Using the Camera Provider API Onboard	5-5

Table

2-1 OSI Layering of SOIS Reference Architecture	2-11
---	------

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 spaceflight, with the goal of enhancing governmental and commercial interoperability and cross support, while also reducing risk, development time, and project costs. Within that organization, working groups have been tasked with looking at different areas of 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 interfaces between the spacecraft and onboard electronic devices.

This division of responsibility can be illustrated by an example whereby a hypothetical client institution designs, builds, and is involved in the operation¹ of a simple onboard instrument on both ESA and NASA spacecraft.

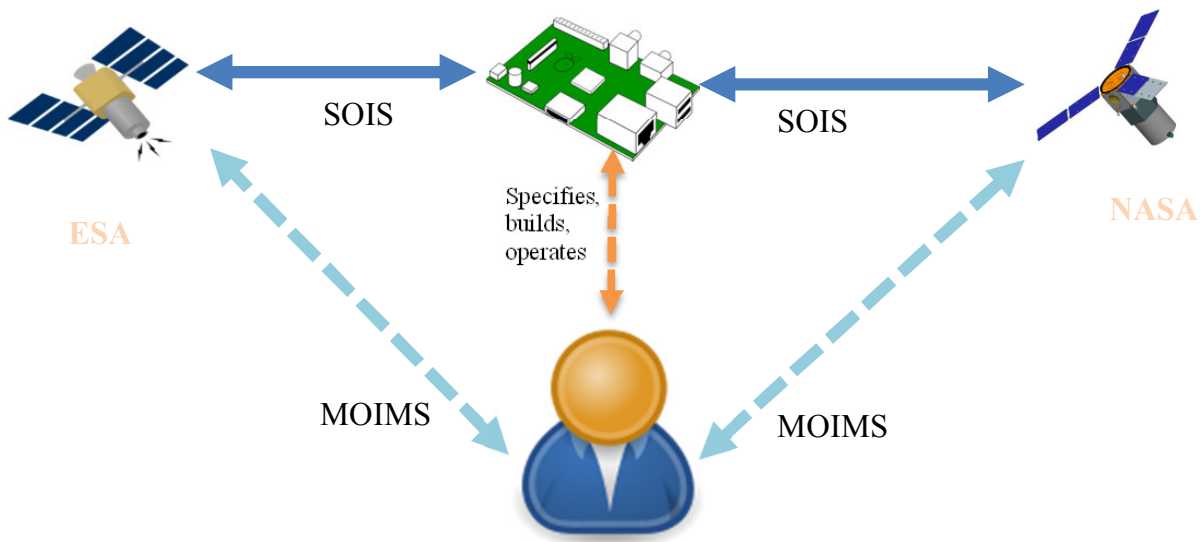


Figure 1-1: Two Hypothetical Missions Using CCSDS Standards

In such a case:

- SOIS is responsible for the information the client institution supplies to the spacecraft prime contractor, in order to integrate the instrument into the overall onboard platform;

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.

- MOIMS is responsible for the information the client institution supplies to the spacecraft prime contractor, and further to the operator (agency), in order to operate the instrument during the mission lifecycle.

If two copies of similar devices fly on different spacecraft built by different prime contractors under the responsibility of, and operated by, different agencies, then, if the CCSDS standards are applied in all cases, the result is minimal extra work for the client, for the prime, and for the operator. The same principles apply in more complex cases, where the client, designer, manufacturer, and operator are not the same, or where 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 standards would interact, with a view to ensuring those interactions are well-defined, well-understood, and unproblematic.

Following this introduction, this report:

- provides a brief overview of both sets of standards;
- performs an analysis of the relation between the two standards;
- describes how the two standards could interoperate on a mission, as in the above example;
- 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.

- [1] *Spacecraft Onboard Interface Services—XML Specification for Electronic Data Sheets*. Issue 2. Draft Recommendation for Space Data System Standards (Red Book), CCSDS 876.0-R-2. Washington, D.C.: CCSDS, June 2016.
- [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.
- [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 (XTCE)*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 660.0-G-1. Washington, D.C.: CCSDS, July 2006.
- [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.

2 INTRODUCTION TO SOIS EDS

2.1 OVERVIEW

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

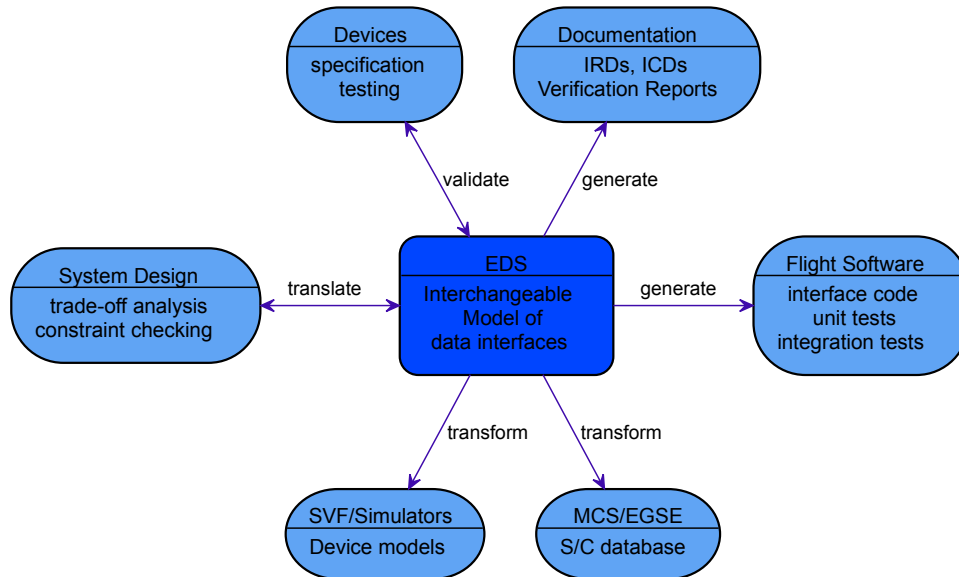


Figure 2-1: SOIS EDS Concept

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 where 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. Consequently, the SOIS Recommended Standard for Electronic Data Sheets (SEDS) takes the form of an eXtensible Markup Language (XML) schema designed for *tool interchange*, i.e., exchanging device data between two software systems.

