

**Report Concerning Space Data System Standards**

**MISSION OPERATIONS  
SERVICES CONCEPT**

**INFORMATIONAL REPORT**

**CCSDS 520.0-G-2**

**GREEN BOOK**  
August 2006

## AUTHORITY

Issue:	Informational Report, Issue 2
Date:	August 2006
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and reflects the consensus of technical panel experts from CCSDS Member Agencies. The procedure for review and authorization of CCSDS Reports is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*.

This document is published and maintained by:

CCSDS Secretariat  
Office of Space Communication (Code M-3)  
National Aeronautics and Space Administration  
Washington, DC 20546, USA

## FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This document is therefore subject to CCSDS document management and change control procedures which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. 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 addressed to the CCSDS Secretariat at the 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.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (Roskosmos)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Organization (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

## DOCUMENT CONTROL

<b>Document</b>	<b>Title</b>	<b>Date</b>	<b>Status</b>
CCSDS 520.0-G-1	Mission Operations Services Concept, Informational Report, Issue 1	May 2006	Original issue, superseded
CCSDS 520.0-G-2	Mission Operations Services Concept, Informational Report, Issue 2	August 2006	Current issue

## CONTENTS

<u>Section</u>	<u>Page</u>
<b>1 INTRODUCTION</b> .....	<b>1-1</b>
1.1 PURPOSE AND SCOPE .....	1-1
1.2 DOCUMENT STRUCTURE.....	1-1
1.3 REFERENCES.....	1-1
1.4 DEFINITION OF ACRONYMS .....	1-2
<b>2 OVERVIEW OF MISSION OPERATIONS SERVICE CONCEPT</b> .....	<b>2-1</b>
2.1 GENERAL .....	2-1
2.2 GOALS .....	2-1
2.3 SCOPE .....	2-4
2.4 SUMMARY OF SERVICE FRAMEWORK .....	2-7
<b>3 DEFINITION OF THE SERVICE FRAMEWORK</b> .....	<b>3-1</b>
3.1 GENERAL .....	3-1
3.2 APPROACH TO SERVICE IDENTIFICATION.....	3-2
3.3 SERVICE STRUCTURE.....	3-7
3.4 IDENTIFIED MISSION OPERATIONS SERVICES .....	3-20
<b>4 DOCUMENT ROADMAP</b> .....	<b>4-1</b>
<b>ANNEX A DEFINITION OF TERMS</b> .....	<b>A-1</b>
 <u>Figure</u>	
2-1 Service Oriented Architecture.....	2-2
2-2 Example Distribution of Mission Operations Functions.....	2-5
2-3 Relationship of Mission Operations Services to Other CCSDS Standards .....	2-6
2-4 Overview of SM&C Mission Operations Service Framework .....	2-7
2-5 Service Tunnelling .....	2-10
2-6 Mission Operations Services Overview .....	2-11
3-1 Service Model .....	3-7
3-2 Mission Operations Service Framework Layers.....	3-11
3-3 Information-Oriented Mission Operations Services .....	3-12
3-4 Generic Interaction Pattern for Mission Operations Services.....	3-13
3-5 Controller and Target Pattern.....	3-15
3-6 Generic Service Object Information Model.....	3-16
4-1 Document Roadmap.....	4-1

**CONTENTS (continued)**

<u>Table</u>	<u>Page</u>
3-1 Mission Operations Functions .....	3-3
3-2 Mission Operation Information—Types and Usage .....	3-6
3-3 Mission Operations Services.....	3-20

## 1 INTRODUCTION

### 1.1 PURPOSE AND SCOPE

This CCSDS Green Book is an informational report that presents an overview of a concept for a Mission Operations Service Framework for use in spacecraft monitoring and control. It has been prepared by the Spacecraft Monitoring and Control (SM&C) working group of the Mission Operations and Information Management Systems (MOIMS) area.

In this context, Spacecraft Monitoring and Control (SM&C) refers to end-to-end services between functions, resident on-board a spacecraft or based on the ground, that are responsible for mission operations.

### 1.2 DOCUMENT STRUCTURE

Following this introduction, the document is organised into three main sections:

#### **Section 2: Overview of Mission Operations Services Concept**

Provides an introduction to the goals and scope of the concept and a summary of the proposed service framework.

#### **Section 3: Definition of the Service Framework**

Outlines the approach to service identification; service structure in terms of three framework layers (SM&C Protocol, Common Services and Mission Operations Services); introduces the identified Mission Operations services.

#### **Section 4: Document Roadmap**

Presents the proposed standards documentation tree and associated prioritisation.

#### **Appendix A: Definition of Terms**

### 1.3 REFERENCES

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

- [1] *Space Link Extension Service Specifications*. [Space Link Extension Service Specifications are in various stages of development. Current issues of CCSDS documents are maintained at the CCSDS Web site: <http://www.ccsds.org>.]

## 1.4 DEFINITION OF ACRONYMS

AMS	Asynchronous Messaging Service (of SIS)
API	Application Programmer's Interface
CFDP	CCSDS File Distribution Protocol
CORBA	Common Request Broker Architecture
GPS	Global Positioning System
HCI	Human Computer Interface
M&C	Monitoring and Control
MCC	Mission Control Centre
MCS	Mission Control System
MDA	Model Driven Architecture
MO	Mission Operations
MOIMS	Mission Operations and Information Management Systems (CCSDS Area)
OMG	Object Management Group
PDU	Protocol Data Unit
PIM	Platform Independent Model
PSM	Platform Specific Model
QOS	Quality of Service
SAP	Service Access Point
SDU	Service Data Unit
SIS	Space Internetworking Services
SLE	Space Link Extension
SLS	Space Link Services
SM&C	Spacecraft Monitoring and Control
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOIS	Spacecraft On-board Interface Services
TC	Telecommand
TM	Telemetry
UML	Unified Modelling Language
WSDL	Web Services Definition Language

## **2 OVERVIEW OF MISSION OPERATIONS SERVICE CONCEPT**

### **2.1 GENERAL**

This section provides an executive summary of the Mission Operations Service Concept and is presented in three subsections as follows:

- **Goals:** problem identification, service framework approach and potential benefits;
- **Scope:** mission operations, system boundaries and relationship to CCSDS standards;
- **Summary of the Service Framework:** context, layering and service overview.

### **2.2 GOALS**

#### **2.2.1 IDENTIFICATION OF THE PROBLEM**

There is a general trend toward increasing mission complexity at the same time as increasing pressure to reduce the cost of mission operations, both in terms of initial deployment and recurrent expenditure.

Closed, or ‘monolithic’ mission operations system architectures do not allow the re-distribution of functionality between space and ground, or between nodes of the ground system. This lack of architectural openness leads to:

- lack of interoperability between agencies;
- lack of re-use between missions and ground systems;
- increased cost of mission specific development and deployment;
- unavailability of commercial generic tools;
- inability to replace implementation technology without major system redesign;
- lack of operational commonality between mission systems, increased training costs.

The result is many parallel system infrastructures that are specific to a given family of spacecraft or operating agency, with little prospect of cross-fertilisation between them.

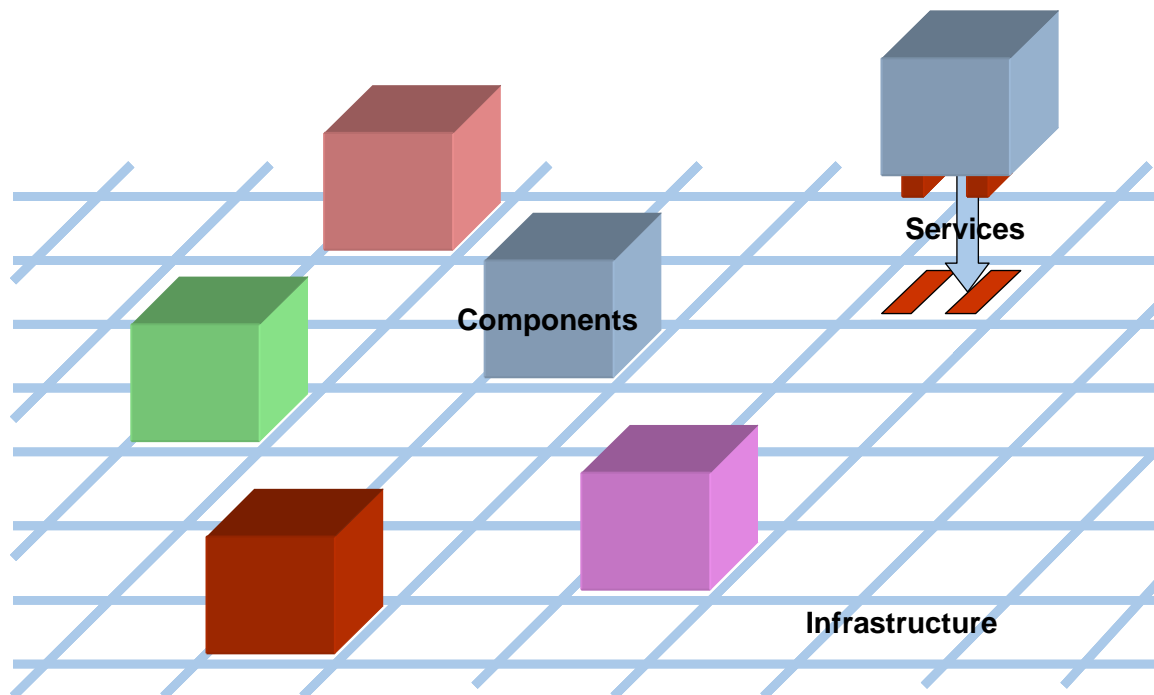
#### **2.2.2 THE SERVICE FRAMEWORK APPROACH**

Service Oriented Architecture (SOA) is gradually replacing monolithic architecture as the main design principle for new applications in both private and distributed systems. It is one of the fundamental design principles of network distributed applications where the interfaces, both operations and data objects, must be well defined as the clients are often heterogeneous.

SOA is an approach to system design that relies not on the specification of a monolithic integrated system, but instead on the identification of smaller, modular components that communicate only through open, published, service interfaces.

This set of standard services constitutes a framework that enables many similar systems to be assembled from compliant ‘plug-in’ components. These components may be located anywhere, provided they are connected via a common infrastructure. This allows components to be re-used in different mission-specific deployments: between agencies, between missions, and between systems.

If services are specified directly in terms of a specific infrastructure implementation, then they are tied to that technology. Instead, by layering the services themselves, the service specifications can be made independent of the underlying technology. Specific technology adapters enable the deployment of the service framework over that technology. This in turn makes it possible to replace the infrastructure implementation as well as component implementations. It is also possible to transparently bridge between different infrastructure implementations, where these are appropriate to different communications environments (e.g., space or ground) or simply reflect different agencies’ deployment choices.



**Figure 2-1: Service Oriented Architecture**

NOTE – Plug-in components communicate only via standard service interfaces through a common infrastructure. The service framework is itself layered and independent of the underlying infrastructure implementation.

It is also important to note that the approach does not prescribe the components themselves or their implementation. Only the service interfaces between components are standardised.

This allows for innovation, specialisation and differentiation in components, while ensuring they can be rapidly integrated into a system. However, for the service framework to be effective it must ensure that meaningful information associated with mission operations can be exchanged across the service interfaces, not merely data.

The service framework must also respect legacy systems. Where an integrated legacy system performs the function of several service framework components, its internal architecture and implementation do not have to be changed. Only those interfaces it exposes to other systems need be 'wrapped' to make them compliant with the corresponding service interfaces. The service framework offers a range of interoperable interfaces, from which the most appropriate can be selected: compliance is not dependent on supporting them all. In this way legacy systems can be re-used in conjunction with other compliant components to build a mission specific system.

