Colouring key:

[Some text]

A non-UML diagram, should not be formal but illustrative. For example a diagram showing book relationships.

[Some text]

A UML diagram illustrating some aspect of the MO architecture or a deployment.

Some text

A cross reference or reference to another specification.

Some text

Some text that needs to be written.

# Introduction

Mission Operations (MO) Services is a standardised set of service specifications and supporting software technologies which aim to enable significant improvements to the operational capabilities of space missions. The broad collection of services supports both interoperability between Agencies and enables simplification and software reuse inside Agencies.

Whilst MO can be effectively applied in relatively small use cases, the scope of MO is potentially large and can therefore involve a wide variety of stakeholders. This Green Book is intended to provide an introduction to MO for all stakeholders.

This Green Book begins with a description of the context for the creation of MO in the form of the motivation and rationale for the services. This is done in the case of four concrete case studies, which are used as examples for the use cases which led to the development of MO and the benefits it can offer to mission development, operability and effectiveness. The following section then presents the key concepts of MO, deliberately set out at a high level to make the description as accessible as possible.

Further sections in this Green Book then offer greater levels of detail about MO from different perspectives. It is anticipated that these sections will be applicable to different stakeholders, and readers may not want, or need, to read all sections.

## MO Stakeholders

Broadly speaking, we identify the following stakeholders for MO.

* **Managers**, who we assume are key decision makers and determine technology adoption and use. We use the Manager stakeholder broadly for anyone with management responsibility across procurement, development and/or operations.
* **Operators**, whether involved in platform or payload operations (or both).
* **System Integrators**, who are responsible for integrating and delivering complete hardware. A good example of this is the System Integrator who is responsible for delivering a complete spacecraft or space segment. A ground segment integrator would also be System Integrator.
* **Application Software Developers**, who are those responsible for providing software at an application level (i.e. software involved in delivering the main mission-related functions), across both flight and ground segments. Application software is likely to be mission-specific, or at least related to a specific platform, payload or concept of operations.
* **Infrastructure Software Developers**, who are those responsible for providing the underlying software framework or environment which may be used by application software developers. This software is much less likely to be mission specific. The distinction between application and infrastructure is not necessarily exact but the separation from application software is useful in identifying the parts of MO that form part of the underling infrastructure and are more likely to be reused from mission-to-mission.

This is not an exclusive list, but is intended to help the reader navigate both this Green Book and the set of standards which define MO. Section 9, at the end of this Green Book, indicates where the reader should go next by providing an overview of the other MO standards documents.

## Reading and Using this Green Book

The intention is that the first three sections of this Green Book, including this one, are applicable to all audiences. As such, it is recommended that all stakeholders read these sections:

* Section 2 (“Mission Operations Motivation and Rationale”)
  + Provides the context and rationale for MO, laying out the problems that MO seeks to address.
* Section 3 (“Mission Operations Concepts”)
  + Presents the technical concepts that MO uses in order to solve the problems identified in Section 2. At a conceptual level, Section 2 poses the questions, and Section 3 provides the answers.

Further detail as to exactly *how* MO implements the concepts described in Section 3 is provided in subsequent sections, starting with Section 4.

It is suggested that readers approach the Sections from 4 onwards depending on their interest, and which stakeholder they most closely identify with.

* **Managers** should consider reading Section 7 (“Managing and Administering MO Systems”). If further detail is required, Section 4 (“Mission Operations Technology”) may be of interest.
* **Operators** should consider reading Section 4 (“Mission Operations Technology”). Beyond this, relatively high level, overview, further reading depends on interest.
* **System Integrators** should consider reading Section 4 (“Mission Operations Technology”) and Section 6 (“Integrating MO Systems”).
* **Application Software Developers** should consider reading Section 4 (“Mission Operations Technology”) and Section 5 (“Developing MO Applications”).
* **Infrastructure Software Developers** require a fuller understanding of MO and, as such, we recommend that these stakeholders consider reading Section 4 (“Mission Operations Technology”), Section 6 (“Integrating MO Systems”) and Section 8 (“MO Communications”). Additionally, Infrastructure Software Developers may find it valuable to gain the perspective of an Application Software Developer by reading Section 5 (“Developing MO Applications”).

It is suggested that all stakeholders then use Section 9 to determine the most appropriate standards to use for further work, if required. All readers are encouraged to read further sections beyond those listed above, in order to gain a more rounded perspective of MO.

# Mission Operations Motivation and Rationale

This section introduces the context for Mission Operations Services. We illustrate this with some concrete examples of mission which would benefit from the use of MO. From these examples, we summarise the main challenges faced by missions in improving operability and development for more complex scenarios. These challenges are used to derive a concrete set of User Needs for MO.

## Background and Context

Operating space assets is complex, and the complexity of space missions is increasing. This complexity comes from a number of sources including:

* greater need for inter-operability between systems, often provided by different organisations or agencies;
* an increase in the number of organisations or agencies involved in operations;
* mission composed of a great number of systems, such as multiple spacecraft, distributed ground segments and combinations of orbiters and landers;
* introduction of operations earlier in development, such as during spacecraft assembly, integration and test (AIT);
* the need to cope with increasingly rapid technological change and equipment obsolescence across the operational period of long-duration missions;
* pressure to increase the efficiency and cost-effectiveness of operations, especially during “routine” phases, through increased automation.