2.2 TERMINOLOGY

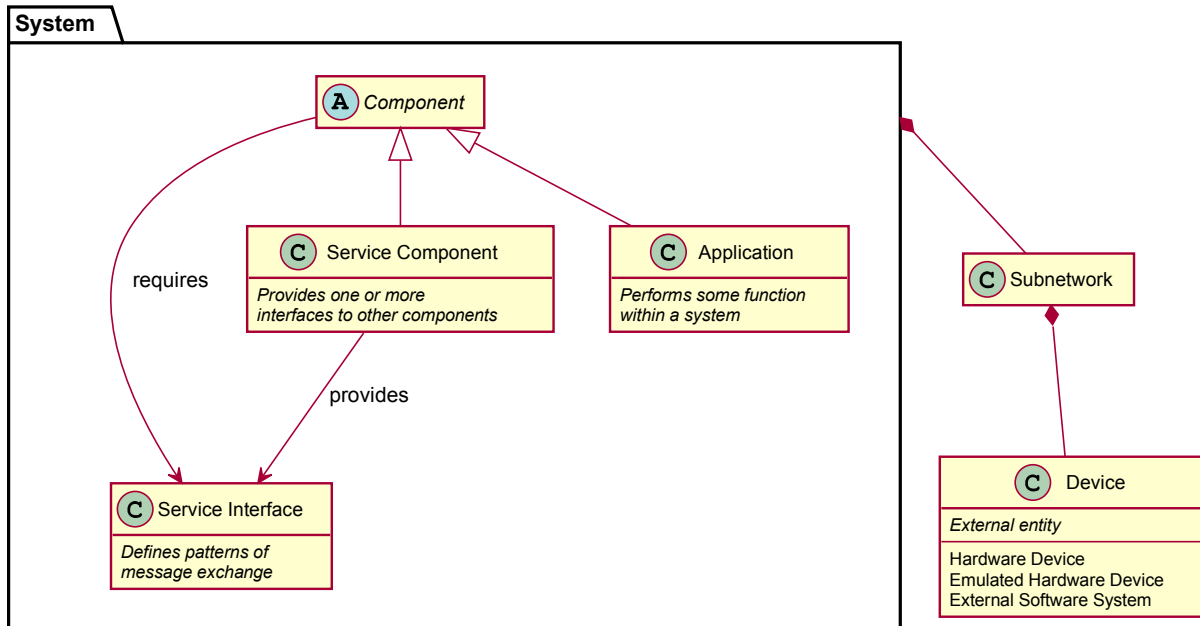


Figure 2-2: Terminology Used

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

Service Interface: Patterns of message exchange between components.

Application: A component that performs some function within a system.

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

Component: An application or service component. May require any number of service interfaces in order to operate.

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 belongs to a single subnetwork.

Subnetwork: A means of communication with a set of onboard devices.

Device: An entity external to the system that cannot naturally be described as operating in terms of service interfaces.

The goal of SOIS is to take an external entity, such as a device, and make it accessible from within a system. This is done by creating 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 *On-board System* (i.e., On-Board Computer [OBC] and FSW), and other on-board *Devices*.

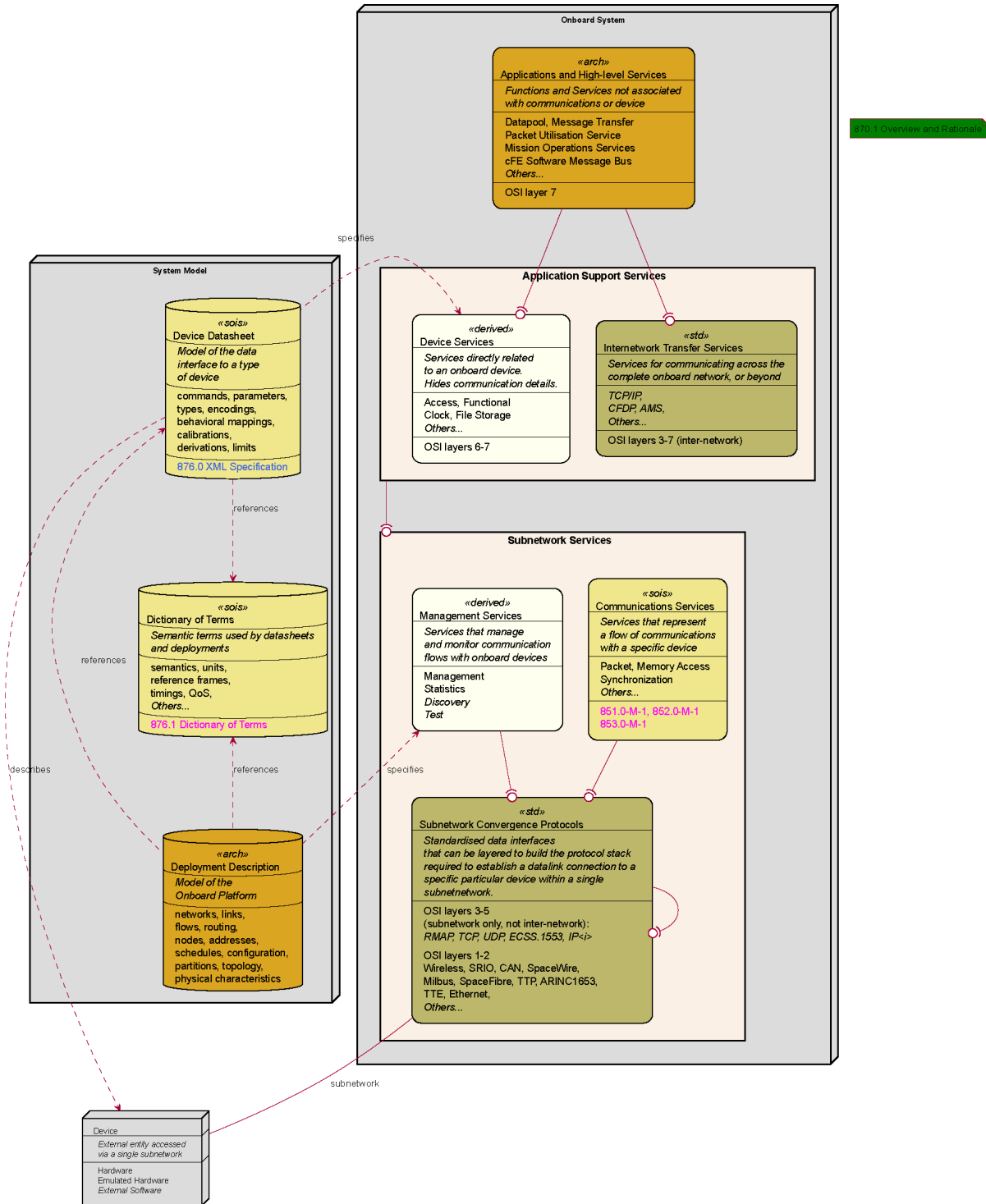


Figure 2-3: SOIS Reference Architecture

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;²
- <<derived>> indicates run-time elements that can be manually or automatically derived from the indicated specifying elements.

In that architecture, the On-board 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*; protocols such as CCSDS File Delivery Protocol (CFDP) that allow communication across different subnetworks.
- *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 also architecture-specific *Management Services*, which are logically derived from the parts of the System Model that specify how everything is connected together.

The SOIS EDS for a device forms part of the overall System Model, describing the details of one particular model of on-board device. This includes how the services it provides at the application layer can be implemented in terms of subnetwork-layer services.

Not shown is the SPACELINK interface to ground, as that is outside the scope of SOIS. It may be handled:

- at application level;
- by the Internetwork Transfer Services;
- as a dedicated subnetwork with its own SPACELINK subnetwork convergence protocols.

² From the point of view of the SOIS reference architecture, such elements may follow any standards, or none.