*The approach is intended to be Evolutionary and not Revolutionary.*

### 2.2.3 POTENTIAL BENEFITS

Standardisation of a Mission Operations Service Framework offers a number of potential benefits for the development, deployment and maintenance of mission operations infrastructure:

- increased **interoperability** between agencies, at the level of spacecraft, payloads, or ground segment infrastructure components;
- standardisation of infrastructure interfaces, even within agencies, leading to **re-use** between missions and the ability to establish common multi-mission infrastructure;
- standardisation of operational interfaces for spacecraft from **different manufacturers**;
- **reduced cost** of mission-specific deployment through the integration of re-usable components;
- ability to **select the best** product for a given task from a range of compatible components;
- greater flexibility in deployment boundaries: **functions can be migrated** more easily between ground segment sites or even from ground to space;
- standardisation of a **limited number of services** rather than a large number of specific inter-component interfaces;
- increased competition in the provision of commercial tools, leading to cost reduction and **vendor independence**;
- improved long-term maintainability, through **system evolution** over the mission lifetime through both component and infrastructure replacement.

## **2.3 SCOPE**

### **2.3.1 MISSION OPERATIONS**

The term Mission Operations is used to refer to the collection of activities required to operate spacecraft and their payloads. It includes:

- monitoring and control of the spacecraft subsystems and payloads;
- spacecraft performance analysis and reporting;
- planning, scheduling and execution of mission operations;
- orbit and attitude determination, prediction and manoeuvre preparation;
- management of on-board software (load and dump);
- delivery of mission data products.

These are typically regarded as the functions of the Mission Control Centre (MCC) and are performed by the mission operations team, supported by the Mission Operations System.

Mission operations includes the capability to archive and distribute mission operations data. While this may include the handling of mission data products, activities specifically concerned with the exploitation of mission data (such as mission specific data processing, archiving and distribution) are considered outside the scope of mission operations.

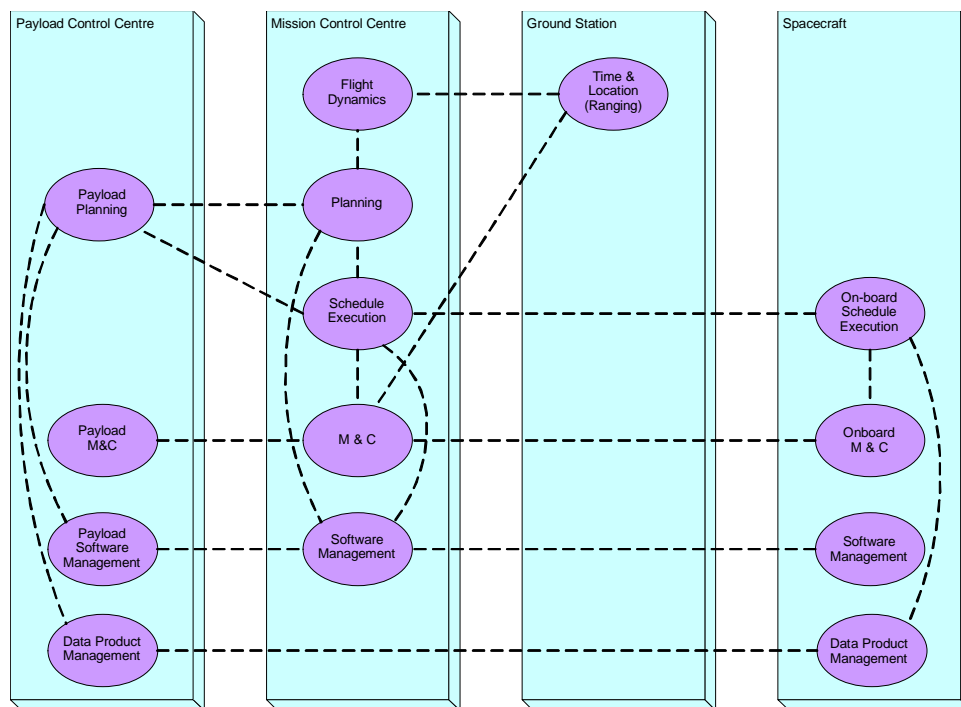
Increasingly, mission operations functions may be distributed between collaborating agencies and ground segment sites, or partially delegated to autonomous functions on-board the spacecraft itself.

The Mission Operations Service Framework is concerned with end-to-end interaction between mission operations application software, wherever it may reside within the space system. It is specifically not concerned with the provision of services for data transport or persistence (storage). It is, however, a user of such services.

### **2.3.2 SYSTEM BOUNDARIES AND INTEROPERABILITY**

The figure below shows one of many possible configurations of a spacecraft mission operations system. It is not the intention that any one distribution of mission operations functions should be prescribed. The needs of individual missions will require flexible collaboration between agencies. Although operational responsibility for a satellite normally resides with its owner agency, it may carry payloads, probes or landers that are owned and operated by third parties.

It is also the case that satellites from several different manufacturers may be owned and operated by a single agency. The demands for greater on-board autonomy and increasing on-board processing power will also allow migration of functionality on-board the spacecraft. This exposes more complex mission operations interactions to the space-ground interface. Standardisation will enable the development of re-usable infrastructure in both ground and space segments.



**Figure 2-2: Example Distribution of Mission Operations Functions**

NOTE – The example shows mission operations functions distributed between a Spacecraft and three separate Ground Segment facilities. The connecting lines show end-to-end interactions between these functions. As each facility could be operated by a separate agency, where interactions cross distribution boundaries these could constitute interoperable interfaces. Other distributions may be used.

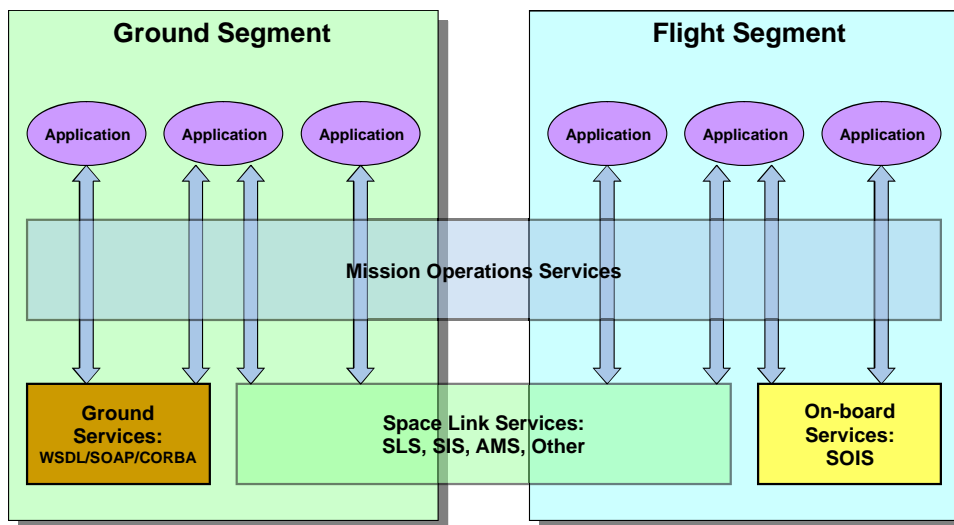
Where an interface is exposed between agencies, it becomes an interoperable interface and a candidate for standardisation. The variability of mission operations system configuration outlined above, means that most of the main inter-functional interfaces of mission operations could be either internal or external to a given system.

Even within an agency or other operating organisation, there are benefits to the standardisation of mission operations services, as outlined in 2.2.3 above.

The concept for a mission operations service framework allows for incremental standardisation as follows:

- a) Priority is given to services that are currently exposed at interoperability boundaries.
- b) Services exposed at key internal interfaces within the infrastructure of multiple agencies will be standardised second to encourage the development of re-usable infrastructure components.
- c) Finally, services are identified to allow future evolution of interoperable interfaces with increased complexity of missions and on-board autonomy.

### 2.3.3 RELATIONSHIP TO OTHER CCSDS STANDARDS



**Figure 2-3: Relationship of Mission Operations Services to Other CCSDS Standards**

NOTE – Mission Operations Services are end-to-end application level services and may be carried over other communications protocols appropriate to the environment.

The Mission Operations Service framework addresses end-to-end interaction between applications that reside within both the space and ground segments. The underlying transport services over which Mission Operations services are carried may be different depending on the nature of the communications path:

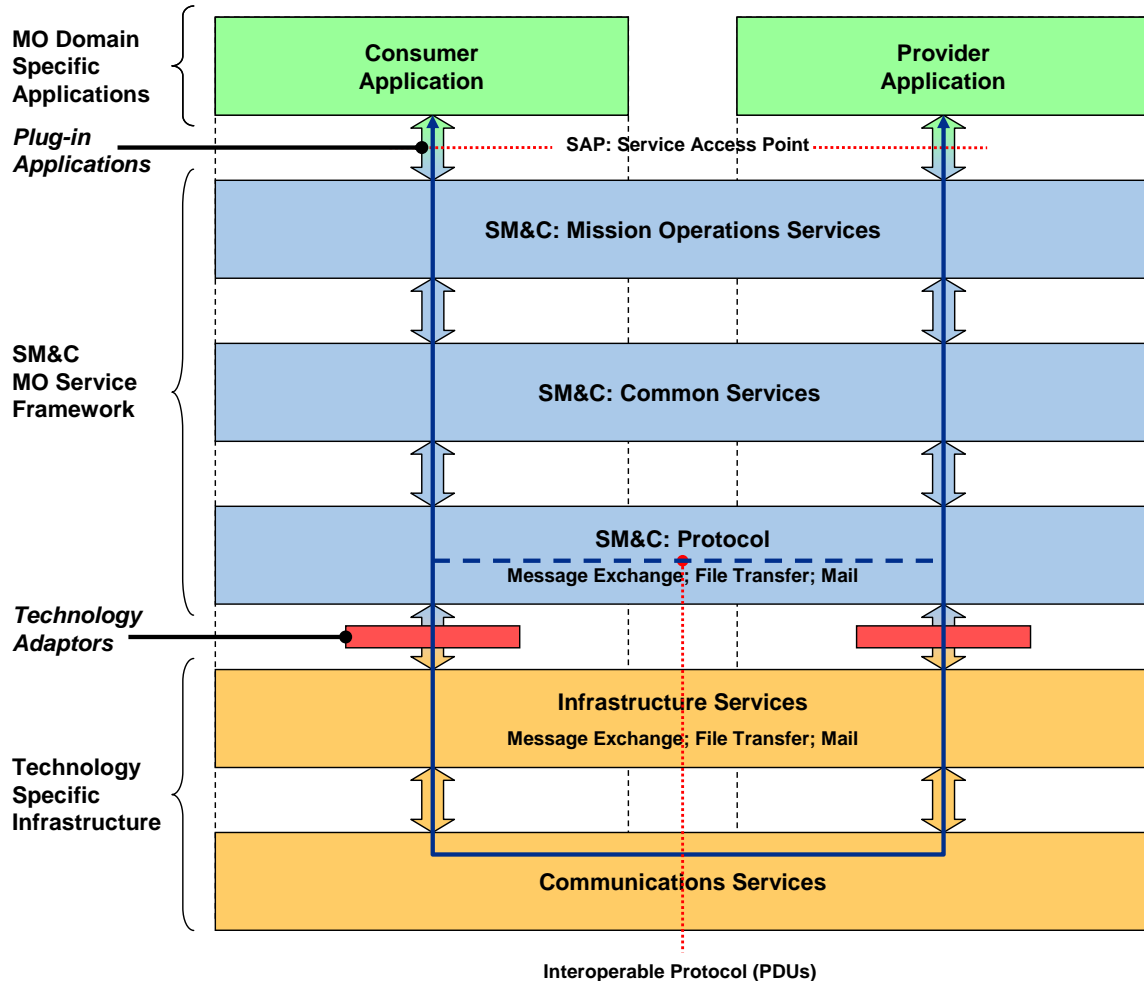
- **Between space and ground**, this is expected to use CCSDS Space Link Services (SLS), Packet TM/TC, and optionally CCSDS Space Internetworking Services (SIS). In particular, the proposed Asynchronous Messaging Service (AMS) offers a messaging layer over which the protocol messages of the Mission Operations Service framework could be carried. Similarly CFDP may be used to support file transfer.
- **Within the ground segment**, wider industry standard middleware services may be used, such as SOAP, JAVA-RMI or CORBA. Alternatively, AMS could be used over TCP/IP. Similarly FTP may be used to support file transfer.
- **On-board the spacecraft** itself, CCSDS Spacecraft Onboard Interface Services (SOIS) could be used: these offer protocols more suited to the limited resource environment on-board the spacecraft.