A key issue at the heart of many of these sources of complexity is that of interfaces: interfaces between functions, technologies and organisations which must be developed, communicated and maintained across the long life-span of a mission under time and cost pressure. Common interfaces can ease some of this pressure, but there is frequently a lack of interface standardisation for mission operations, both within and between organisations. Coupled with this lack of standardisation is a lack of reuse which arises from many different sources, including changes in software and hardware technology, and changes in available skills of engineers. These challenges together have typically resulted in increased costs for development, deployment and operator training.

To place this complexity in context, the following sections present four example mission scenarios which range in overall complexity:

* an agency Earth science mission carrying a hosted payload from a different space agency;
* a multi-agency radio telescope mission, requiring the coordinated use of both space-based and ground-based resources;
* a multi-agency lunar exploration mission involving ground-based resources, spacecraft in both terrestrial and lunar orbits, and lunar surface-based assets;
* an Earth observation constellation comprising a large number of small satellites each with a limited lifetime, requiring ongoing development and replenishment over the mission lifetime.

### Example 1: Inter-Agency Hosted Payload

A national space agency, Agency A, commissions and launches an Earth science mission which hosts an instrument provided by Agency B. The science team for the hosted payload passes operational plans for their instrument to the operations team at Agency A. Science data is then returned to the science team, together with platform telemetry which is necessary to interpret the instrument data. It is found that telemetry from the platform, as well as the main instrument, indicate good conditions for operating the Agency B payload. It is therefore useful for the Agency B science team to be able to monitor platform/payload telemetry to guide operations for the hosted payload. A live feed of telemetry is required, an interface must be defined and agreed between Agencies because of the different technologies used in Agencies A and B, and the process for requesting operations for the hosted payload may be manual and slow, leading to missed science opportunities.

[Diagram showing mission in here]

The key challenges of this use case are:

* the sharing of housekeeping telemetry, in both directions, between organisations in a standard way for use in spacecraft and payload operations such as for health and accountability;
* sharing operational plans/schedules between organisations in a standard way,
  + for use in Agency A and B planning and scheduling activities and
  + for collaboration of spacecraft planning and scheduling such as the building of realisable and integrated schedules for uplink which includes negotiation of resources and the resolution of conflicts;
* the delivery of mission products such as orbit data and instrument payload data products from Agency A to Agency B.

This list of challenges is non-exhaustive.

### Example 2: Radio Astronomy

A multi-agency spacecraft is flying a payload which can be used for radio astronomy. The most effective science data can be achieved by coordinating space-based measurements with those taken from several other sources, space-based and possibly also ground-based. As such, an observation may require the coordination of the spacecraft as well as three or four other space- and ground-based observatories. As these resources are controlled by a number of different organisations which may only be part time contributors to the mission (i.e. brought in during specific observations), the planning and coordination effort is substantial and potentially slow. This may lead to missed opportunities for observation and inefficient and expensive operations.

[Diagram showing mission in here]

The key challenges of this use case are:

* broader collaboration between more organisations than the first use case;
* creating and coordinating operational plans/schedules involving multiple space and ground assets belonging to multiple organisations in a responsive manner, even though some assets may be part time contributors to the mission;
* distributing science and associated platform/system data to multiple organisations in a standard way.

As before, this list of challenges in non-exhaustive.

### Example 3: Lunar Exploration Mission

A multi-agency mission is created to explore the lunar surface in great detail, especially focusing on the lunar poles and the far side of the moon. The main exploration vehicle is a pair of lunar rovers which have a range of tools and instruments. Communications with the rovers is possible through the use of a lunar-orbiting satellite which acts as a data relay when either of the rovers is out of sight of Earth. When in view of the Earth, the communications signal to the rovers, and therefore the bandwidth, can be improved through the use of Earth-orbiting data relay satellites and a number of ground stations. There will be a single Mission Operations Centre for the mission which must coordinate the aspects of the mission, across flight and ground. The coordinated approach to operations lowers costs in development, operator training and maintenance.

The mission comprises a large number of resources across space and ground with many complex planning tasks to be coordinated across multiple agencies and organisations. When all phases are considered, the mission is also expected to be of significant duration with the need to maintain, and potentially improve, all systems across that time period.

[Diagram showing mission in here]

The key challenges of this use case are:

* integrating a number of complex and capable systems into a single large system-of-systems in order to achieve mission objectives;
* achieving a single coherent operational concept across diverse, distributed systems provided by different agencies;
* maintaining a large, complex system comprising many space and ground assets for a long period of time, both during development and operations.

This list builds on those of previous use cases and is non-exhaustive.

### Example 4: Small Satellite Earth Observation Constellation

A responsive Earth observation system is to be created using a constellation of between 50 and 100 small satellites with a miniaturised instrument pack and a network of ground stations. The satellites will be placed in relatively low orbit, and have limited station-keeping capability, giving an orbital lifetime of 3-5 years. The overall mission is expected to last at least 10-15 years.

The mission intends to take advantage of this by creating a modular space system which will be continuously evolved over the mission lifetime. Functionality will be scalable, and capabilities will be added in response to changing mission demands. This responsiveness requires the rapid development, deployment and configuration of both space and ground systems and continual improvement rather than one-off (“big-bang”) developments.

Once successful, the approach taken by the mission can be spun out to other missions, even those only utilising a single spacecraft, in order to lower development costs, time and risk.