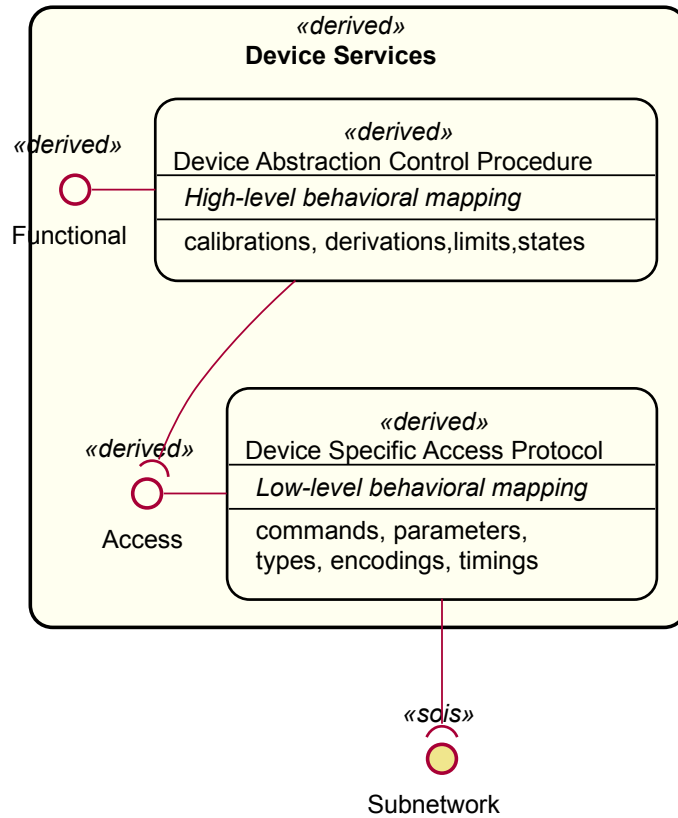


Figure 2-4: Device Services Det

Such a Device Datasheet defines the contents and interpretation of messages exchanged by applications and the device across the subnetwork layer and also the state machines describing message exchange protocols and device states. By specifying the device data interface 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.³ This validated datasheet can then be used as an input to the development and testing of those systems that interact with the device (i.e., the spacecraft FSW, system engineering database, checkout systems, etc.).

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

- a Device Specific Access Protocol (DSAP), consuming 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), consuming that access 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.

³ 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.

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.⁴

In the typical case, there will be a distinct single 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.⁵

A key characteristic of SEDS interfaces is that, while they support 2-way data exchange, they are partitioned into:

- parameters:⁶ messages coming from the device, plus those 2-way exchanges whose sole purpose is to get or set a discrete value on a device;
- commands: messages sent to a device, plus 2-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 split, which was taken from XML Telemetric & Command Exchange (XTCE) (reference [5]), reflects the way mission control systems 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; otherwise, the same device datasheet could not be used when the same hardware device is used for missions of different software architectures. Instead, the information required to do this is the concern of the code generation toolchain, which would either be architecture specific, or highly configurable. The result is code that uses the required coding standards and the native synchronization and communication APIs of the target architecture, not an additional universal wrapper layer.

All interfaces provided and required 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,⁷ and low-level binary interfaces containing data encoded in a specific way, as 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

⁴ It is common for there to be no requirement to perform calibration onboard. 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.

⁵ For example, for a GPS device there might be a standardised interface for positional data, supplemented by device-specific diagnostic and recovery interfaces.

⁶ SEDS parameters are commonly aggregates of primitive values; as such they arguably more resemble packets than the individual parameters of typical datapool-based software architectures.

⁷ This can take the form of an API, or a messaging interface following known systematic encoding rules.

mapping is typically static, i.e., done at system design time and recorded in a Deployment Description.⁸

As a consequence of the above, SEDS interfaces 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 where that interface was designed and implemented without knowledge of the SEDS or SOIS. In other words, SEDS allows wrapping an existing, well-tested, and certified legacy device as an interface of a 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.

⁸ In some cases, the mapping may be modifiable through 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

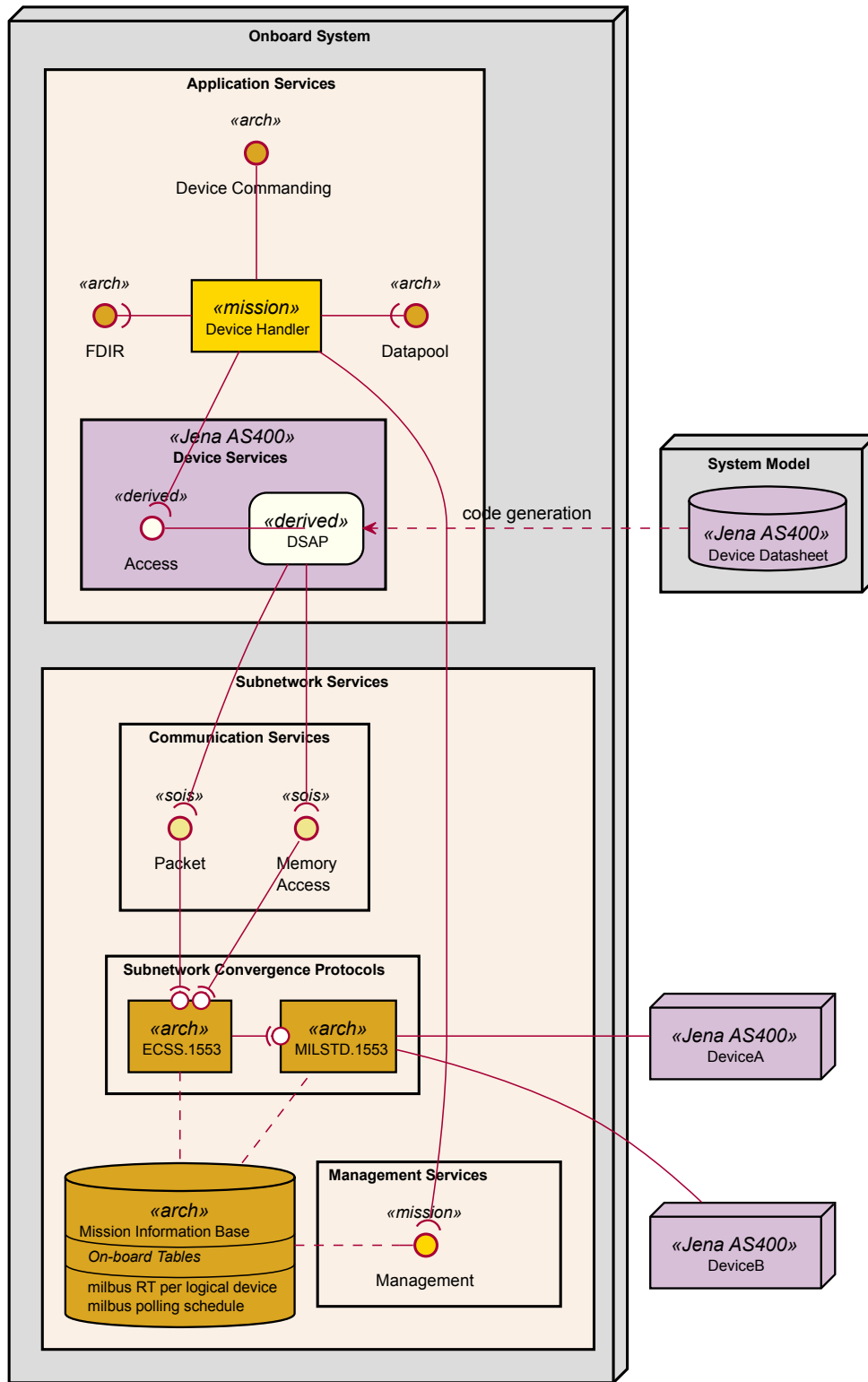


Figure 2-5: Sample Realization of SOIS Reference Architecture

The above diagram provides an example of how the SOIS Reference Architecture could be realized to manage a particular model of on-board device, in this case the Jena AS400 Star Tracker, of which prime and redundant instances are available as Remote Terminals (RTs) on a MILSTD.1553 bus, of which the On-board 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 Handle is the corresponding Access Interface (i.e., calls those functions) to implement:
 - commanding of the device, either from the ground, or on-board autonomy functions;
 - reading of telemetry from the device and storing it in a data pool, from where it can be reported to the ground and accessed by on-board autonomy functions;
 - Triggering Fault Detection, Isolation, and Recovery (FDIR) logic on failures.
- The Memory Access Interface is known 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 the corresponding architecture-specific implementation.
- The Packet Interface is known 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 the corresponding architecture-specific implementation.