Some applications may bridge between one underlying communications environment and another: e.g., a ground mission control system may use Packet TM/TC to communicate with the spacecraft, but SOAP to forward parameter status to a payload control centre.

CCSDS Cross Support Services (SLE), which are not shown in the diagram, may also be used transparently to the Mission Operations Services to extend the space link from ground stations to a mission operations centre. There is overlap between the two areas where a service terminates at the Ground Station (e.g., for Tracking data), and close collaboration will be required to avoid duplication.

## 2.4 SUMMARY OF SERVICE FRAMEWORK

### 2.4.1 GENERAL



**Figure 2-4: Overview of SM&C Mission Operations Service Framework**

NOTE – The framework isolates mission operations applications from each other and the underlying communications infrastructure. The framework itself has three layers: application level mission operations services, common services on which these are built, and a protocol layer that ensures interoperability between different framework implementations.

The CCSDS Spacecraft Monitoring & Control (SM&C) working group has developed a concept for a Mission Operations (MO) Service Framework, which follows the principles of Service Oriented Architectures. It defines an extensible set of end-to-end services that support interactions between distributable mission operations functions, e.g., software applications specific to the mission operations domain.

## 2.4.2 CONTEXT OF MISSION OPERATIONS SERVICE FRAMEWORK

The MO Service Framework sits between application software specific to the domain of spacecraft mission operations and the underlying technology used for communications between distributed applications. This isolates compliant software applications both from each other and the underlying communications technology.

Applications may be ‘plugged in’ to the service framework through standardised Service Access Points (SAPs), one for each end-to-end service. The SAP is defined in terms of a Platform Independent Model (PIM). For any given service, this offers two compatible interfaces that support the service Consumer and service Provider applications respectively.

This approach is consistent with the OMG’s Model Driven Architecture (MDA) and is capable of being fully modelled using UML. Tools exist to support this approach that support and even automate the process of generating first the Platform Specific Model (PSM) for a given deployment technology and then the associated program code. When the SAP is bound to a specific deployment technology (e.g., programming language) it is cast as an Application Programmers’ Interface (API) specific to that technology. This API forms the interface to a re-usable library that implements the MO Service Framework, and is called directly by the application software. Provision of a standard UML model together with a service specification would greatly facilitate the development of software compliant with the standard.

The MO Service Framework itself is also independent of the underlying technology specific infrastructure. Each deployment technology requires an adapter that allows the framework to be deployed over it. This abstraction of the MO Service Framework from the deployment infrastructure implementation means that the entire framework can be migrated from one deployment technology to another without modification to the domain specific applications themselves. It also allows bridging between different technologies, where these are suited to particular communications environments, or to accommodate different implementation choices between agencies.

## 2.4.3 SERVICE FRAMEWORK LAYERS: OVERVIEW

The Mission Operations Framework has three layers as illustrated in figure 2-4 above:

- a) Mission Operations Services Layer;
- b) Common Services Layer;
- c) SM&C Protocol Layer.

These are introduced below, but a fuller description is given in 3.3.

The **Mission Operations Services Layer** provides the end-to-end services that are exposed to mission operations applications. Multiple services have been identified, each corresponding to a particular class of information that is exchanged between mission operations applications. Examples include:

- The **Core M&C** services of Parameter status provision, Action (Command) invocation, and Alert notification;
- **Schedule** (operations timeline) delivery, control and execution status monitoring;
- Spacecraft **Time** and **Location**;
- Management of on-board **Software** loading and dumping.

The **Common Services Layer** provides a common service infrastructure for all Mission Operations services. This ensures a common approach, simplifies the task of implementing each Mission Operations service, and isolates the Mission Operations service layer from the underlying implementation technology. The layer comprises:

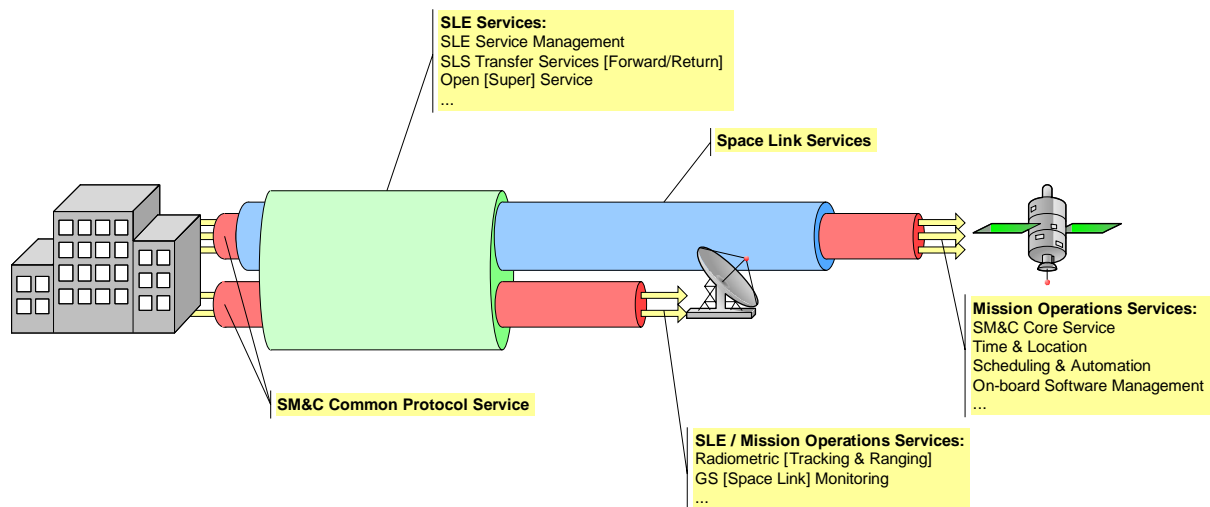
- common service elements used by all Mission Operations services that are directly exposed to mission operations applications (e.g., the Service Directory);
- generic service elements that provide a basis for, and are specialised by, the Mission Operations services themselves.

The **SM&C Protocol Layer** provides an interoperable protocol between the Consumer and Provider sides of the service framework. This, together with standardised bindings between the MO Service Framework layers, ensures that different implementations of the service framework can interoperate across the service interfaces, providing the underlying communications protocol stack is equivalent on both sides of the interface. The protocol layer supports three fundamental communication mechanisms:

- Message Exchange;
- File Transfer;
- Mail.

Another way of viewing this layering is that the mission operations services are transported through pipes provided by the underlying transport protocol. The common service and protocol layers provide an end-to-end pipe, which itself is transported via the transport protocol, such as that provided by Space Link Services.

For example, Space Link Extension (SLE) services extend the space link services to control centres by providing a pipe through which both SLS and other services which terminate at the ground station can be transported:



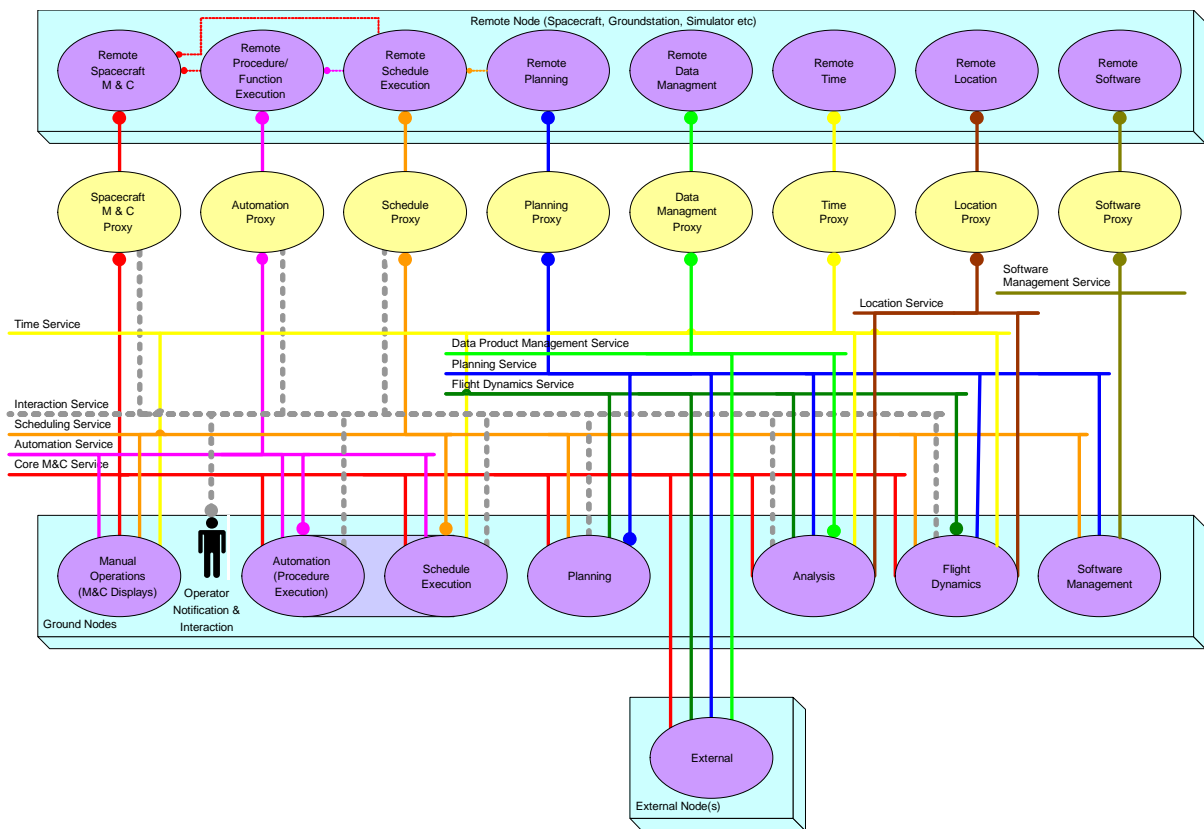
**Figure 2-5: Service Tunnelling**

NOTE – Services are carried through ‘pipes’ provided by underlying layers of communications services. CCSDS Space Link and Space Link Extension are examples of services over which the mission operations services can be deployed.

### 2.4.4 MISSION OPERATIONS SERVICES: OVERVIEW

The figure below illustrates the concept of a set of end-to-end application level Mission Operations services for the monitoring and control of spacecraft. The core set of operational functions shown are ones that exist, or may exist, in many current and future spacecraft control systems. These functions may be distributed between physical locations, organisations and systems in many different ways. Increasingly, part or all of these functions are being deployed on-board spacecraft.

The intention is to identify services that allow different distributions of operational functions to be adopted by individual organisations or missions.



**Figure 2-6: Mission Operations Services Overview**

#### NOTES

- 1 Interconnecting lines represent services that provide the end-to-end interface between functions.
- 2 The Mission Operations services are shown as horizontal lines, each service in a different colour. The vertical connections to functions show those functions which are potential providers (line ends in a circle) or consumers of the service. Note that there can be multiple providers within the system, and functions can act as both provider and consumer. Other connections may be present in some systems.

The Core Monitor and Control service is architecturally one of the application level mission operations services shown in the figure, but has a key role as it provides the most basic services for monitoring and controlling various system elements. Other services are identified to support more complex operations, such as management of on-board schedules and software.

All the Mission Operations services shown are layered over the SM&C Protocol and Common Services layers introduced above.

The services may be used to interface functions wherever they reside: within the ground system, or on-board the spacecraft, or an external system. For on-board functions, ground-based functions may access their services through a proxy function resident on the ground. This proxy can manage intermittent connectivity, provide service history and also encapsulate a legacy protocol on the space-ground interface.

Another key feature of the concept is an information base that defines the capabilities of the various devices and components that are managed by these services. This comprises the configuration data for each service, with a separate copy for each occurrence (or instance) of a service (e.g., per spacecraft). This service configuration data defines the objects that are exposed by the service, together with their associated attributes and actions.

### **3 DEFINITION OF THE SERVICE FRAMEWORK**

#### **3.1 GENERAL**

This section provides a more detailed overview of the concept for the Mission Operations Service Framework.

Firstly, the approach to the identification of Mission Operations services is outlined, concentrating on the identification of the main Mission Operations functions and the derivation of the fundamental Information classes that flow on the interfaces between these functions.