[Diagram showing mission in here]

The key challenges of this use case are:

* effectively meeting the operational needs of an evolving, heterogeneous system;
* extending the coherent conceptual operational framework to development of flight and ground software;
* using modularity to ease system evolution and reuse;
* creating maintainability through change, rather than maintenance of a largely static system;
* scaling to meet the needs of a distributed system with a very large number of assets;
* proving an approach which can, if desired, be applied to a wide range of missions.

These challenges build on those of previous use cases.

## User Needs

The challenges posed by the four example use cases span all aspects of the mission, from ground to space and from development through to operations. What is needed is a solution to help integrate currently disparate and both physically and functionally distributed elements of a mission in a way which helps reduce complexity. This integration is visible at simple level, for example at a single interface between agencies as in the first example use case, or at a much more complex level in the third example use case, where a large number of interacting systems are within scope.

Although it may be possible to develop bespoke solutions to each of the examples presented, we consider that the long-term cost of this is too high and a preferable solution is to develop a suitable, standards-based technology which will address the key user needs for multiple future missions.

User needs for this technology, i.e. MO, fall into 4 broad categories:

* **Inter-Operability and Intra-Operability**. There is an increasing need for greater cooperation between different agencies which, together with the increasingly complex operational needs of operators with multiple satellites of different types and sources, contributes to the rise in systems-of-systems architectures. This need for cooperation applies not only during operations, but also during development, assembly, integration and test of both flight and ground systems.
* **Space System Concept**. Conceiving the space and the ground segments as one single, distributed system, or system-of-systems, is important in being able to obtain more operational effectiveness from missions and meet the operational challenges of new mission types. Viewing the system this way, as opposed to two distinct systems with a limited interface, leads to considerable change of the development lifecycle of ground and space data systems.
* **Standards-Based Modularity**. The ability to bring together parts of a complete system, each developed separately, is essential for enabling multiple organisations to work together on complex missions. Basing this modularity, and the interactions between system parts on standards allows investment in infrastructure and would lower per-mission costs. Maintaining standards across missions also opens the door to reuse, potentially enabling common functions and capabilities to be applied to multiple missions with minimal effort or adaptation.
* **Technology Independence and Long-Term Maintainability**. The long-term maintenance of systems for the duration of the space mission, and the need to upgrade or replace components for various reasons, including technical obsolescence, are significant challenges.

The key user needs for MO have been explicitly enumerated below.

### Inter-Operability and Intra-Operability

1. **Interoperability.** It should be possible to interact with mission functions through a well-defined interface from “outside” of the system. This permits external interfaces to existing and separate systems.
2. **Location Independence.** The conceptual way that mission functions are used should be independent of where those functions are located, whether in space or on the ground. Although there will obviously be impacts from the communications capability with remote locations, such as bandwidth and availability, the basic operational concept should be location-independent.
3. **Separation of Functionality from Infrastructure.** Functions which achieve mission-level goals should be kept separate from the infrastructure that allows those functions to be used. This means that, for example, communications capabilities should be treated as infrastructure.
4. **Operability.** As an operator, whether human or machine, it should be possible to access all mission functions and to control their behaviour.
5. **Observability and Traceability**. As an operator, whether human or machine, it should be possible to observe the behaviour of all mission functions and to track/record the progress of operations across both local and distributed systems.
6. **Data Intensive Operations**. It should be possible to carry out operations which involve the generation and management of large and small data products.
7. **Autonomy**. It should be possible to automate the operation of system functions such that they may be carried out without human interaction. The level of autonomy necessary will be dependent on mission requirements.
8. **Audit Trail and History.** There should be a standard means for recording the history of operations and observed behaviour to facilitate analysis across the complete system.
9. **Security.** It should be possible to control access to all mission functions, such that operations may be not be carried out without suitable authorisation, and mission behaviour remains confidential without suitable authorisation.

### Space Systems Concept

1. **Conceptual Consistency.** The conceptual view of the system should be applied across all systems, subsystems and functions such that the complete system-of-systems is approached through a single, consistent framework.
2. **Time Management.** The complete system should have a harmonised view of time with facilities for appropriately synchronising time across distributed elements.
3. **Validation and Testing.** The approach to conceptualising and developing systems should be amenable to simplified validation and testing of individual elements, functions, partial systems and complete systems.
4. **Environment/Fault Tolerance.** The consistent conceptual approach should permit the inclusion of redundant functions which may be used to adjust to environmental constraints or system failures with the possibility that the user of the function may not need to be aware of the change. Redundancy and failure management could be either centralised, or decentralised.

### Standards-Based Modularity

1. **Modularity and Reuse.** It should be possible to approach the development of system in functions in such a way as those functions are implemented as distinct modules. This should promote a scalable system which is more amenable to test, and encourage the reuse of functions across missions.
2. **Openness and Marketability.** The architecture should promote use of open, standardised interfaces encouraging common use by many users. One effect of this is to encourage commercial innovation in service providers offering expertise, and additional tools/services resulting in increased competition leading to cost reduction and vendor independence.
3. **Plug-And-Play and Discoverability.** Whilst some aspects of a wider space system may be static over the life of, for example, mission operations, other aspects may need to change and adapt to different mission phases or operational conditions. To facilitate this it should be possible for system to functions to detect and discover other functions, and to establish relationships between those functions dynamically, as required.
4. **Flexibility in Response to Change.** The way that the architecture encourages decomposition of functions should permit the system to adapt to change more readily by limiting the impact of change on other functions.