In this simple example, there is no explicit Deployment Description; the corresponding data is available as a set of hand-written on-board 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, where logical communication between peer layers, 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.

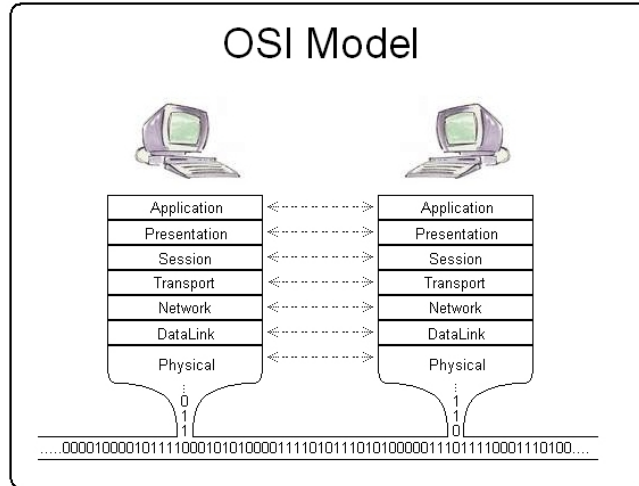


Figure 2-6: OSI Model

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

- CCSDS SPACELINK 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.
- Spacecraft Internetworking Service (SIS) standards govern application-to-application communication onboard 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 2-1: OSI Layering of SOIS Reference Architecture

Layer	Function	Per-Device	Cross-subnetwork	Per-Subnetwork
7	Application	Device Services		
6	Presentation			
Communications Service Interfaces				
5	Session		Internetwork Transfer Services	Subnetwork Convergence Protocols
4	Transport			
3	Network			
2	Datalink			
1	Physical			

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 the table above.

- A SOIS EDS specifies the interface to a device at the presentation layer (6).⁹ 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 layer¹⁰ 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.¹¹
- 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 5 to 3.

⁹ 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.

¹⁰ 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.

¹¹ 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.

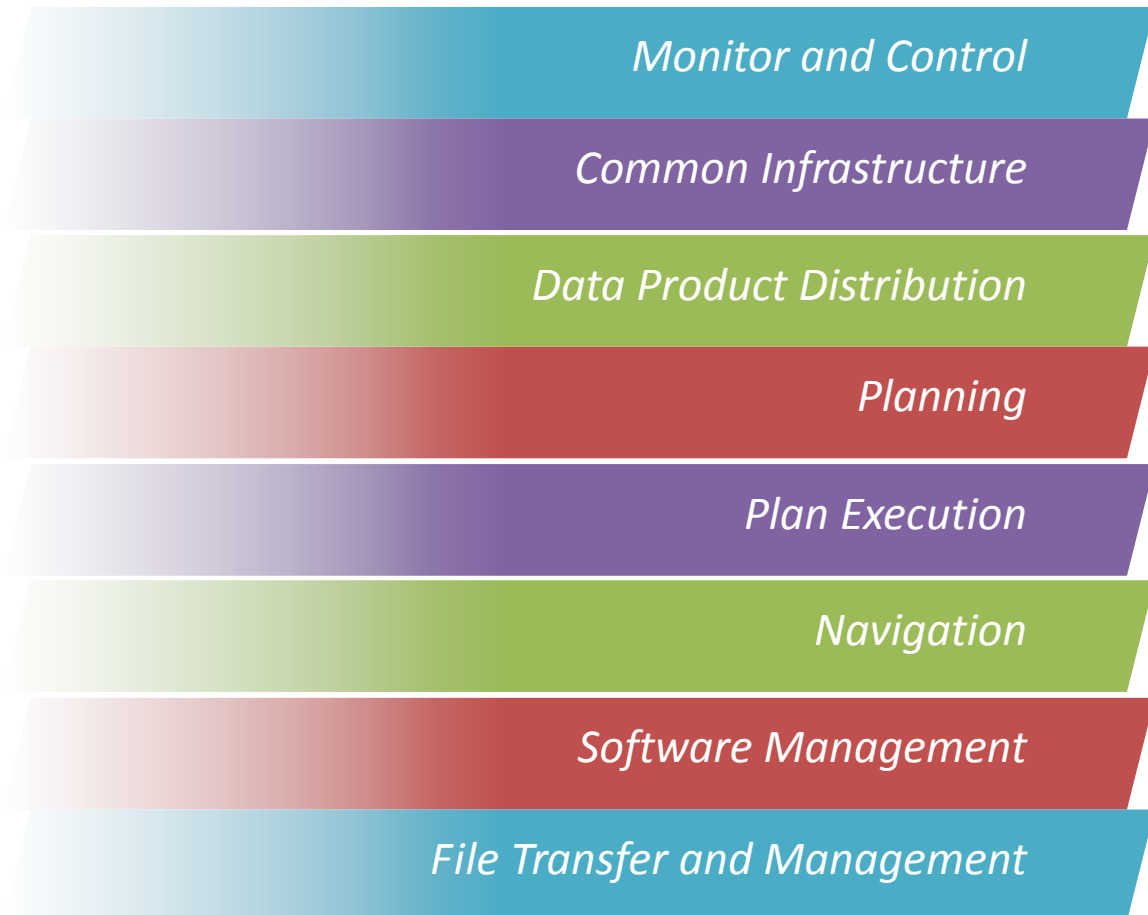



Figure 3-1: CCSDS MO Scope

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

To support these standardized services CCSDS has also defined an 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.

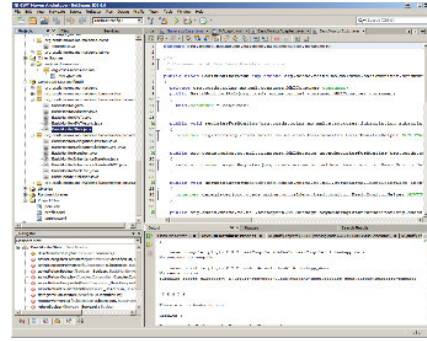
The MO concept is supported by the MO framework, which, in an abstract manner, allows the specification of an MO service, its operations, its data model, and the related generic facilities, such as archiving.