Secondly, the structure of Mission Operations services is described, in terms of:

- Basic Service Model;
- Service Framework Layers, including Common Services and SM&C Protocol;
- Information Model for service objects;
- Some of the General Concepts behind the service framework.

Finally, the identified Mission Operations services are introduced.

## 3.2 APPROACH TO SERVICE IDENTIFICATION

### 3.2.1 OVERVIEW

The approach taken to Mission Operations Service identification proceeded in the following main stages:

- 1) Identification of the main Mission Operations **functions**;
- 2) Identify the principal **interfaces** between functions;
- 3) Identify common types of **information** flow across interfaces;
- 4) Identify the **operations** associated with information transfer (both for monitoring and control);
- 5) Group homogeneous information and operations into **services**;
- 6) Identify commonality between services and derive **common** and **generic** service elements;
- 7) Specification of an interoperable **protocol** that supports the common and generic service elements.

The granularity of the functions selected as the basis for service identification is crucial. At too high a level of granularity (too few functions), there is insufficient flexibility in the potential deployment of systems compliant with the service framework, and few of the benefits of service oriented architectures will be realised. At too low a level of granularity, the service framework becomes overly prescriptive in the component architecture; a given function is presented with too many potential interfaces, ultimately reducing the scope for compatible 'plug-in' components; and the size of the standardisation task becomes unmanageably large.

The identification of the fundamental types of Information that are exposed at the end-to-end inter-functional interfaces at the selected level of granularity is a key step in the identification of services.

Stages 1 and 3 are therefore explored in more detail below. Stage 5 is the subject of 3.4 below, and the results of Stages 6 and 7 are apparent in the discussion of service structure given in 3.3.

### 3.2.2 MISSION OPERATIONS FUNCTIONS

A set of principal mission operations functions has been identified as common to the experience of multiple agencies responsible for mission operations. These functions, which are listed in table 3-1 below, have been used as the basis for mission operations service identification.

**Table 3-1: Mission Operations Functions**

<b>Function</b>	<b>Description</b>
<b>Monitoring and Control</b>	Core functions of mission control system, including status monitoring, commanding and notification of alert conditions.
<b>Manual Operations</b>	HCI function in support of human operators, including status displays and manual control interfaces that enable manual execution of mission operations.
<b>Automation</b>	Automated execution of mission operations at two levels: <ul style="list-style-type: none"> <li>– Schedule (mission timeline) execution</li> <li>– Procedure or Function execution</li> </ul>
<b>Planning</b>	Planning of future mission operations, taking into account requests for operations from various sources, predicted orbital events and operational constraints. Generates schedules (mission timelines) for manual or automatic execution.
<b>Software Management</b>	Management of the process of deploying software on-board a spacecraft: loading, patching and dumping of software images.
<b>Flight Dynamics</b>	Orbit and Attitude determination and prediction; manoeuvre planning.
<b>Time Management</b>	Correlation and synchronisation of on-board time.
<b>Location</b>	Measurement of spacecraft position, whether through ground-based tracking and ranging, or on-board positioning.
<b>Analysis</b>	Performance analysis, trending and reporting.
<b>Data Product Management</b>	Control, management and transfer of mission data products including their delivery to mission exploitation systems.

A criterion in selection of the functions has been to consider the potential distribution boundaries within the mission operations system. Where there is scope for functions to be distributed between different agencies, sites or systems then the interfaces which are potentially exposed become candidates for standardisation on interoperability grounds.

Even where such interfaces are not exposed between agencies, there is a strong case for standardisation to enable the development of re-usable infrastructure components.

The combined effect of increasing power in on-board computers and an increased requirement for autonomy in operations, is that functions are progressively being implemented within the space segment that have previously been found only on the ground. As this progresses, functional interfaces that were previously internal to the mission control system will be exposed to the space-ground interface. Standardisation of these interfaces will limit the amount of mission specific customisation required to adapt ground segment infrastructure to interact with spacecraft from different manufacturers or production lines.

The identification of this particular set of functions is not intended to prescribe the architecture of mission operations systems, particularly in terms of the level of aggregation of mission operations functions. In any given mission operations system deployment, it is probable that several of the functions identified will be grouped together in a single system component.

For example:

- A typical Mission Control System product may encompass Monitoring and Control, Manual Operations, Software Management and Time Management functions, while ancillary systems support Mission Planning, Flight Dynamics and Analysis.
- Automation may be fully integrated within the Mission Control System, external to it, or be partly delegated to autonomous on-board functions such as on-board schedules and procedures.
- Mission Planning may be managed as a separate activity to Mission Control (even at a different site), be fully integrated with Automation in the Mission Control System, or even partially delegated to a goal-based on-board autonomous planning system.

It is precisely the point that there can be many different deployment architectures, but there are many benefits in terms of interoperability and re-use if all these architectures can be harmonised to work within a common Mission Operations Service Framework.

This approach allows:

- Selection of the appropriate standard service interfaces for a given mission operations application, to enable its deployment in conjunction with other compliant products. Each system only needs to expose those interfaces that are appropriate to its function.
- Legacy systems to be integrated by wrapping their external interfaces to be compliant with standard Mission Operations services. In this way, a manufacturer or agency can

wrap their proprietary or legacy interfaces through the provision of a technology adapter compliant with the service framework. This can either be integrated with the legacy system itself, or provided to an operating or partner agency to be installed in their system in much the same way a device driver is installed on a computer.

- Evolution of infrastructure towards a set of re-usable software applications compliant with the Mission Operations service framework. This does not have to be done as a single step.

### 3.2.3 MISSION OPERATIONS INFORMATION

Following the approach introduced in 3.2, the principal types of information exposed at inter-functional interfaces are listed in table 3-2.

This shows the information classes together with those functions that are potential providers or consumers of services associated with the information class. Provider functions may be located on the ground, or on-board the spacecraft. While many of the provider functions identified are typically ground-based in existing systems, there is scope for future migration of at least part of the function on-board spacecraft in the future.

Where a function is on-board, a ground-based *proxy* function may manage the interface with ground-based functions. Such a proxy can:

- Encapsulate legacy or proprietary interfaces on the space-ground interface;
- Manage discontinuity in the space-ground link;
- Provide information persistence (storage and retrieval of history).

The last four columns of the table indicate at what level these information types are exposed at interoperable interfaces. Four categories are shown:

- **Internal:** well-established interfaces within the mission infrastructure of multiple agencies. Candidate for standardisation on grounds of re-use.
- **Space-Ground:** exposed on the space-ground interface. Candidate for standardisation on grounds of re-use and commonality of interfaces between different spacecraft manufacturers/buses.
- **External:** interfaces to organisations outside the operating agency itself (e.g., Principal Investigators, Mission Exploitation Centres, Spacecraft Manufacturers).
- **Inter-Agency:** interoperable interface between agencies on collaborative missions.

**Table 3-2: Mission Operation Information—Types and Usage**

Information Class	Information Exchange		Interoperability			
	Potential Provider Function	Potential Consumer Function	Internal	Space-Ground	External	Inter-Agency
<b>Core M&amp;C: Parameters Actions Alerts</b>	M&C (Spacecraft)* M&C (G/Station) Any other ground-based function or proxy	Manual Operations Automation Analysis Flight Dynamics External	✓	✓	✓	F
<b>Time</b>	Time Management*	M&C Analysis Flight Dynamics	✓	✓	✓	✓
<b>Location</b>	Tracking (G/Station) On-board Positioning	Flight Dynamics Analysis	✓	F	✓	✓
<b>Orbit Vector Attitude Vector Predicted Events</b>	Flight Dynamics*	Planning Analysis Ground Station External	✓	F	✓	✓
<b>Data Product</b>	Data Product Management* Analysis	External	✓	✓	✓	✓
<b>Planning Request/ Goal</b>	Planning*	Operations Team External Flight Dynamics Software Management Analysis	✓	F	✓	F
<b>Schedule</b>	Automation: Schedule Execution* Any other automated function	Manual Operations Mission Planning Flight Dynamics On-board Software Man.	✓	✓	F	F
<b>Procedure or Function</b>	Automation: Procedure Execution* Any other automated function	Manual Operations Automation: Schedule Execution Procedure Execution	✓	✓	F	F
<b>Software</b>	Spacecraft	On-board Software Man.	✓	✓	✓	F
<b>Interaction</b>	Operations Team Manual Operations	M&C Automation Any other function	✓	F	F	F
<b>Remote Buffer</b>	Spacecraft Ground Station	Manual Operations Automation	✓	✓	F	F
<p>* indicates a provider function could be deployed on-board or be a proxy for an on-board function.                      ✓ indicates an interface that is currently exposed.                      F indicates an interface that could be exposed in the future.</p>						

### 3.3 SERVICE STRUCTURE

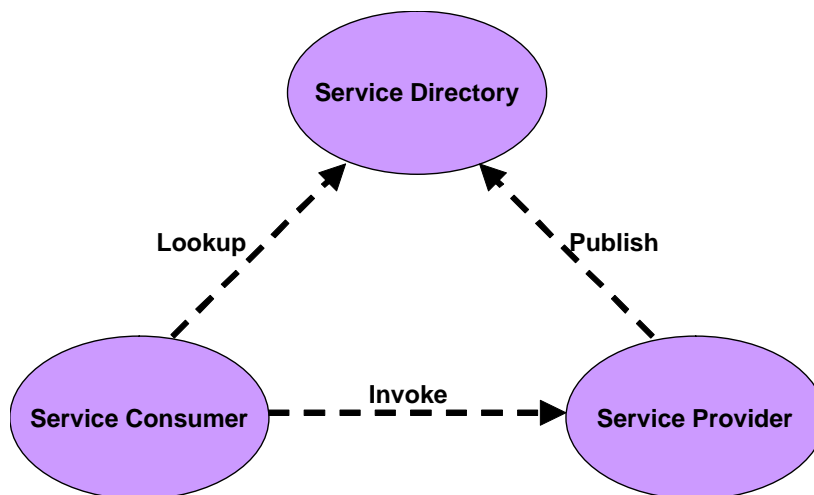
#### 3.3.1 GENERAL

The essential structure of the Mission Operations Service Framework has already been presented in 2.4. This section expands upon this and introduces:

- The basic Service Model, with Service Provider, Consumer and Directory;
- The three Service Framework Layers;
- The generic Information Model for a service object;
- General Concepts that are key to the Service Framework, including *domain* hierarchy and *sessions*.

#### 3.3.2 THE SERVICE MODEL

##### 3.3.2.1 Overview



**Figure 3-1: Service Model**

NOTE – The Service Provider publishes its services to a Service Directory; the Service Consumer looks up required services in the directory and then invokes these directly with the Service Provider(s).

An SOA is essentially a design that starts with an interface definition and builds the entire application based around the interfaces, interface semantics (state machines of the interface) and interface calls (operations allowed and data objects passed). Whilst there are no universally recognised standards for service based architecture, the concepts and terminology used are nevertheless well developed and consistent, having evolved in line with their wide utilisation. Comparison shows that these are analogous to the service model conventions used for SLE (R1).

**Service:** A service is an operation, or set of operations, that is well defined, self-contained and does not depend on the state or context of another service. A service may be implemented in terms of, or use another service but this should not be apparent to a service consumer.

**Service provider:** A service provider is an entity that implements a service, equivalent to the target in a controller and target pattern. A service provider may also be a service consumer of lower level services. However, this would and should be transparent to the consumers of the service; i.e., this is an implementation detail.

**Service consumer:** A service consumer is an entity that uses a service being supplied by a service provider. A service consumer may also be a service provider to higher level service consumers. However, this would and should be transparent to the lower level service being invoked; i.e., this is an implementation detail.

**Service directory:** A service directory is an entity that provides publish and lookup facilities to service providers and consumers. Strictly speaking a directory is not required if a well known service is to be used; however, in most circumstances a directory provides required flexibility in the location of services. Service location can be statically configured, dynamically discovered through a service directory or a combination of the two; this is an implementation choice. The service directory is itself, by definition, a service.

The preference of the terms *provider* and *consumer* with respect to the service architecture is driven by the fact that they reflect the use of the service. One entity provides a service, whereas another entity consumes the service. An alternative to consumer is user, i.e., ‘service user’; however, this can conflict with the generic term ‘user’ which often means a person involved in the system.