### Technology Independence and Long-Term Maintainability

1. **Adaptability.** The system should be designed to expect evolution and change throughout its lifetime; it shall be robust and be capable of safely accommodating changes to its environment.
2. **Configuration Handling.** The system should always be in an unambiguously known and observable state which includes all involved elements, both hardware and software. It should be possible to use this state and configuration visibility to track system changes, such as hardware configuration and software updates.
3. **Technology Independence**. The architecture should encourage independence from implementation technology wherever possible, including underlying hardware, software and communications technology. This should make the system more maintainable and contribute to achieving a longer operable life span.

## Technical Vision

The vision for MO Services is to improve operability and address the challenges posed by modern mission through a coordinated approach to the software and communications infrastructure which is being used in both ground and flight systems.

The intention is to specify a software technology which is capable of addressing the full range of user needs, but to do so in a modular way which will permit parts of the technology to be adopted where appropriate. This means that both the technology itself and its standardisation approach must be modular, but with a coherent conceptual framework.

# Mission Operations Concepts

This chapter introduces the key concepts behind Mission Operations and the high-level architecture of MO.

## Services

The examples introduced in the previous chapter have, at their core, a recognition that even diverse missions have broadly similar functions and it is often necessary for those functions to be accessible across physical and organisational boundaries. What matters most is what the function is, such as the provision of telemetry data, and how it should be accessed, not where it is located or how it is implemented.

To achieve this vision, it is possible to view the functionality in the system in terms of *services*. In this context, a service is a set of functions with well-defined semantics which may accessed through a well-defined interface. The functions are made available by a *service provider* and are used by *service consumers*. For example, a service could be provided to distribute updates of key telemetry values. This service could be used across agency or organisational boundaries to provide a standard way of sharing telemetry.

[Insert Diagram Here. Something showing the terms introduced above.]

Using this service-oriented approach, a system can be constructed from multiple, potentially distributed, parts which interact with each other via services. Some will provide services, others will consume, whilst many will do both. Services may be used to facilitate interactions between relatively small software elements, or they may be used between large installations or sites.

## Defining and Standardising Services

To allow a service consumer to interact with a service provider both entities must have the same understanding of the service being used. This requires a common *service model*: a “language” for describing services. The service model allows us to describe what functionality may be accessed using the service, and how it may be accessed. Like many languages, to be able to easily exchange information, the service model also has a written form (a syntax). This permits one developer, or organisation, to define a service which may be used by another.

MO standardises a service model, which is at the heart of Mission Operations. Building on this standardised service model, MO goes on to define a number of standard services, covering the functions which are known to be useful in a wide range of use cases. For each of these services, a set of service functions are defined, using the “language” of the service model. The standard services include (but are not limited to):

* Monitor and Control Services, which cover many of the key functions relating to basic telemetry handling and simple commanding;
* Automation Services, which permit complex operational functions to be constructed from simpler functions;
* Planning Services, which permit future operations to be planned based on desired objectives or goals, and time-based sequences of operations to be executed;
* Data Product Management Services, to permit access to, transfer of and management of data products, such as payload data;
* File and Transfer Management Services, to permit management of, and transfer to/from, remote file stores;
* Time Services, which permit the distribution and synchronisation of time;
* Software Management Services, which permits the maintenance or replacement of software, typically on remote and/or embedded parts of a distributed system;

Each of these *functional services* is standardised in a separate book, and a guide to the full set of MO books can be found in Chapter 9.

[Diagram showing a space system with various services located in various places]

The Monitor and Control (M&C) Services are perhaps the most important of the standard functional services as they cover much of the basic functionality associated with space operations, and for inter-operability between organisations involved in operations. Broadly speaking, the M&C Services cover the following functions:

* access to telemetry, providing the capability to access parameter values on demand, and to be able to set the value of parameters;
* telemetry reporting, where individual or groups of telemetry values are distributed under specified conditions, such as periodically, or in response to change;
* the gathering of statistics on telemetry values, such as minimum and maximum values, averages, standard deviations as so on;
* checking of telemetry, where parameter values are checked against specified conditions, such as being within range, or not changing too quickly;
* the ability to alert any part of the system to a change in conditions, such as a failed check; and
* the ability to define operations which can be invoked (including remotely) and provide a means for invoking those operations.

It is not necessary for a system to implement all of these services. In fact, it is a feature of the MO service model that it is easy to describe how services can be used in part and still remain compatible and/or inter-operable. This is achieved by capturing the *capabilities* of a service that are required by a consumer or made available by a provider. The topic of selecting which parts of a service are needed is discussed in greater detail in section 6.

## Service Interactions and Communications

The MO service model does not does just permit the definition of the functionality associated with a service, but also the way in which that service can be accessed. The MO service model captures the communications involved in a service in terms of *interactions* between the service consumer and the service provider. The way that these interactions take place for a given service function is know as the *interaction pattern* and the MO service model provides a set of standard interaction patterns to choose from.

An interaction pattern specifies the pattern of the “conversation” between the service consumer and provider. For example, the simplest interaction pattern might be the consumer passing some information to the service provider without expecting any kind of response. A more complex interaction pattern may involve multiple responses, with the service provider keeping the consumer informed as to how it is progressing in carrying out the original request and notifying the consumer when it is complete.