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/HTTP for a service deployed on the ground, Binary/SPP for a service deployed on the space-to-ground link, and Binary/SOIS for a service deployed onboard.


```

<oms:request name="GetCurrentTransactionList"
             number="108"
             comment="The getCurrentTransactionList consumer to obtain the
                    of parameter checks file"
             >
  <oms:response>
    <oms:request>
      <oms:type name="CheckForFileFilter" area="
    </oms:request>
    <oms:response>
      <oms:type name="CompleteTransaction" area="
    </oms:response>
  </oms:response>
</oms:request>

```



RECOMMENDED STANDARD FOR MISSION OPERATIONS COMMON OBJECT MODEL

Table 3-2: Archive Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
CODE	Archive	2	2	1
Interaction Point	Operation Identifier	Operation Number	Support In-Pillar	Capability Set
INVOKE	archive	1	Yes	
PROGRESS	query	2	Yes	1
INVOKE	erase	3	Yes	
REQUEST	save	4	No	2
SUBMIT	update	5	No	3
REQUEST	delete	6	No	4

3.4.2 COMMENT SERVICE USAGE

3.4.2.1 For each stored object, an 'ObjectStored' event may be published to the event service.

3.4.2.2 For each updated object, an 'ObjectUpdated' event may be published to the event service.

3.4.2.3 For each deleted object, an 'ObjectDeleted' event may be published to the event service.

3.4.2.4 The source link of the generated event shall link to the object being stored/updated/deleted.

3.4.2.5 Archive service events shall be persisted identically in order not to trigger an infinite event loop.

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.

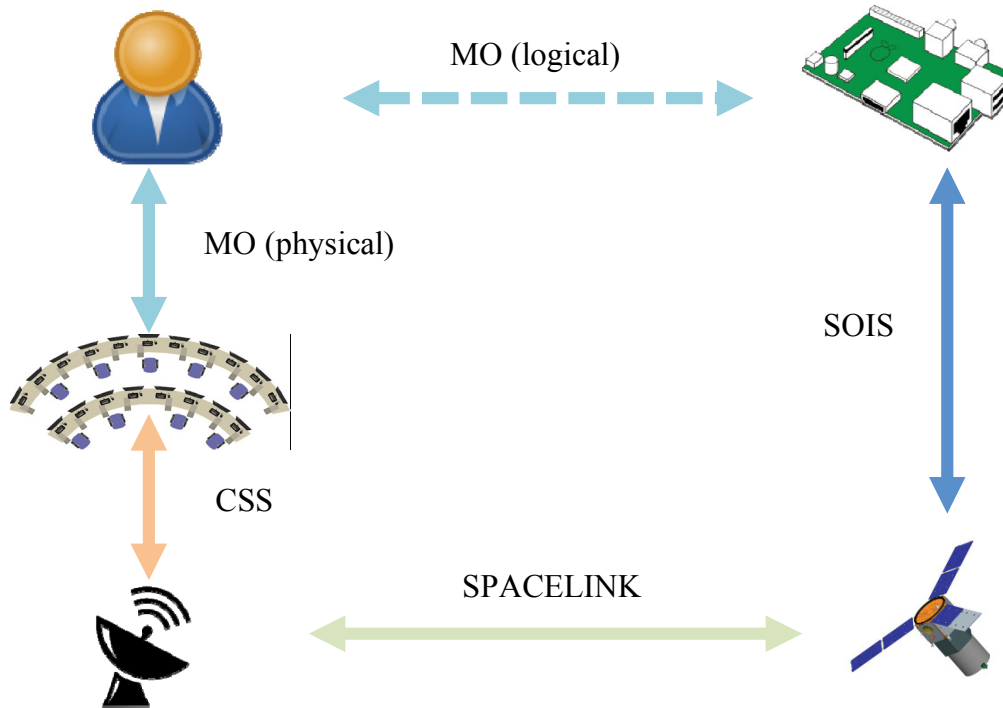


Figure 3-4: MO, SOIS, and Other CCSDS Services

The diagram above outlines how this works:

- The client institute uses a MO service to monitor or control a device. Logically, the corresponding set of messages flow to and from the device.
- Those messages pass through each of:
 - a suitable MO transport (e.g., HTTP) to get to the control center;
 - CSS protocols to get to the ground station;
 - SPACELINK protocols to get to the satellite;
 - SOIS protocols to get to the device.
- At each stage, the messages may be either translated into, or layered inside, the protocols used in the next step.¹²
- The opposite path is followed for a reply from the device to the user.

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

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

¹² For example, the control center typically generates SPACELINK TC transfer frames which are then encapsulated into appropriate CSS SLE messages.

4 ANALYSIS

4.1 AREAS OF OVERLAP BETWEEN THE STANDARDS

The two standards have different scopes and purposes, but do have two areas of overlap:

- Interfaces, which describe the set of possible message exchanges between two or more communicating entities
- Types, which describe and constrain the contents of those messages.

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.

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: TelecommandUserDataTypes

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 TelecommandUserDataTypes

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 exchanged with the device are split into fields with associated encodings, types and semantics.

The same interface can be expressed in MAL, allowing operators or high-level software applications to configure those settings on the device.

When such an interface is expressed in MAL, the encoding and layout details are left out. This is suitable for the intended usage of MAL, as it allows decisions on how to efficiently encode data to be taken centrally, and therefore consistently.

For interfacing 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.