The terms service provider and service consumer are also predominantly used in the distributed web application domain.

### 3.3.2.2 Service Versions

In order to support the evolution of services over time, service provider functions, and hence the service directory, will need to have the capability to support multiple versions of the same service at the same time. This is essential as it will not be possible to upgrade all components of the ground data system simultaneously.

Two cases that will need to be supported are:

- The service provider offers a service instance that is backwards compatible with previous versions of the service, and publishes all supported versions to the service directory. The service consumer invokes the service, specifying the required version.
- The service provider offers distinct service instances, each corresponding to a version of the service. The service consumer invokes the appropriate service instance.

### 3.3.2.3 Capability Sets

Services are identified and defined around core classes of information (*objects*) that exist at a *service interface*. However, different providers and consumers of the same basic type of information may be concerned with different aspects of that information.

It is not desirable for all implementations of service providers or consumers to necessarily have to support aspects of a complex service that they do not require. This is particularly relevant when a service is to be deployed in an environment where there are limited resources, such as on-board the spacecraft. Equally, it is not desirable that a service definition be reduced to the lowest common denominator of service provision, in order to accommodate all implementations, as this limits the benefit of standardisation. Nor is it desirable that closely coupled aspects of a common information object should be divided between different services.

The concept of *capability sets* is introduced to manage this complexity by offering a way of decomposing service functionality. A capability set comprises a particular aspect of the information model for the service. Typically, this does not correspond to a sub-set of *service objects*, but to optional attributes and *operations* associated with a class of service object.

For example, for a parameter status provision service could have distinct capability sets for:

- Parameter Calibration;
- Parameter Monitoring;
- Parameter Statistics.

The service definition identifies the capability sets, and specifies whether these are mandatory or optional for compliance with the standard. Optional capability sets may also have dependencies on other optional capability sets. The existence of such dependencies may be a key reason for combining capability sets within a single service.

A given implementation of a service provider can then offer the mandatory capability sets, plus a subset of optional capability sets. However, where it offers an optional aspect of the service, this should be compliant with the service definition for that optional capability set.

A given implementation of a service consumer may only require (and support) a subset of optional capability sets. It should be able to determine whether these are offered by a service provider. It should also be possible for the service consumer to interact with a service provider offering optional capability sets that the consumer does not support. When a service provider publishes a service instance to the service directory, it must also list the optional capability sets provided, such that a service consumer is able to determine the level of service provided.

A service provider may also extend the set of capabilities offered and publish these as custom capability sets. Evidently, only a compatible service consumer would be able to make use of such custom capability sets. This flexibility supports service extension as needed, and the subsequent adoption of new capability sets as part of the standard service.

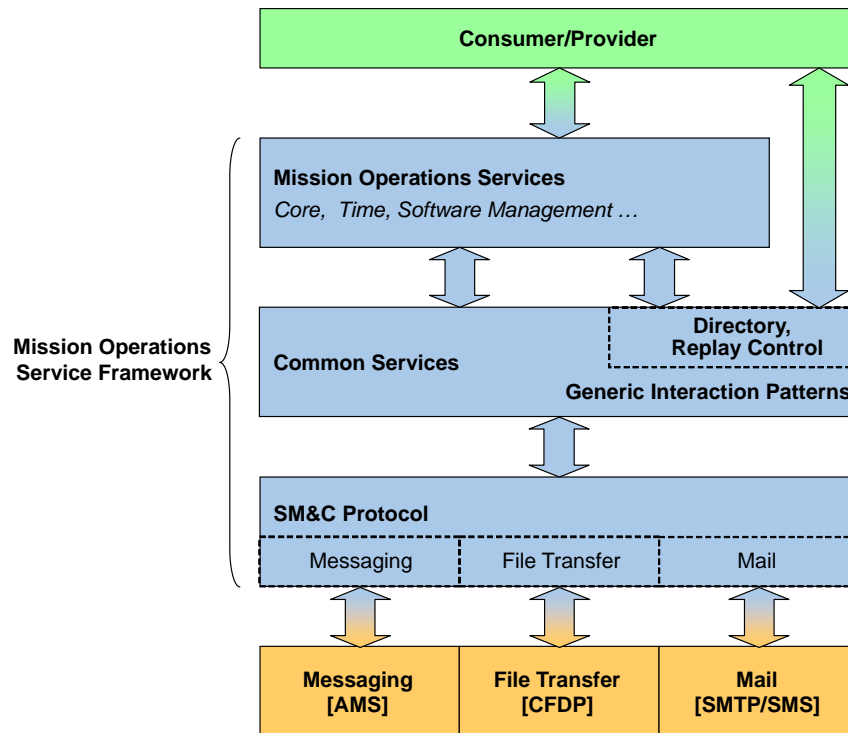
To summarise, capability sets allow:

- Functional decomposition of a service;
- Compliance for service providers/consumers offering a subset of service capability;
- Integration of legacy systems offering a subset of service capability;
- Negotiation between consumer and provider of the level of capability required;
- Service extension.

### 3.3.3 SERVICE FRAMEWORK LAYERS

#### 3.3.3.1 General

The Mission Operations Service Framework has three layers as introduced in 2.4.3 and figure 2-4. In practice the relationship between the layers is slightly more complex, as illustrated below.



**Figure 3-2: Mission Operations Service Framework Layers**

NOTE – Mission Operations services and Common services such as the Service Directory are exposed to applications. Mission Operations services are implemented as extensions to generic services provided by the Common layer. The Common layer is bound to the Protocol layer which provides generic Messaging, File Transfer and Mail interaction methods.

Each layer exposes Service Access Points (SAPs) to the layer above. When deployed on a particular technology these are cast as Application Programmer's Interfaces (APIs).

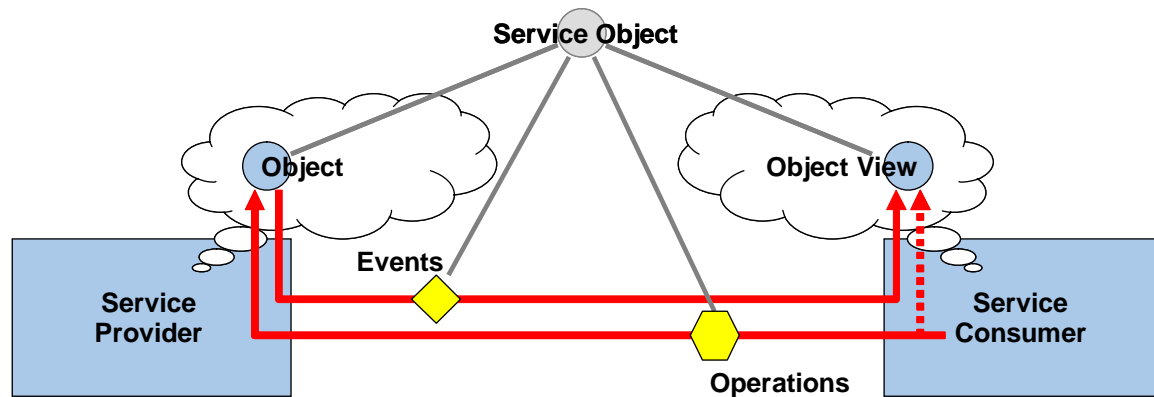
The services exposed to Mission Operations applications are the Mission Operations services and those elements of the Common Services layer that provide service common to all Mission Operations Services: primarily the Directory Service.

The Mission Operations services may be implemented as specialisations or extensions to generic interaction patterns provided by the Common Services layer. They may also utilise common service elements.

The SM&C Protocol layer provides interoperability between dissimilar implementations of the Mission Operations Service Framework, and isolation from the underlying deployment technology.

The following subsections describe each layer in turn.

### 3.3.3.2 Mission Operations Services



**Figure 3-3: Information-Oriented Mission Operations Services**

NOTE – Service Objects (e.g., a Parameter) are exposed at the service interface. The consumer registers interest in a subset of service objects and receives event messages that synchronises its view of object status with that of the provider. Similarly the consumer can invoke operations on the object (e.g., set parameter). The symbols  $\blacklozenge$  (for Event) and  $\blacklozenge$  (for Operation) are used throughout this document.

This layer provides the end-to-end services that are exposed to mission operations applications. Multiple services have been identified, each corresponding to a particular class of information that is exchanged between mission operations applications. An overview of the identified services is given in 3.3.5.5.

Each service is defined in terms of an information model that defines a set of service objects that are shared by providers and consumers of the service. Examples of such service objects are status *parameters*, control *actions* and notification *alerts*. These constitute the basic elements of the Core M&C service. Other services concern specialised information such as orbit vectors, schedules, planning requests and software images.

In addition to definition of the static information model, the service defines the interactions required between service provider and consumer to allow:

- the service consumer to observe the status of objects through a flow of *event* messages;
- the service consumer to invoke *operations* upon the objects.

The service definition specifies the structure of the information objects exposed at a particular service interface. In most cases, however, each deployment (or instantiation) of a service will also require service configuration data that details the actual service objects that exist for that service instance. For example, the Core M&C service may define what *parameters*, *actions* and *alerts* are, but it is the associated service configuration data that specifies the set of parameters, actions and alerts that exist for a particular spacecraft.

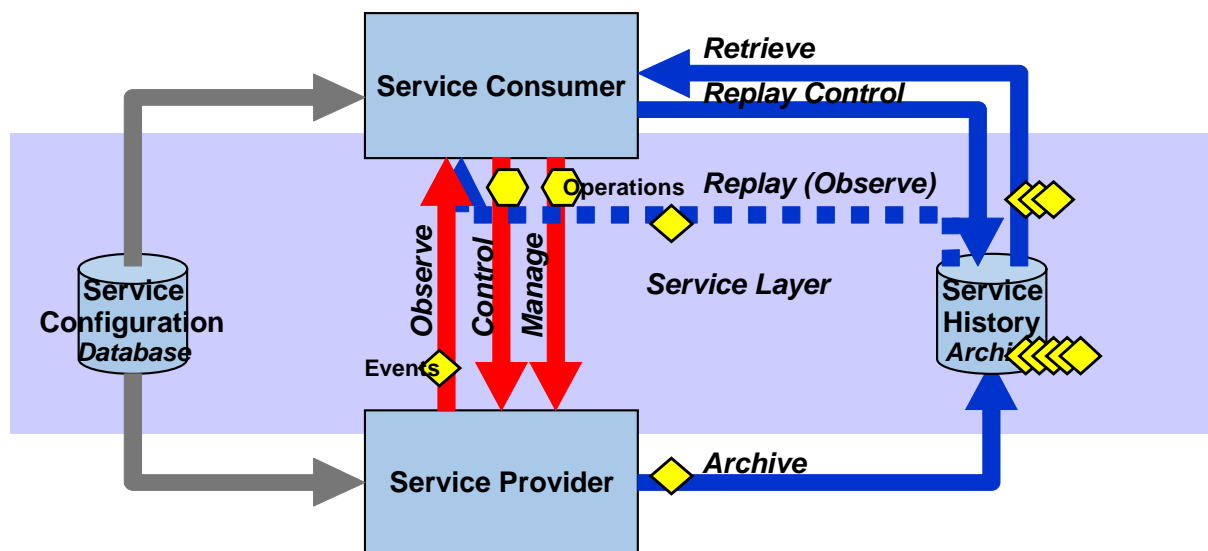
### 3.3.3.3 Common Services

This layer provides two things:

- **Common Service Elements;**
- **Generic Service Elements.**

In addition to the horizontal layering of services, the Mission Operations Service Framework can be broken down into a number of vertical elements. Some of these elements or sub-services are common to all MO services. These common service elements are directly exposed to applications. Examples of common services are:

- the *directory service* that allows consumers to locate providers of required services;
- the *replay control service* that allows consumers to initiate, configure and control a historical replay session (see 3.3.5.1), potentially coordinated across multiple services.



**Figure 3-4: Generic Interaction Pattern for Mission Operations Services**

NOTE – Generic Observe, Control and Manage services support any higher layer service object. Additional services support distribution of service configuration data and access to service history in both dynamic replay and batch retrieval modes.