MO calls each bit of information sent between the consumer and provider a *message*. An interaction pattern is therefore a pattern of message exchange. It is important to note, though, that although the word “message” is used, this does not imply that the message is implemented as a packet. In fact, one of the benefits of capturing service interactions in terms of these abstract patterns is that it permits service definitions, and the service model they rely on, to be completely independent of implementation and technology. The same service specification can be used to describe the interactions between a service provider and consumer that exist within the same piece of software, on the computer, or between a provider on spacecraft and a consumer on the ground with a separation of thousands of kilometres. In this way, the semantics of the service interactions are separated from the means of achieving the interactions, which is the underlying communications technology.

[Simplified diagram showing a service interaction]

## Technology Independence and Modularity

Whilst the MO service model is completely implementation- and technology-independent, to achieve interoperability a specific set of implementation choices must be made. To facilitate this, MO provides two major types of technology *mapping*:

* Mapping from the service interaction model and data structures to specific message transport technologies such as TCP/IP or CCSDS Space Packets. These allow services to interact across different parts of a distributed system independently of how those parts are implemented.
* Mapping the underlying service interaction infrastructure, the means for accessing or providing services, to various programming languages such as Java or C++.

Chapter 5 discusses developing applications using MO, chapter 6 covers integrating systems using MO, chapter 8 covers mapping to communication technologies, and chapter 9 provides more information on how these mappings are captured as individual standards and books.

In practice, the high-level service interactions described so far are a fairly simplistic view of a system based on services. To meet the needs of operational space systems, service providers and consumers require other features, such as authentication, service location and archival storage and retrieval. MO facilitates these features by specifying a set of common patterns for service specification and a set of common services which leverage these patterns to deliver core aspects of the infrastructure. These *Common Services* are described in more detail in Chapter 4 and are defined by MO individually which permits them to be included, or omitted, from an implementation as required.

## Applying the Service-Oriented Model

To illustrate the service-oriented approach taken by MO, it is worth returning to the four example use cases presented in the previous chapter.

### Example 1: Inter-Agency Hosted Payload

The major challenge in the hosted payload example, is the effective implementation of the inter-agency interface to permit successful data exchange and collaboration. To achieve the mission goals it is necessary to be able to:

* share telemetry values;
* transfer data products;
* and collaborate on plans and schedules.

The use of standard services for those functions provides a clear set of predefined semantics for achieving these goals. Furthermore, by using a well defined technology mapping for communications it is possible to generate documentation at the protocol level, achieving interoperability in a data formats style where the formal service specification defines the required behaviour and the generated protocol documentation defines the data format specification (see section 6.2 for further discussion of this approach).

The modularity of MO means that there are no additional constraints on the implementations at either Agency A or B beyond those needed to facilitate the interaction between services. It is not necessary for either agency to significantly modify their operation software; instead the interfaces may be adapted to expose their functionality as MO services.

[Diagram showing mapping of services onto mission in here]

### Example 2: Radio Astronomy

As with the hosted payload example, the radio astronomy case greatly benefits from the use of standardised services and communications technologies. The second example goes beyond the first in two areas:

* the scale of the distributed system which, to be most effective, needs to encompass and represent all of the different organisations and sites involved;
* the level of integration between the services exposed across the inter-organisational interfaces and the functionality implemented at that site.

For these reasons it may be advantageous to consider MO in the construction of some or all of the systems within the various organisations, rather than simple exposing existing functionality through MO services, as in the hosted payload case however this is not at all required. The use of MO services across the mission encourages a consistent approach to data handling, especially of telemetry, which could make analysis more efficient and simplifying the specific processing necessary to obtain comparable and synthesisable measurements.

[Diagram showing mapping of services onto mission in here]

### Example 3: Lunar Exploration Mission

In the lunar exploration case the advantages of employing an MO approach are most apparent. By treating the complete mission as a single distributed system, employing a consistent set of concepts across the mission, it is possible to employ a consistent approach to operations opening up the possibility of greater efficiency and greater scientific return.

Whilst the physical complexity of mission, with its many distributed assets, is not removed by the use of MO, the use of a uniform service-oriented approach does help to reduce operational complexity. The single Mission Operations Centre can be presented with a consistent view of the complete mission, and every aspect of the mission can be operated consistently regardless of whether it is a rover, an orbiter, or some other asset. This makes it easy to compare and synthesise telemetry even from diverse mission elements, making operations more efficient and potentially safer.

The technology independence built into the MO architecture permits the underlying implementation technology to evolve over the life of the mission. This helps simplify maintenance and assists with combatting issues around technology obsolescence.

[Diagram showing mapping of services onto mission in here]

### Example 4: Small Satellite Earth Observation Constellation

Whereas previous examples have focussed solely on the benefits of MO to operations, the small satellite constellation example also considers the benefits MO offers during development.

The service-oriented approach not only scales well to a large constellation, but it also lends itself to expressing software modularity and encapsulating software implementation behind service interfaces. This encourages the re-use of software elements, permitting software modules to be evolved either without modifying their interfaces, or using the formal descriptions of services captured by MO to help control system change and ease integration.

In simple terms this means that by using well defined services it is possible to present a consistent interface for elements of the constellation regardless of the software or hardware behind that interface. So as the constellation evolves and ages, as old assets are decommissioned and replaced by newer assets, the interface presented to operations remains the same.