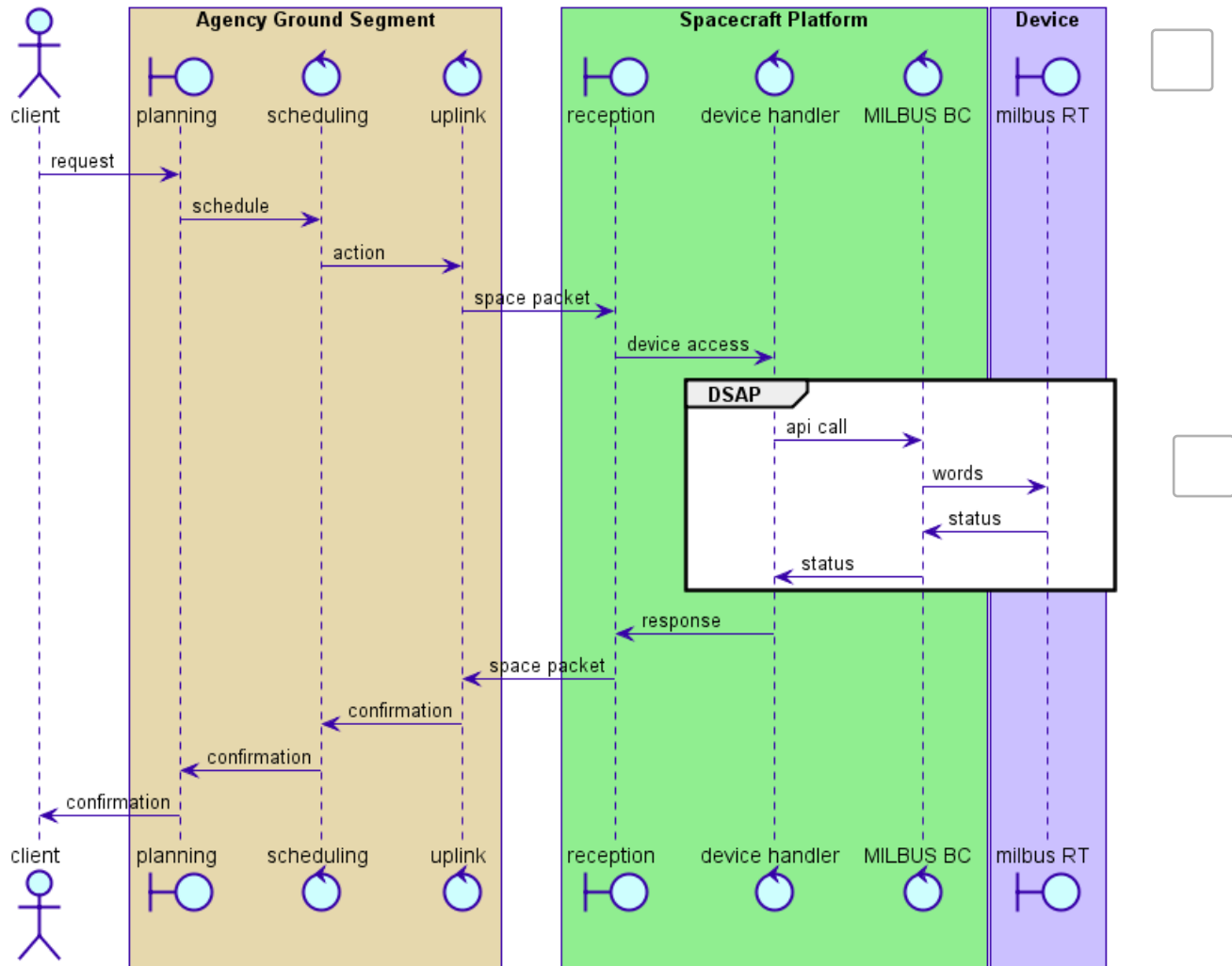


Figure 4-2: Sequence Diagram: Adjusting a Setting on a Device at the Request of the End User

The above sequence diagram shows a request being made by an end user through a representative range of services on the ground and space, and ultimately resulting in a sequence of data exchanges of binary words across a MILBus that conforms to the description specified in the DSAP of a CCSDS EDS.

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. Ideally, it would be technically possible to expose every hardware interface in this way, leaving the decision as to which interfaces *should* be so exposed to be based on the operations concept for the mission. From this perspective the access interface specified in a device EDS is a logical super set of the corresponding MO device service specification.

Of course, in some cases, devices would not be directly managed by the end user, but instead by autonomous onboard services, such as thermal control, 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 between the end user and thermal control, and thermal control and the device.

4.2 SPECIFYING INTERFACES

IEEE defines the verb to interface as ‘*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. Between programming languages, standards, middleware tooling, etc., there is 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.

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.

Formalism	Terminology	Encoding	Cardinality	Directions	Message Grouping
C family ¹³	set of functions	implicit ¹⁴	1:many	one-way	paired ¹⁵
PUS ¹⁶	service	explicit	many:many	two-way	none
RASDS ¹⁷	port	explicit	1:1	two-way	none
EDS	interface	optional	unspecified	two-way	paired
MAL	service	implicit	1:1 ¹⁸	one-way	patterned

¹³ The programming language C is included because of its historical influence on both 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.

¹⁴ The compiler selects the actual layout of data in memory, according to properties of the target CPU.

¹⁵ The return value of a function is inherently associated with the corresponding call.

¹⁶ ESA Packet Utilisation Standard, ECSS-E-ST-70-41C.

¹⁷ Reference Architecture for Space Data Systems (RASDS), CCSDS 311.0-M-1

4.3 DETAILED COMPARATIVE ANALYSIS

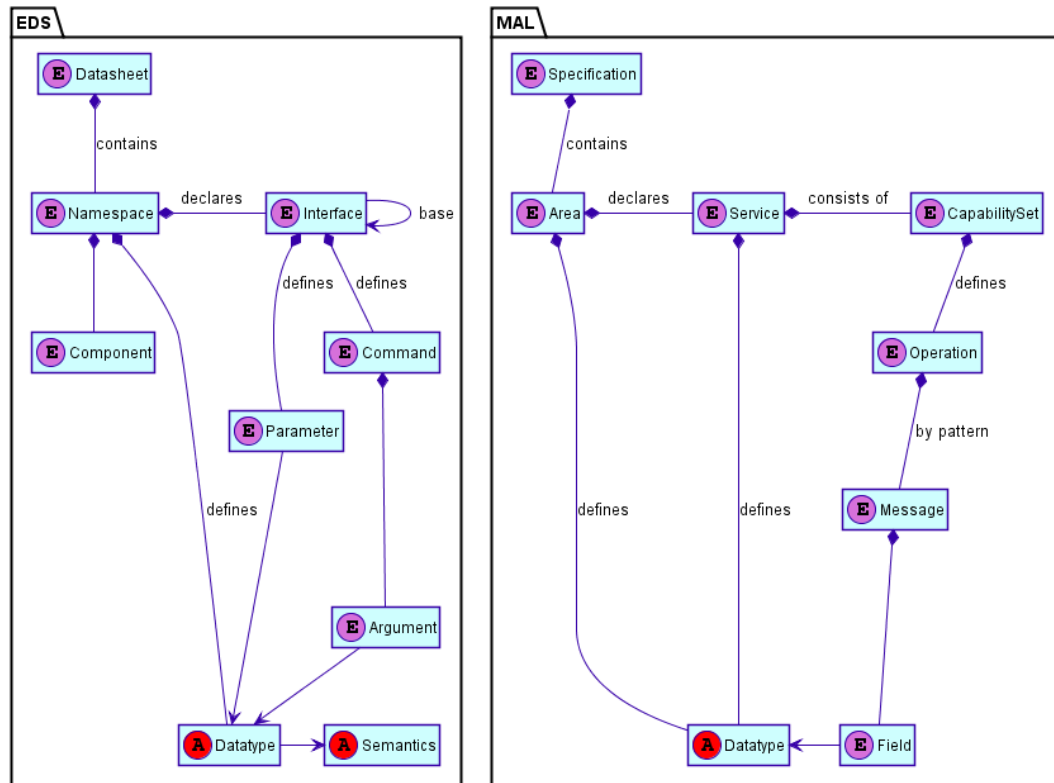


Figure 4-3: XML Structure of EDS and MAL

Areas marked with 'A' are abstract, hiding further detail.

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 use inheritance, and contain parameters and commands;
- commands have arguments;

¹⁸ Except a PubSub operation, which has 3 classes of participants, including any number of subscribers.

- 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;
- 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;
- each message has a number of named fields;
- each field has a data type.