In analysing the requirements for several potential end-to-end mission operations services, it became apparent that there is a lot of commonality between services. While the definition of objects, events and operations are specific to the service, multiple services can be based on the same fundamental interaction pattern. This includes abstract service elements such as:

- *observe* interface in which the service consumer registers for interest in particular service objects and then receives status update events for those objects (e.g., obtain parameter, action or schedule status);
- *control* interface in which the service consumer can initiate operations on objects (e.g., set parameter, send command, load schedule);
- *manage* interface in which the service consumer can modify the behaviour or processing of the service provider.

Layering of the Mission Operations Services over these generic interaction mechanisms simplifies the task of building adapters to the underlying communications technology, as only one adapter is required to support all Mission Operations Services.

In addition to these direct interfaces, there is scope to provide generic infrastructure support for recording and retrieving service history, as multiple services are based on the same generic observe interface, and also for the distribution of service configuration data.

#### **3.3.3.4 SM&C Protocol**

The previous layers have been concerned with end-to-end interaction between Service Provider and Consumer. Emphasis has been on the definition of Service Access Points and the standardisation of the vertical communication between the layers.

Given that the implementation of the service framework itself may differ between agencies and systems, it is critical for these infrastructures to be able to interoperate that there is standardisation of the messages (or Protocol Data Units [PDU]) that pass between Provider and Consumer.

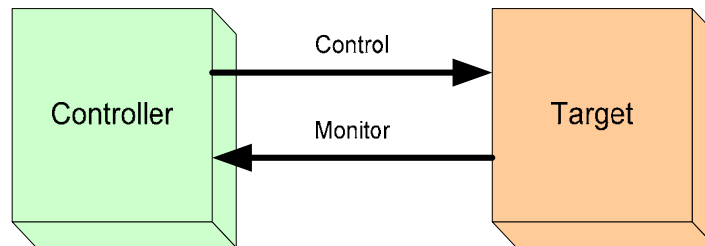
The SM&C Protocol Layer provides this horizontal standardisation between interoperable implementations of the MO Service Framework. The communications protocol stack beneath the MO Service Framework must be equivalent on both sides of the interface. Within the MO Service Framework itself, it is the protocol layer that ensures interoperability, as the bindings between it and the higher layers are standardised.

The protocol layer allows for three fundamental communications methods:

- **Messaging** (which could be implemented over AMS);
- **File Transfer** (which could be implemented over CFDP);
- **Mail**.

It provides a thin layer within the service framework that allows separate technology adapters to be implemented for the underlying communications protocols used for each of these communications methods.

Emphasis is placed on the messaging protocol, which is based on the Target-Controller pattern for monitoring and control.



**Figure 3-5: Controller and Target Pattern**

NOTE – The controller can issue control directives and receive monitored status of the target. The protocol layer defines the interoperable message structure for control and status messages.

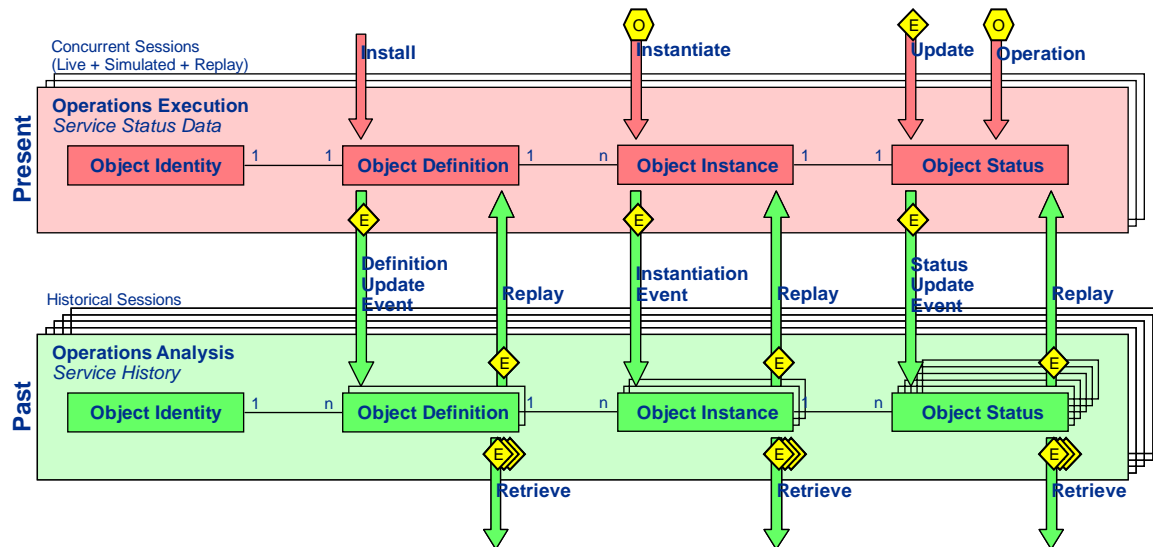
In the context of the Mission Operations Service Framework, the Controller is equivalent to the service consumer and the Target is equivalent to the service provider.

A controller can be a ground control system, an onboard data handling subsystem, or a processor of a payload/subsystem. A target can be a device, a subsystem, or even an entire spacecraft. This target - controller pattern can be applied recursively.

There is a standard set of operations that the SM&C Protocol provides, that may be used to transfer directives from any controller to any target and similarly to transfer reports from any target to any controller. This standard pattern of interaction may be used to implement any of the following standard operations:

- **Send Directive to Target**
  - Confirmed or Unconfirmed
- **Read State of Target**
  - Confirmed or Unconfirmed
- **Send Indication to Controller**
  - Confirmed or Unconfirmed
- **Send Event to Controller**
  - Confirmed or Unconfirmed

### 3.3.4 GENERIC SERVICE OBJECT INFORMATION MODEL



**Figure 3-6: Generic Service Object Information Model**

NOTE – Object identity is unique within a particular service instance and session. Object Definitions are contained within service configuration data and may change over time. Each invocation of an object results in a new Object Instance whose Status evolves over time.

The Common Layer supports a generic information model for all Mission Operations service objects. This is illustrated in the figure above. Service objects correspond to the principal information types exposed at service interfaces, as described in 3.3.3.2 above.

Within the context of a particular service instance (e.g., for a given spacecraft) and session (see 3.3.5.1), the identity of an object (e.g., a *parameter* or *action*) is unique and does not change over time.

The definition of the object contained within the corresponding service configuration data, however, can change over time as the result of the installation of a new version of that service configuration data. Within a current *session*, however, there is only one current object definition.

For some objects, like *parameters*, there is only ever one copy of the object, which is fully defined by its definition. These are statically instantiated objects that have a continuous existence and a state that evolves over an unlimited period of time.

For other objects, like *actions*, a new copy of the object is created with each new invocation; this is the *object instance*. These dynamically instantiated objects have a transient existence and a state that evolves over a limited lifetime. The definition of these objects may include a set of arguments, whose value is only defined when it is invoked. Multiple instances of the same object definition may be active in the system at the same time.

This gives rise to the four-part representation of the service object shown in the diagram:

- **Object Identity;**
- **Object Definition;**
- **Object Instance** (for dynamically instantiated objects only);
- **Object Status.**

For a service consumer to keep its view of an object in step with the service provider, any change object status must be notified as an event across the *observe* interface. Three types of event can be identified:

- **Definition Update Events;**
- **Instantiation Events;**
- **Status Update Events.**

If these events are also stored in history, then a consumer can reconstruct a historical view of the service object, as it was at the original execution time, based solely on the information contained in the service history. Access to service history can also be supported in both dynamic replay and static (batch) retrieval. Dynamic replay is possible as all history is stored as timed events. Co-ordinated replay of the history of multiple services is also possible as the same approach to the storage of history can be applied to all mission operations services.

This generic service object information model makes it possible to conceive of generic infrastructure components that support the distribution of service configuration data and storage of service history. While this is not imposed by the standardisation of Mission Operations Services, it offers a clear benefit in terms of re-use and information exchange between agencies.

### **3.3.5 GENERAL CONCEPTS**

#### **3.3.5.1 Operations Sessions**

For a given mission operations service, it may be possible to observe both current (live) data and also (initiated via a historical data replay service) data replayed from stored history. In some systems it may be possible to observe both live and historical data in parallel. It may also be possible to observe data originating from a simulator or test configuration in parallel to that originating from the live operational system.

The entities being controlled in the live, simulated or test cases (and monitored in both these and historical replay cases) are the same. In order to distinguish these parallel operational scenarios, it is necessary to partition mission operations data by operational *session*. While partitioning can be achieved physically, in a distributed network environment it is preferable that operational services are defined in such a way that *session* is explicit to avoid any possibility of confusion, and to enable data to be combined in a single system.

### 3.3.5.2 Operations Domains

Mission operations does not always simply relate to the control of a single spacecraft. Many existing space agencies and missions require the control of multiple remote assets (spacecraft fleets and constellations, ground stations, etc.).

In order to ensure that unique referencing of operational entities and data items is possible, the concept of a hierarchy of system components or operational *domains* is required. This is used to scope the frame of reference of monitoring and control (e.g., agency>mission>satellite>subsystem). It provides a framework for the control of *namespaces for operational data*, such as telemetry monitoring parameters and telecommands.

Hence commands are represented with a full referential context. Command X1234 'Heater D On' becomes:

AgencyA.MissionB.SatC.X1234 *or even* AgencyA.MissionB.SatC.HeaterD.ON

which cannot inadvertently be sent to AgencyX.MissionY.SatZ and executed.

Within a specific detailed operation, the *domain* may be contextually implicit to allow generic (multi-domain) operations to be defined and to ensure that the specification of operations is not unduly verbose.

### 3.3.5.3 Security and Access Control

To ensure that only authorised consumers have access to mission operations services, it is critical that some form of client authentication is an integral part of service definitions. To avoid the need for a client to support multiple authentication methods, it is highly desirable that all mission operations services use the same mechanism and that client authentication is only required once per client 'login' even if multiple services are used.

It is expected that the concept of roles and associated privileges would also be required where the operations of the services would require a certain privilege or role to be held by the operator. In multi-user environments there is often the support for privilege transfer and this should also be taken into consideration. This information would be a configuration controlled aspect of the system.

Where services are supported over open or public communications paths, then a level of security is required to avoid unauthorised access or intrusion. Services must be defined in such a way as to allow them to make use of secure communications channels. These security

services are expected to adhere to the recommendations made by the CCSDS Security Working Group.

#### **3.3.5.4 Operational Responsibility**

There are cases where a remote or automated mission operations application requires an input or authorisation from the responsible member of the operations team. This may be necessary to deal with asynchronous operational decision points, alarms or failure conditions.

For such interactions to be routed to the appropriate user role, there is a need for a responsibility model that can be used to associate responsibility for a particular node of the *domain* hierarchy with a particular user and his/her current location. Such interactions must also be re-routed where the current responsible user logs out or there is an equipment failure.

It may be that this mechanism is entirely encapsulated within the proposed Operator Interaction service, although it could also be supported within Alert Notification and failure notifications for other services.

#### **3.3.5.5 Quality of Service**

Quality of Service relates to the provision of different levels of service or performance guarantee that an operational function or service may offer. Issues that fall within this include:

- prioritisation—methods by which support for service clients can be prioritised in order to guarantee control actions (e.g., commanding) for critical applications, or a minimum delay for monitoring data provision;
- bandwidth management;
- delivery guarantee;
- error management—retransmission, etc.

A given service provider need not offer all QOS levels, or may provide a restricted set over restricted bandwidth communications paths. Means must be provided to determine available QOS levels and to negotiate for required levels of service during connection establishment.

### 3.4 IDENTIFIED MISSION OPERATIONS SERVICES

#### 3.4.1 GENERAL

The SM&C Protocol and Common Services layers have been described previously.

The following table lists the application level Mission Operations services that have currently been identified. It is to be stressed, however, that the service framework is designed to be extensible and additional services may be identified in the future to address additional requirements for end-to-end interaction in mission operations.

The Mission Operations Services are illustrated in figure 2-6.