[Diagram showing mapping of services onto mission in here]

# Mission Operations Technology

In this chapter we take the conceptual description from the previous chapters and show how this is all actually represented in MO.

## The MO Service Model

The MO service model is closely aligned to the model outlined in the previous section:

[MO service model diagram (effectively figure 2-3 from RM)]

Fundamentally the MO model is based on the simple model previously shown, however the MO model expands on it to provide support for the more complex, real world, interactions that services may be required to make.

Not all operations are created equal, many different interaction patterns may be needed, so to this end the MO service model supports six different patterns (more in section 4.2.2). These different patterns allow the service specification to fully express the richness required by the service designer.

It is also recognised it often makes sense to group operations together, where several operations are required to perform more complex activities. These groups in MO are called capability sets and they allow both service consumers and providers to work with only the parts of a service which are necessary for their application.

## MO Layering

The core behaviour of MO, that of allowing service to interact, is provided by the *MO framework*. Conceptually, the MO framework breaks down into three layers, as show in Figure X. It is important to note that these layers are conceptual only; they do not need to have a one to one corresponding layer in any software: it is perfectly reasonable to merge one or more layers together in software for optimisation reasons (or any reasons you may have). It is also perfectly reasonable to keep the layers separate in software, to utilise the flexibility that gives.

[MO layer diagram, simple one without Common or COM]

From the top to the bottom the layers are:

* Service Adaption layer
  + Maps between the high-level service interface to the lower level MAL interface.
* Message Abstraction Layer
  + Provides checking of message pattern state model, authentication/authorisation validation, and mapping between its generic message interface and the interface provided by the layer below.
* Message Transport Layer
  + Provides the message transportation function to the layer above.

Each of these layers is expanded upon in the following sections. In Section 4.5 the expanded layers are shown in full and a mapping from those layers to the original requirements from Section 2.2.

### Service Adaption Layer

The service adaption layer is responsible for providing to applications the actual service interface to the service consumers and service providers. Its job is to provide the mapping from the lower level MAL interface to the specific service level interface:

[diagram showing the more details view of this layer]

When the service specification is mapped to a programming language this layer is represented by the high level, service specific, API that is most likely used directly by the service consumer and providers.

### Message Abstract Layer

The Message Abstraction Layer (MAL) is conceptually responsible for three main things:

* Ensuring the operations adhere to the state charts for the chosen interaction pattern
* Applying any authentication/authorisation checks
* Moving the messages from the service adaption layer to the message transport layer.

[diagram showing the more details view of this layer]

The MAL specification defines six different interaction patterns:

* SEND
* SUBMIT
* REQUEST
* INVOKE
* PROGRESS
* PUBSUB

The specifics of the patterns will not be covered here, see RX for more details, but each pattern has a well-defined state chart for the interaction. It is role of the MAL to ensure that the state model of a specific interaction is enforced, so for example that an error is not returned when one is not permitted.

Authentication and authorisation support is integrated directly into the MAL concept. The specification defines an “Access Control” component that all messages received and sent are required to be filtered through. This component is where any authentication and authorisation policy may be applied; the MAL itself does not do anything, it delegates to the (optional) Access Control component.

Once all these checks have been passed the message is then passed to the corresponding layer, specifically the Service Adaption layer if it is an incoming message, and the Message Transport Layer if it is an outgoing message.

### Message Transport Layer

The message transport layer is responsible for providing the mechanism that moves the messages that form the interaction pattern between the service consumer and service provider:

[diagram showing the more details view of this layer]

This can be implemented in many different technologies, from network technologies such as binary encoded packets over TCP/IP, to space links to in process function calls. It is the job of the message transport layer to hide this and adapt from/to this and provide a standard interface to the layer above (the MAL).

It is also possible that the message transport layer may be involved in the authentication and authorisation process, for example if message signing is used, this cannot be done until the message is encoded.

## The Common Object Model (COM)

The COM provides a standard object model for MO Services to utilise. An object is defined is a thing which is recognised as being capable of an independent existence and which can be uniquely identified. An object may be a physical object such as a spacecraft or a ground station, an event such as an eclipse, or a concept such as telemetry parameter. It forms the fundamental part of a service specification, e.g., a parameter definition, a parameter value at a given point in time, a command.

An object is fundamentally different to something like a data structure, a data structure is a holder of information or data values whereas an object represents something. For example, a data structure holding the definition of a parameter may contain fields such as description and parameter unit but it does not *mean* anything, whereas a Parameter Definition object may contain the same information but it comes with a defined meaning (it is an attribute of something being monitored), and in the context of a service, a specification of what can be done with that objects (get, observer, possibly setting its value).

The COM specification is split into two parts, the first specifies a standard data object model, and the second specifies some support services and service patterns.

### Standard Object Model

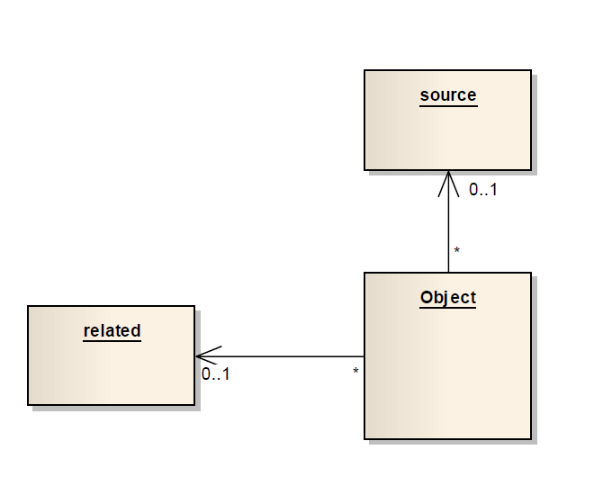
Whereas the MAL provides the building blocks that can be used to define the operations of a MO service, the COM provides the building blocks for the specification of the data objects of a service. This builds upon the MAL to define a standard data model for an MO service.