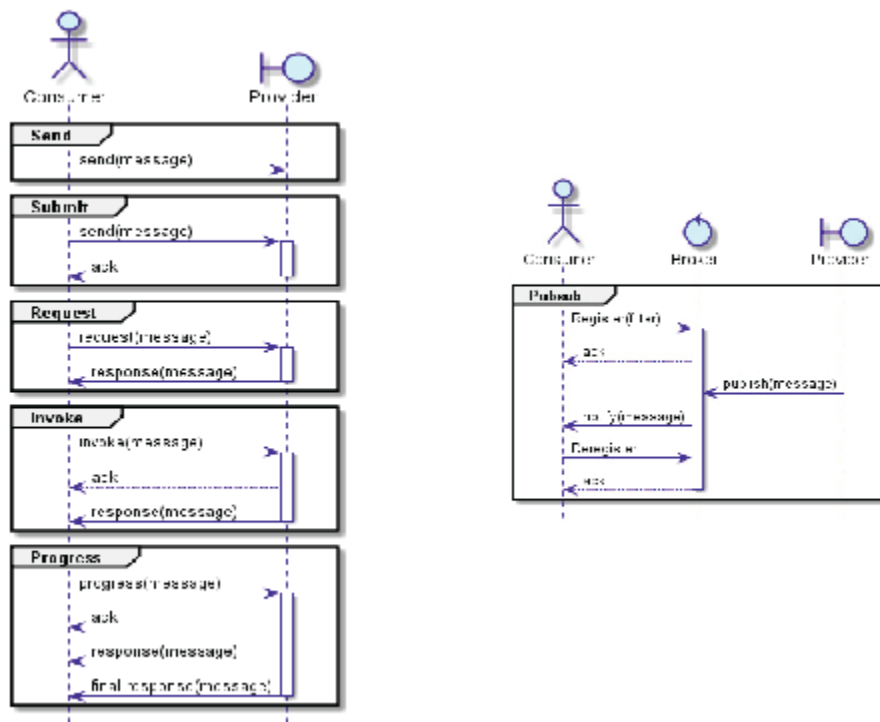


Figure 4-4: MAL Interaction Patterns

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

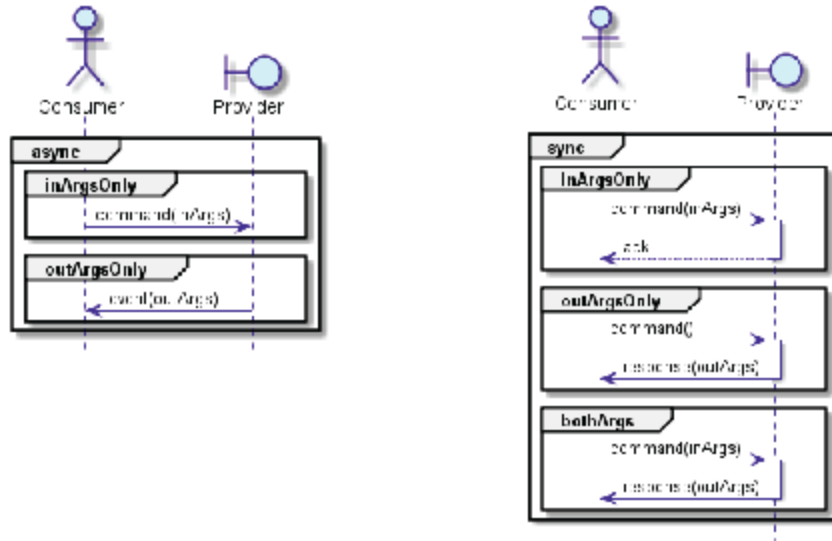


Figure 4-5: EDS Interaction Patterns

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.

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.

In short, there are three possible approaches:

- a) use a bespoke MO service automatically generated from a SOIS EDS;
- b) use Mission Operations (MO) Monitoring and Control Services (M&CS) Action and Parameter service with EDS;
- c) use device-category MO services with EDS.

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 replace 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* corresponding to the instantiated *Interface* specification as used by a particular component is defined.
- f) A MAL *Capability Set* for each Interface Specification involved in defining that interface is defined. 
- g) A MAL *Operation* for each EDS *Command* is defined, with 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 *bespoke* MO service.

Such a service could be:

- made available to ground directly;
- consumed by an autonomous device management application which 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 covers 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 onboard or not.

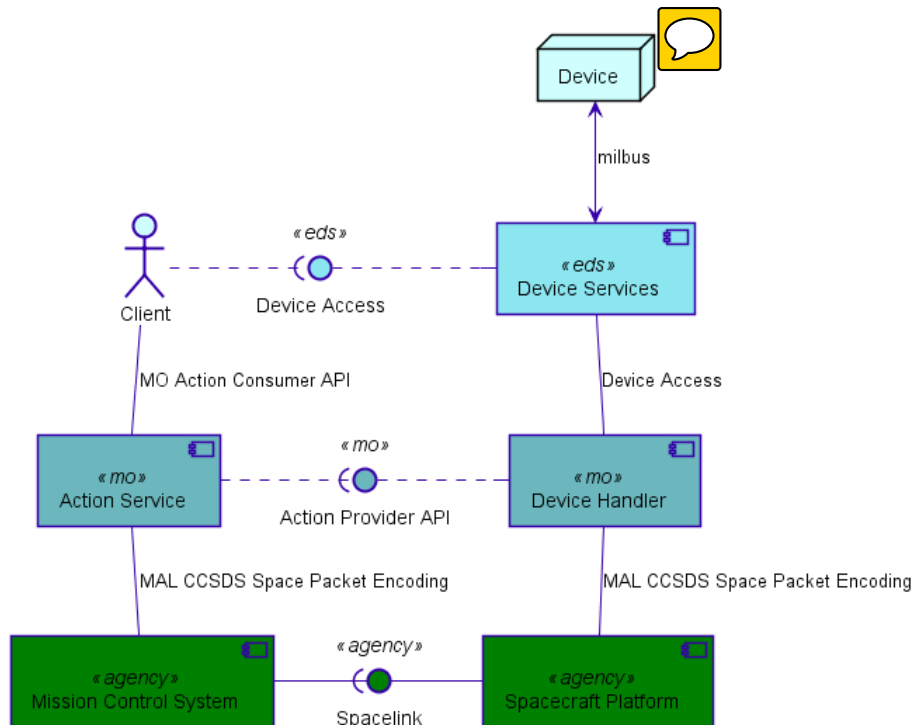


Figure 5-1: MO Action Service Implemented Using the Action Provider API Onboard

NOTE – Usage of the parameter service would be similar.

For its electronic device, the Client provides an electronic datasheet, which, in the EDS Device Access interface, gives the set of parameters and commands supported by the device. This allows establishing a logical link to communicate to the Device.

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

The client sends an action `configureMode(STANDBY)` using the MO Action Consumer API. The action service in the ground segment can logically, at the peer-to-peer layer, talk to the same layer onboard, using the standard MO Action Provider API.

To implement this, it encodes those messages using the MO space packet encoding, and sends them to the Agency layer as a CCSDS TC packet. This physically sends the space packets up to the satellite.

Onboard, the implementation at that layer is the Device Handler. This, when it receives the corresponding MAL-level message, uses the subnetwork interface described by the datasheet to talk to the device to physically 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. A 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.

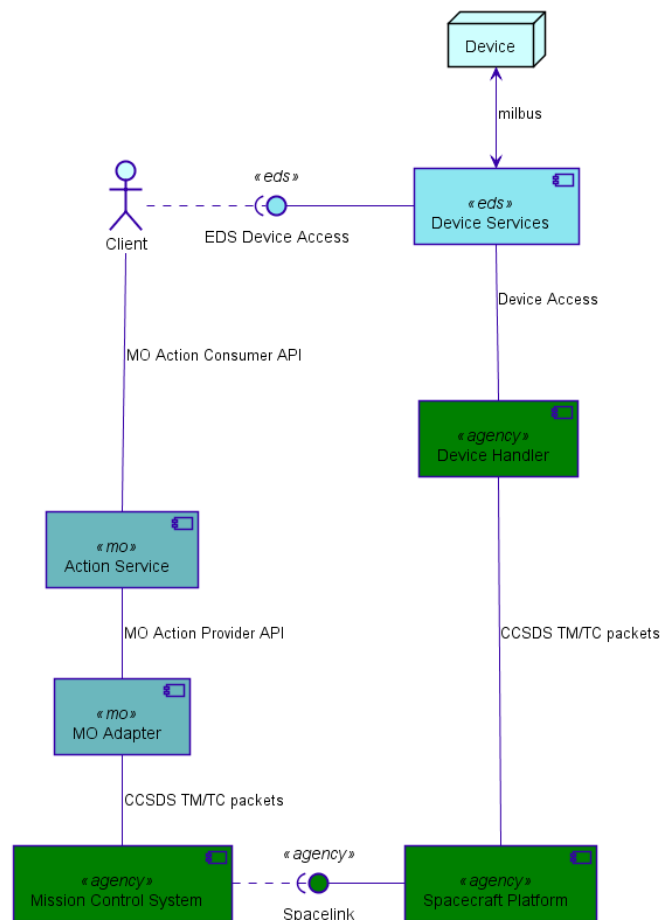


Figure 5-2: MO Action Service Implemented Using the Action Provider API On Ground