**Table 3-3: Mission Operations Services**

<b>Name</b>	<b>Service Objects and Operations</b>	<b>Priority</b>
<b>Core Monitoring &amp; Control</b>	<b>Parameters:</b> publish status; set <b>Actions [Commands]:</b> publish status; invoke/send <b>Alerts [Events]:</b> notify; raise	1
<b>Time</b>	<b>Time:</b> report; set; correlate; notify	2
<b>Software Management</b>	<b>On-board Software:</b> load; dump	2
<b>Planning Request</b>	<b>Planning Request/Goal:</b> request; response	3
<b>Scheduling</b>	<b>Schedule:</b> distribute; edit; control; progress reporting	3
<b>Automation</b>	<b>Procedure/Function:</b> control; progress reporting	2
<b>Data Product Management</b>	<b>Data Product [Payload Data File]:</b> directory; transfer	3
<b>Location</b>	<b>Position:</b> tracking, ranging, onboard positioning	3
<b>Flight Dynamics</b>	<b>Orbit/Attitude/Predicted Events:</b> determination, propagation, manoeuvre preparation	4
<b>Operator Interaction</b>	<b>Message/Alarm/Query:</b> notify; operator response	4
<b>Remote Buffer Management</b>	<b>Buffer:</b> catalogue; retrieve; clear	4
NOTE – Services are listed together with a summary of the associated Service Objects and Operations. Service definition is prioritised: the Core M&C Service is highest priority; work is in progress on priority 2 services; remaining services for future definition are grouped as priority 3 and 4.		

In accordance with the approach outlined in 3.2, a service has been identified for each of the principal types of mission operations information identified in table 3-2, which also lists the potential providers and consumer functions of each service. The following paragraphs give a summary description of each of the identified services.

### 3.4.2 CORE M&C SERVICE

The Core M&C service provides basic monitoring and control capability through three basic classes of information:

- **Parameters** provide status monitoring capability.
- **Actions** allow control directives to be invoked and their evolving status monitored: spacecraft telecommands are an example of an action.
- **Alerts** provide a mechanism for asynchronous notification of operationally significant events or anomalies by the service provider to the service consumer.

Each of these is supported by a separate sub-service that follows the generic interaction pattern for mission operations (*observe*, *control* and *manage*) described in 3.3.3.3. The Core M&C service configuration data identifies the set of parameters, action definitions and alert definitions that exist for a given service instance (e.g., spacecraft).

#### **Parameters**

Parameters are static objects: the object instance is fully defined by the definition contained in the service configuration data. Parameters have a continuous existence, with a state that evolves over time.

Parameters may be considered to have multiple observable status attributes, depending on the capability sets used, for example:

- Raw Value;
- Calibrated or Interpreted Engineering Value;
- Monitored Behaviour Check Violations;
- Statistics.

Parameters may also be aggregated into groups that can be referenced by name. By registering interest in an aggregation, the consumer is guaranteed to receive a coherent set of parameter states.

Control operations on Parameters include the ability to set the Parameter value. This may be used to reset counters, or to allow parameters to be used as commonly accessible operational modes or run-time configuration options.

## **Actions**

Actions are dynamic objects: each invocation of an action creates a new object instance whose state evolves over a limited lifetime. Each action definition may have an associated set of action arguments, which can be set at invocation time. Multiple copies of the same action may be active in the system at the same time.

Control operations on Actions include:

- Create Action and set Arguments (if any);
- Perform Pre-Transmission Check on Action;
- Transmit Action.

Actions evolve through various stages of verification status. For example, these stages could include:

- Action Invocation;
- Pre-Execution Validation Check;
- Transmission Status;
- Verification of Receipt and/or Execution by Target;
- Post-Execution Verification.

## **Alerts**

Like actions, alerts are dynamic objects with arguments. Consumer applications register interest in alerts via the *observe* interface.

Control operations on Alerts include:

- Raise Alert, through which a consumer function can inject an Alert;
- Enable/Disable Alerts.

### **3.4.3 TIME SERVICE**

Time correlation between on-board clocks and the system reference time is required to support mission operations, for basic M&C purposes, on-board scheduling and flight dynamics.

The time service includes the following operations:

- Report Time;
- Correlate Time;
- Set Time;
- Configure rate of Time Report generation.

### 3.4.4 SOFTWARE MANAGEMENT SERVICE

This service supports the management of software loaded into the remote system. At present, the service definition reflects the fact that most on-board software is managed in terms of binary images that are stored at known memory addresses. Future versions of the service may need to address on-board file-based systems. The operations supported include:

- Load Software Image;
- Dump Software Image;
- Check Software Image.

### 3.4.5 PLANNING REQUEST SERVICE

The Planning Request service allows a consumer application to raise a request (and responses) for an operational task or goal to be included in a plan. The service would be provided by a Mission Planning application; potential consumers include:

- Operations Team (via HCI);
- External Users (Principal Investigators, Mission Exploitation System, End Users);
- Flight Dynamics (manoeuvre requests);
- Software Management (software load requests).

The service definition does not prescribe the planning process or algorithms used by the Planning application. Nor does it address automation of the planning process, which can be achieved by the Planning application exposing the Core M&C service.

### 3.4.6 SCHEDULING SERVICE

The Scheduling service is one of two services that support automation of mission operations. Scheduling is concerned with the distribution, monitoring and control of scheduled timelines of mission operations intended for automated execution.

The service provider is an application capable of executing the schedule, whether ground-based or on-board the spacecraft. Consumer functions include Planning applications that generate schedules and require feedback on their execution, and Manual Operations displays that allow interactive monitoring and control of schedule execution.

The essential service object is a **schedule** which is a container for individual items that appear on the schedule timeline, including:

- Predicted **Events**;
- Planned **Contacts** (periods of connectivity between space and ground);
- Scheduled **Tasks**, potentially containing multiple **Activities**.

Predicted Events correlate to those notified via Flight Dynamics services, Scheduled Tasks correlate to a source Planning Request, while Activities correlate to individual Procedures or Functions that can be initiated via the Automation service.

Events, Tasks and Activities are defined in the service configuration data.

A Planning application generates a schedule for delivery via the service to the Schedule Execution function, it can also receive schedule status in return. A Manual Operations client can control the execution of the schedule and observe its dynamically evolving execution status via the service. The service also supports insertion, deletion and modification of individual scheduled items.

### 3.4.7 AUTOMATION SERVICE

The Automation service is the second of the two services that support automation of mission operations. The service provider is an application capable of executing pre-defined Procedures (or autonomous Functions), whether ground-based or on-board the spacecraft. Consumer functions include Schedule Execution and Manual Operations displays.

The essential service object is a **procedure** (or function) which may act as a container for constituent elements, including:

- Threads of execution control within the procedure;
- Execution Steps;
- Executable Actions.

The constituent Procedure Threads and Step objects are dependent on the model of procedure execution employed in the procedure definition. Executable Actions correlate to Core M&C Service actions, or the Control sub-service of any other Mission Operations Service.

Schedule Execution or Manual Operations may invoke and execute a new instance of a procedure via the service. Subsequent control over the procedure may be exercised: to suspend, resume or stop its execution; and to perform manual control over its execution. The execution status of the procedure can also be observed: either at the level of the procedure object itself, or potentially at a lower level of detail in terms of the procedure model.

### 3.4.8 DATA PRODUCT MANAGEMENT SERVICE

The Data Product Management service is concerned with the management and transfer of sizeable binary data products. This is typically observation or payload data gathered on-board the spacecraft, but it could also be used to manage the output of analysis and reporting functions.

The principal service objects are **Products** (Files) and **Folders** (Directories).

The service supports operations to:

- obtain explorer-style directory tree listings of the content of the remote data product store;
- transfer products in both directions (a service consumer can both get products from and put products into the remote data product store);
- perform data product store management operations, such as delete, move, rename, etc.;
- provide information about changes to the on-board data product store, such as new product events, etc.

It is anticipated that this service would be a thin layer over CFDP or FTP.

### **3.4.9 LOCATION SERVICE**

The Location Service supports the provision of spacecraft positioning information. This includes:

- Position reports (e.g., from on-board GPS);
- Spacecraft Ranging and Range Rate measurements (e.g., from ground station ranging or laser ranging equipment);
- Antenna Tracking azimuth and elevation (e.g., from ground station in auto-track mode).

The provider is either the spacecraft or a ground station facility. In the latter case, there is overlap with already identified Space Link Extension services. Any service definition in this area will be fully coordinated.

The service also includes control operations required to initiate or control the frequency of acquisition/delivery of positioning measurements.

### **3.4.10 FLIGHT DYNAMICS SERVICES**

Flight Dynamics services are concerned with the provision of information classes that are specific to Flight Dynamics. These include:

- Orbit Vectors;
- Attitude Vectors;
- Predicted Orbital Events (including Ground Station visibilities);
- Physical State;
- Fuel Budget Assessments.

It is noted that a Flight Dynamics application may use other Mission Operations services to support aspects of its function. For example:

- Planned manoeuvres may be communicated to other applications via the Planning Request or Scheduling services;
- Automation Flight Dynamics tasks can be achieved by the application exposing the Core M&C or Schedule Execution services.

### **3.4.11 OPERATOR INTERACTION SERVICE**

Ultimate responsibility for mission operations is vested in a team of people. The interactive user interfaces provided by the Mission Control System allow this team to monitor current operations status and initiate control actions. There have always been cases, however, where it is necessary for the automated systems to alert the human operators to a significant event; typically this is achieved through the raising of alarms.

With increasing automation, however, there are occasions where the local or remote system needs to request authorisation, solicit input or request an operational decision. This constitutes an asynchronous operator interaction, where the system is invoking the operator, rather than the other way round.

Standardisation of an Operator Interaction service would allow any automatic function to initiate interactions in the same way, irrespective of the local implementation of the mechanisms to achieve this (audible alarms, pop-up dialogues, pagers, etc.).

The following classes of operator interaction are identified:

- Log: message logged, but not forwarded to operator;
- Notification: message forwarded to operator, no response required;
- Acknowledgement: operator must acknowledge;
- Option Selection: operator must select from list of options;
- Request Input: operator must respond with valid input data.

Interactions may also be associated with a severity that may, for example, indicate that an audible signal or alarm should be sounded.

Routing of interactions is a matter for implementation, but may make use of external services such as pagers or SMS text messaging to alert staff not present in the control room.

### **3.4.12 REMOTE BUFFER MANAGEMENT SERVICE**

As the connection to the spacecraft may be intermittent, many missions implement intermediate buffering of data either on-board the spacecraft or within a ground station. Such remote buffers may contain messages relating to any or all of the services described above.