The object model is based on the principle of RESTful architectures; namely, each object can be identified by a unique identifier. There are no requirements on what an object may be except that it must be possible to uniquely identify an instance of it so that it can be referenced.

Each service that utilises the COM must define the object, or set of objects, that form the data model of the service and how each object is managed (created/updated/deleted), archived, and interacted with using the operations of that service.

### Object Structure

Whilst the COM does not limit what may be considered an object by a service specification, it does define a basic structure for the objects:

 COM Structure

<text explaining the links, that they are other COM objects, and why we might need them (give example (Parameter value linking to its parameter definition)>

### COM Service and service patterns

The COM specification also includes a single support service and two service patterns that build upon the basic object model; these are:

#### Archive service

The archive service provides a generic means for persisting objects. It follows the basic Create/Retrieve/Update/Delete (CRUD) principles and therefore fits with most archiving systems. It provides a simple basic set of operations and a basic requirement on the information required to persist service objects in it. A further discussion of the use of the COM archive is given in section 6.3.1.1.

#### Event service pattern

An event is a specific object representing ‘something that happens in the system at a given point in time’. The event service pattern defines a common mechanism for the distribution of events and also defines how a service that creates events should interact with the archive service. See section 6.3.1.2 for more information.

#### Activity tracking service pattern:

The activity tracking service pattern provides a means for tracking the progress of activities; an activity is anything that has a measurable period of time (a command, a remote procedure, a schedule, etc.). It defines an event pattern that supports the tracking of activities from the initial consumer request, tracking its progress across a transport link, to reception by the provider and execution in that provider. See section 6.3.1.3 for more information.

## The Common Services

The Common services provide infrastructure services support the mission operations services, namely:

* Directory
  + For publishing and locating providers of specific services
* Login
  + For allowing operators to log into systems
* Configuration
  + For the storage and sharing of service provider configuration and also the activation of those configurations in providers.

### Directory Service

The directory service provides publish and lookup facilities to service providers and consumers. It allows providers to publish their location in the form of a URI (Universal Resource Indicator) so that consumers can locate it without having to know in advance the location. 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.



Directory Service Concept

The directory service provides a similar function to what DNS provides to the Internet, it maps from provides to the network information required to locate and communicate with them. A service provider publishes to the directory service the set of services, and the capabilities of those services, that it supports combined with the URI information required to locate and connect to it. Service consumer can then query the Directory service to locate the providers of the services that they require.

### Login Service

The Login service defines a standard interface for MO applications to provide security credentials to the security mechanism used by a system. It also allows the MAL to use that information in its Access Control component (see section XX) for authentication and authorisation purposes allowing these two functions to be implemented at the infrastructure level.

The Login service and the access control provided by the MAL are fully dependent on a deployment-specific security architecture (for example the authentication protocol Kerberos). Both layers (Common and MAL) provide access to, and use of, this security service. Neither implement it themselves.

See XXX for more information regarding access control.

### Configuration Service

The Configuration service defines how to:

* access the configuration of a service provider,
* list available configurations,
* and request activation of those configurations in the service providers.

This allows a service consumer to obtain service specific configuration from a service provider without prior knowledge of that provider.

For example, using the Configuration service a provider of the Parameter service can allow consumers access to the set of Parameter definitions that it supports, listing of possible configurations that it supports, and allow switching of the configuration of itself (switching between versions of parameter definitions).

## Full MO layering

Below is the fully expanded layering of the MO framework:

[diagram showing the expanded layers from the above sections including the COM and Common specifications]

<Some text about this>

## Applying the MO framework

Let’s map this back to the User Needs list in section XXX.

### Inter-Operability and Intra-Operability

|  |  |  |
| --- | --- | --- |
| User Need | Description | Comment |
| UN-10: | **Interoperability.** It should be possible to interact with mission functions through a well-defined interface from “outside” of the system. This permits external interfaces to existing and separate systems. | The service approach ensures all aspects of the interface is well-defined, so behaviour is explicitly defined (as opposed to implicitly with simple data formats) which ensures functional interoperability.  The use of standardised mappings to encodings ensure on-the-wire interoperability. |
| UN-20: | **Location Independence.** The conceptual way that mission functions are used should be independent of where those functions are located, whether in space or on the ground. Although there will obviously be impacts from the communications capability with remote locations, such as bandwidth and availability, the basic operational concept should be location-independent. |  |
| UN-30: | **Separation of Functionality from Infrastructure.** Functions which achieve mission-level goals should be kept separate from the infrastructure that allows those functions to be used. This means that, for example, communications capabilities should be treated as infrastructure. | The separation of functional logic in the Service Adaption Layer and the communications logic in the Message Transport Layer ensures location independence. |
| UN-40: | **Operability.** As an operator, whether human or machine, it should be possible to access all mission functions and to control their behaviour. |  |
| UN-50: | **Observability and Traceability**. As an operator, whether human or machine, it should be possible to observe the behaviour of all mission functions and to track/record the progress of operations across both local and distributed systems. |  |
| UN-60: | **Data Intensive Operations**. It should be possible to carry out operations which involve the generation and management of large and small data products. |  |
| UN-70: | **Autonomy**. It should be possible to automate the operation of system functions such that they may be carried out without human interaction. The level of autonomy necessary will be dependent on mission requirements. |  |
| UN-80: | **Audit Trail and History.** There should be a standard means for recording the history of operations and observed behaviour to facilitate analysis across the complete system. |  |
| UN-90: | **Security.** It should be possible to control access to all mission functions, such that operations may be not be carried out without suitable authorisation, and mission behaviour remains confidential without suitable authorisation. | Fundamental to the MAL concept is the use of Access Control to provide authentication and authorisation functionality centrally. |