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

A straightforward adaptation of the above approach works in the case where MO is not supported onboard.¹⁹ The Device Handler is now implemented using the same techniques as the rest of the spacecraft platform, and supports arbitrary TM/TC, as defined in a spacecraft database. A ground-side MO Adapter translates that TM/TC into the calls to the action provider API that would have been made using the onboard approach. In other words, the Device Handler implements an MO services wrapper allowing communication with a legacy

¹⁹ This is the case of today’s spacecraft operated by any of the space agencies participating in CCSDS.

onboard architecture using MAL. However, this relaying comes with an overhead, whose estimation is beyond the scope of this technical note.

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.

Either implementation choice would be transparent to the end user.

5.4 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.

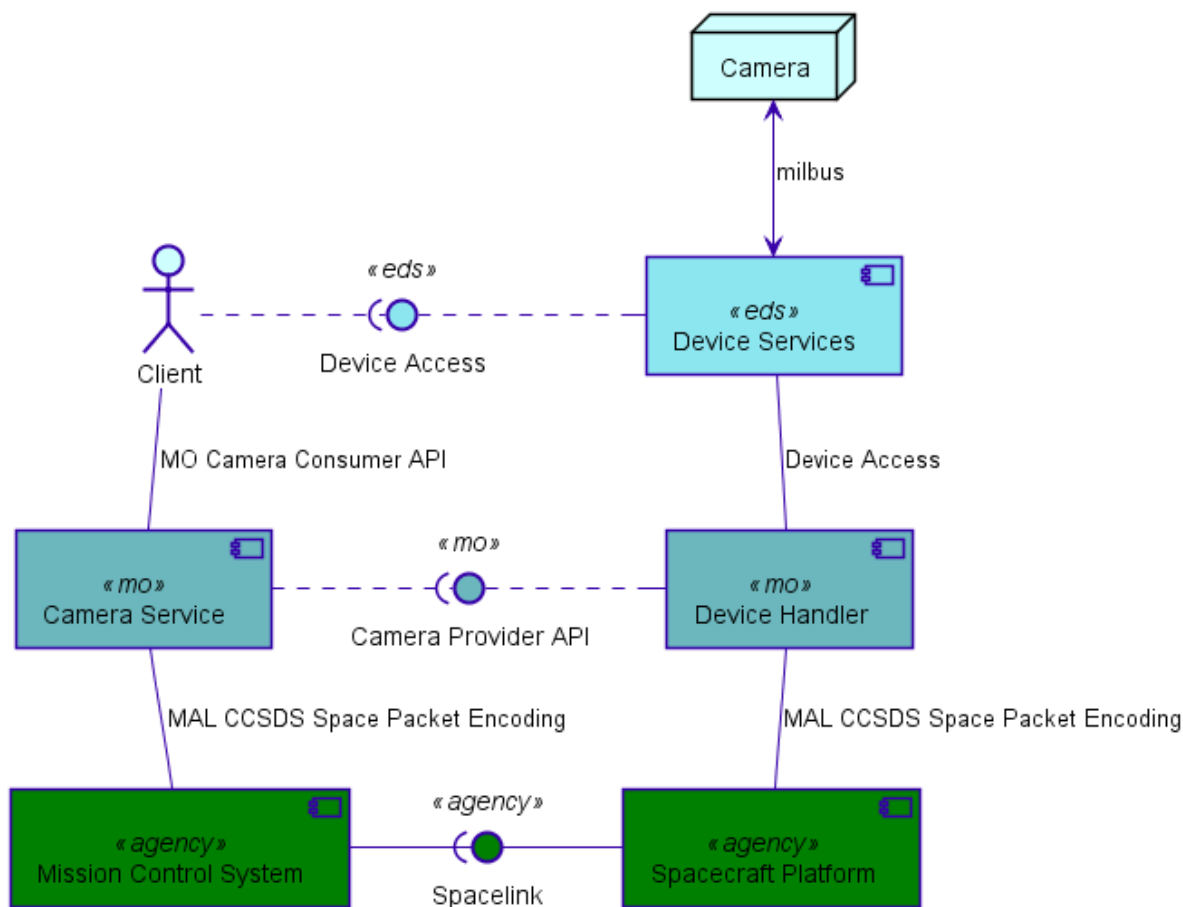


Figure 5-3: Bespoke MO Camera Service Implemented Using the Camera Provider API Onboard

For instance, considering a camera as the 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 this Camera service.

The implementation is very similar to the one 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 RECOMMENDATIONS AND CONCLUSION

The CCSDS MOIMS and SOIS working groups have different scopes, but have seen some convergence in technical approach, with specifically SOIS EDS and MOIMS MAL having a certain degree of overlap in capability, and also both of them using XML schema for their respective device and service specifications.

However:


- That overlap is limited to perhaps 15 to 20% of the scope of each specification.
- Analysis of a scenario where both MO services and SOIS EDS interfaces were in use shows there is no  to translate between the two, as they operate on different levels of abstraction.
- Translating between the two representations is in any case straightforward .

Therefore it can be deduced that attempting to create a common core specification, which the two usages would then differently extend, would be unlikely to be a worthwhile exercise.

Instead, lessons learned from this analysis should be fed back into the corresponding specification development processes, in order to improve areas where 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;
- 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/ASYNCR 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:

- The concept of ‘semantics’ is currently missing; parameters and arguments have at most an engineering unit, and they do not have any other structured information required to understand their meaning and .

ANNEX A

ABBREVIATIONS AND ACRONYMS

Term	Definition
AMS	Asynchronous Message Service
API	application programming interface
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
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
IRD	interface requirement document
M&CS	Monitoring and Control Services
MAL	Message Abstraction Layer
MCS	mission control system
MO	Mission Operations
MOIMS	Mission Operations and Information Management Services
OBC	on-board computer
OSI	Open Systems Interconnection
PDU	protocol data unit
PUS	Packet Utilisation Standard
QoS	quality of service
RASDS	Reference Architecture for Space Data Systems
RMAP	remote memory access protocol
RMI	Remote Method Invocation
RT	remote terminal
S/C	Spacecraft
SEDS	SOIS EDS
SLE	Space Link Extension
SOIS	Spacecraft Onboard Interface Services
SPP	Space Packet Protocol

Term	Definition
SVF	software validation facility
TC	Telecommand
TCP	Transmission Control Protocol
TM	Telemetry
UDP	User Datagram Protocol
UML	Universal Modelling Language
XML	eXtensible Markup Language
XTCE	XML Telemetric & Command Exchange