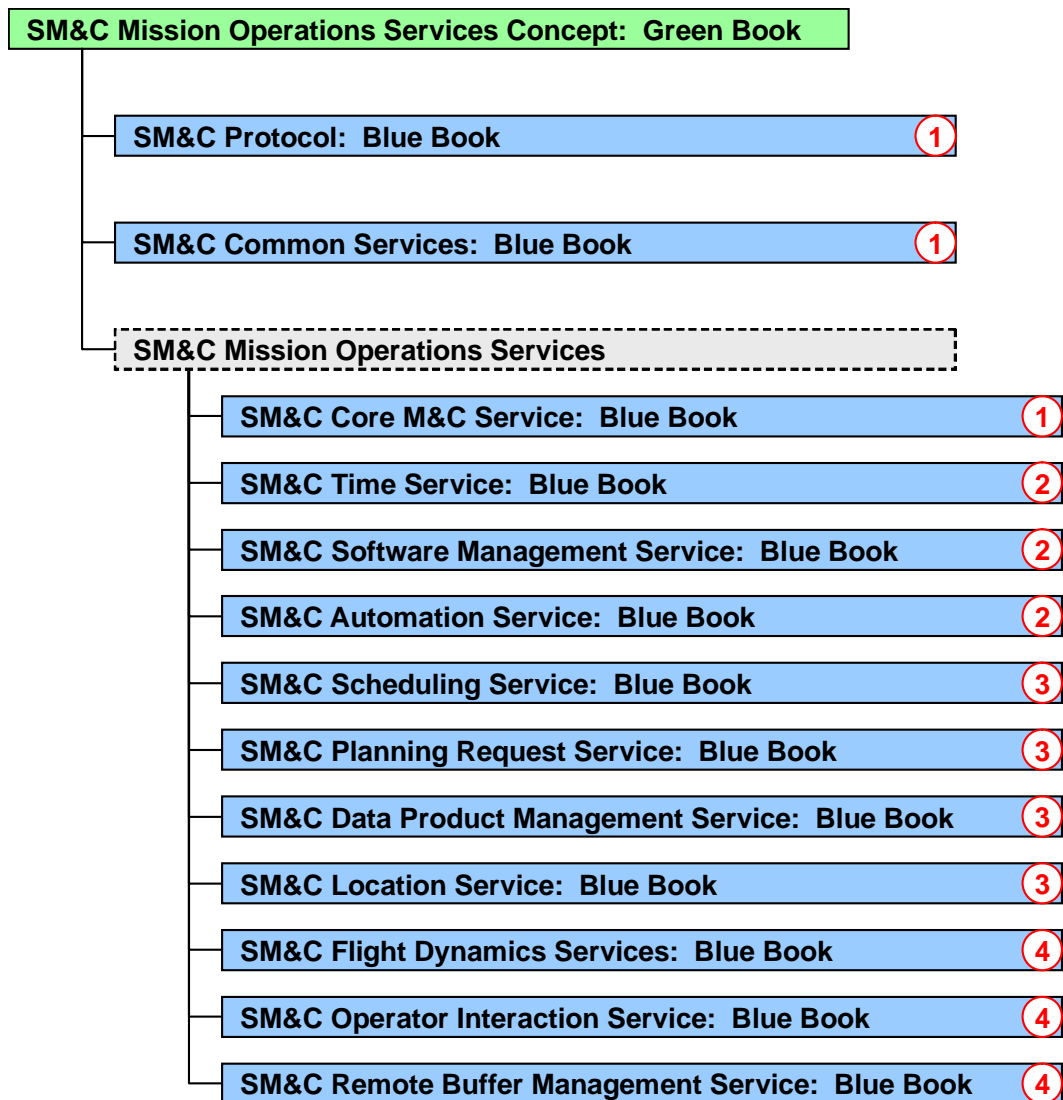
The purpose of the Remote Buffer Management Service is to provide a standardised approach to the operation of such buffers. Supported operations include:

- Obtain Catalogue of Buffered Data;
- Retrieve Data from Buffer;
- Delete Data from Buffer;
- Clear Buffer.

## 4 DOCUMENT ROADMAP

This section summarises the set of standards documentation proposed in support of the SM&C Mission Operations Services Concept.

The following diagram presents the proposed documentation tree, together with their prioritisation for production:



**Figure 4-1: Document Roadmap**

NOTE – The SM&C Mission Operations Services Concept is an Informational Report. It proposes the specification of standards corresponding to 3 framework layers: SM&C Protocol, Common Services and Mission Operations Services. Each of the first two layers will be specified in a single Blue Book. The specification of each Mission Operations Service will constitute a separate Blue Book. The roundels show the prioritization for the production of standards.

## ANNEX A

## DEFINITION OF TERMS

Term	Definition
<i>action</i>	An atomic (non-interruptible) control directive of mission operations (equivalent to a telecommand or ground segment directive) that can be initiated by manual or automatic control sources, via the Core M&C service. An action may have <i>arguments</i> and its evolving status can be observed. An action can typically apply pre- and post-execution checks.
<i>active service interface</i>	The on-line aspect of a service (as opposed to off-line configuration or access to <i>service history</i> ). The active interface between the Service Provider and Service Consumer instances, where <i>events</i> and <i>operations</i> are exchanged. These allow the Consumer to monitor (or observe) status and effect control.
<i>activity</i>	An automated mission operations function, typically an operations procedure, batch task or other software process. An activity can be individually scheduled or initiated. In principle, an activity is non-atomic, has duration, and can be controlled once initiated. An activity may have <i>arguments</i> and its evolving status can be observed. An activity may generate multiple <i>actions</i> , and its behaviour can be dependent on status observed at run-time.
<i>adapter</i>	An adapter (in an SOA context) is a software component that implements a higher level service in terms of a lower-level service or specific technology. In this way different adapters can map a high-level service on to different underlying technologies, transparently to all higher layers including the application. Adapters can also wrap non-service oriented applications so that they can be used as Service Providers in service oriented architecture.
<i>alert</i>	An asynchronous notification, such as a non-nominal event, of significance to mission operations. Alerts may be used to notify such events to operators, initiate an automatic response, or synchronise asynchronous functions. Alerts may have <i>arguments</i> .

Term	Definition
<i>application programmers' interface</i>	The definition of the exposed or 'public' interface to a software component that can be used by another software component. In an SOA context, an API corresponds to a language- or technology-specific implementation of a <i>service access point</i> . This constitutes the code classes, types and functions utilised by a programmer when implementing the <i>service provider</i> and <i>service consumer</i> .
<i>argument</i>	A run-time parameter provided to various control items on invocation, e.g., telecommand arguments. Arguments apply to <i>actions</i> , <i>activities</i> and <i>alerts</i> among other items.
<i>capability set</i>	A grouping of functions offered by a <i>service</i> that are logically related. Capability sets are used to decompose a service into smaller functional areas. They are used to identify aspects of a service that are mandatory or optional for a given implementation.
<i>(SM&amp;C) Common Services</i>	The common service layer that sits between the SM&C Mission Operations Service and Protocol layers.
<i>consumer, service consumer</i>	A software application that uses a <i>service</i> being supplied by a <i>service provider</i> . An individual software application can act as the consumer of multiple services.
<i>control (interface)</i>	A generic interface that supports the initiation by a <i>service consumer</i> of <i>operations</i> supported by the <i>service provider</i> , that constitute routine use of a <i>service</i> . See also <i>manage</i> .
<i>domain</i>	A namespace that partitions separately addressable entities (e.g., <i>actions</i> , <i>parameters</i> , <i>alerts</i> ) in the space system. The space system is decomposed into a hierarchy of <i>domains</i> within which entity identifiers are unique.
<i>dynamic object dynamically instantiated object</i>	An entity that is instantiated, invoked or created, at run-time based on a static definition. Examples include <i>actions</i> , <i>alerts</i> and <i>activities</i> . Multiple copies (instances) of such objects may exist concurrently, but share a single definition.
<i>encapsulation</i>	A software design approach that provides code users with a well-defined interface to a set of functions in a way which hides their internal workings or means of implementation. In object-oriented programming, the technique of keeping together both data structures and the methods (procedures) which act on them.

CCSDS REPORT CONCERNING MISSION OPERATIONS SERVICES CONCEPT

Term	Definition
<i>event</i> ◆	A time-stamped message containing (changes in) information about information objects associated with a <i>service</i> , that are exchanged across <i>service interfaces</i> and potentially stored in <i>service history</i> .
<i>exposed interface</i>	A published (or ‘public’) interface provided by a software component that is available for use by other software components.
<i>manage (interface)</i>	A generic interface that supports the initiation by a <i>service consumer</i> of <i>operations</i> supported by the <i>service provider</i> , that constitute run-time configuration of a <i>service</i> , typically modifying the on-going behaviour of the <i>service provider</i> . See also <i>control</i> .
<i>(SM&amp;C) Mission Operations Services</i>	A suite of end-to-end application level services that constitute a service-oriented architecture for space mission operations.
<i>object/ information object/ service object</i>	In SM&C, information objects are passed across a <i>service interface</i> . These are defined in the information model of the service.
<i>object instance</i>	Alternative term for <i>object</i> that distinguishes between the multiple run-time invocations of an object and their associated static definition (see <i>statically</i> and <i>dynamically instantiated objects</i> ).
<i>observe (interface)</i>	A generic interface that supports the provision of status information by the <i>service provider</i> to the <i>service consumer</i> . This is achieved by subscription of the <i>service consumer</i> to a flow of status update <i>events</i> relating to the information objects associated with the <i>service</i> .
<i>operations</i> ◆	In object-oriented programming, the methods / functions / messages defined for a Class of Objects. Specifically in the <i>mission operations services</i> context, the control primitives that can be performed across the <i>service interfaces</i> .
<i>parameter</i>	An item of mission operations status information that can be individually subscribed to by a <i>service consumer</i> , via the Core M&C service. A parameter has multiple attributes, including: raw value, engineering value, validity, check status and (optionally) statistics.

Term	Definition
<i>plug-in</i>	A software component that can be integrated with other components conforming to the same service-oriented architecture, without the need to modify the implementation of other components. In the SM&C context, this could apply to both <i>service consumer/provider</i> applications and infrastructure components that implement lower levels (protocol layers) of the <i>service interface</i> .
<i>protocol data unit</i>	Elemental data message for exchange between peer service layers of two applications using a particular implementation protocol.
<i>provider, service provider</i>	An application that publishes a <i>service</i> , exposing the <i>service interface</i> , while hiding details of its implementation.
<i>proxy</i>	In the context of SM&C, a proxy function or component is one that acts locally in the place of a remote <i>service provider</i> , such as a spacecraft. There is a proxy function for each <i>service</i> . It provides a dual role. Firstly it provides a permanent point of contact for <i>service consumers</i> where the link to the remote <i>service provider</i> is intermittent, maintaining an image of current status, buffering <i>operations</i> and managing the <i>service history</i> . Secondly it can act as an isolation layer and adapter to actual protocols employed on the space-ground interface.
<i>replay</i>	The act or interface associated with viewing data from a <i>service history</i> in the same manner as live operation (via an <i>observe</i> interface). Service <i>events</i> are dynamically replayed over an evolving time period.
<i>retrieval</i>	The act or interface associated with of withdrawing a data set by a time range from a <i>service history</i> . Retrieval is mainly intended for fast access to a block of service history for display of data trends or logs over a period of time, or to be used in analytical tasks.
<i>service, service interface</i>	An abstraction of a function within a service oriented architecture, as an exposed public interface, which allows independently developed applications to interact in a standard manner. Services should encapsulate concisely, representing only those information objects and operations needed at the abstract service interface. They should be platform and implementation independent.

Term	Definition
<i>service access point</i>	A specification of the <i>service interface</i> used by <i>service consumer</i> and <i>service provider</i> applications, expressed in implementation independent terms. This is realised (implemented) by an <i>application programmers' interface</i> (API).
<i>service configuration data</i>	Configuration data (in the form of a database or other file) that defines the characteristics of a specific instance of a service. Typically this identifies the information objects that exist in the context of a particular service, for a particular <i>domain</i> , e.g., the specific <i>actions</i> , <i>parameters</i> and <i>alerts</i> applicable to a given, would be defined in the Core M&C service configuration data for that spacecraft. Access to the service configuration data is required by both <i>service consumer</i> and <i>service provider</i> (or its <i>proxy</i> ).
<i>service data unit</i>	Elemental data message for exchange between two applications using a particular service, across the <i>service access point</i> interface. This may correspond to an <i>event</i> or <i>operation</i> associated with a service information object.
<i>service directory</i>	A service directory is an entity that provides publish and lookup facilities to <i>service providers</i> and <i>consumers</i> .
<i>service history</i>	The operational data archive for a <i>service</i> . This is the data required to reconstitute a historical view of information at the <i>service interface</i> , either using <i>replay</i> or <i>retrieval</i> access methods. It corresponds to the persistent sequence of all service <i>events</i> over a period of time, to which a <i>service consumer</i> could have subscribed. Examples of service histories include <i>parameter</i> history, <i>action</i> history and <i>alert</i> history. Alternative implementations are possible, based on archiving of protocol messages (e.g., packets) and re-processing.
<i>service instance</i>	A deployment copy of a <i>service</i> , typically for a specific <i>domain</i> . A <i>service provider</i> constructs and publishes a service instance. <i>Service consumers</i> may then subscribe to that service instance.

Term	Definition
<i>session</i>	In SM&C, a session defines the time-frame for a service. A session may be live or historical, real or simulated. A <i>service consumer</i> may join any existing session by subscribing to a service for that session. Within a given system there may be multiple concurrent sessions, to support simulated and/or historical replay sessions in parallel with live operations. Within <i>service history</i> there may be multiple session histories, corresponding to live operations and simulated sessions.
<i>static object</i> <i>statically instantiated object</i>	An entity that is effectively instantiated at operations preparation time, e.g., a <i>parameter</i> . It has a static portion (the definition) and a dynamic portion (its current status). See also <i>dynamically instantiated object</i> .