### Space Systems Concept

|  |  |  |
| --- | --- | --- |
| User Need | Description | Comment |
| UN-100: | **Conceptual Consistency.** The conceptual view of the system should be applied across all systems, subsystems and functions such that the complete system-of-systems is approached through a single, consistent framework. |  |
| UN-110: | **Time Management.** The complete system should have a harmonised view of time with facilities for appropriately synchronising time across distributed elements. |  |
| UN-120: | **Validation and Testing.** The approach to conceptualising and developing systems should be amenable to simplified validation and testing of individual elements, functions, partial systems and complete systems. |  |
| UN-130: | **Environment/Fault Tolerance.** The consistent conceptual approach should permit the inclusion of redundant functions which may be used to adjust to environmental constraints or system failures with the possibility that the user of the function may not need to be aware of the change. Redundancy and failure management could be either centralised, or decentralised. |  |

### Standards-Based Modularity

|  |  |  |
| --- | --- | --- |
| User Need | Description | Comment |
| UN-140: | **Modularity and Reuse.** It should be possible to approach the development of system in functions in such a way as those functions are implemented as distinct modules. This should promote a scalable system which is more amenable to test, and encourage the reuse of functions across missions. |  |
| UN-150: | **Openness and Marketability.** The architecture should promote use of open, standardised interfaces encouraging common use by many users. One effect of this is to encourage commercial innovation in service providers offering expertise, and additional tools/services resulting in increased competition leading to cost reduction and vendor independence. |  |
| UN-160: | **Plug-And-Play and Discoverability.** Whilst some aspects of a wider space system may be static over the life of, for example, mission operations, other aspects may need to change and adapt to different mission phases or operational conditions. To facilitate this it should be possible for system to functions to detect and discover other functions, and to establish relationships between those functions dynamically, as required. |  |
| UN-170: | **Flexibility in Response to Change.** The way that the architecture encourages decomposition of functions should permit the system to adapt to change more readily by limiting the impact of change on other functions. |  |

### Technology Independence and Long-Term Maintainability

|  |  |  |
| --- | --- | --- |
| User Need | Description | Comment |
| UN-180: | **Adaptability.** The system should be designed to expect evolution and change throughout its lifetime; it shall be robust and be capable of safely accommodating changes to its environment. |  |
| UN-190: | **Configuration Handling.** The system should always be in an unambiguously known and observable state which includes all involved elements, both hardware and software. It should be possible to use this state and configuration visibility to track system changes, such as hardware configuration and software updates. | The Common Configuration service provides a standard mechanism to the observation of and activation of different system configurations. |
| UN-200: | **Technology Independence**. The architecture should encourage independence from implementation technology wherever possible, including underlying hardware, software and communications technology. This should make the system more maintainable and contribute to achieving a longer operable life span. | The use of the MAL to separate the service definition from the technology used to implement them allows the technologies to change over time. |

# Developing MO Applications

This chapter covers in more detail the MO concepts and technologies but from the specific point of view of an application developer. Firstly, we look at what an MO application is, examine how applications relate to the rest of the system, including standard services, to achieve their objectives. Finally, we look at developing applications using the current tooling technology.

## A typical MO application

[Text required to describe what is an MO application]

A typical MO application may be either a consumer of MO services, a provider of MO service, or both a consumer and a provider:

[diagram]

As was shown in figure [XXX layering diagram] the application itself is most likely built upon layers such as an MO Service Adaption Layer and a MAL. It will use a message transport to communicate with any providers and consumers as required:

[Component diagram of an MO application interacting with peers]

For example, consider the case of a consumer that wishes to receive live updates for a set of parameters. The set of parameters is prearranged and the parameter service provider is using a fixed and known address:

[simple diagram showing basic setup]

Most likely, in non-trivial deployments, infrastructures services such as Directory (for address lookup) and Login (for authentication) will be used. There may also be the possibility to interact with a COM Archive service provider (for historic values) and possibly a Configuration service provider (for more flexible parameter definition lookup):

[expanded diagram with infrastructure services]

These are relatively simple examples of consumer side only, next we look at a more complex example of a complex service; the application is both providing a service and consuming several services. Consider the case of a Planning Request service provider that is interacting with planning clients, allowing them to submit their plan requests whilst it interacts with an automation service provider to facilitate the calculated plan. It may be receiving live parameter values from a parameter service provider, but it more likely would want to use a COM Archive service provider to obtain historical parameter values:

[initial deployment diagram]

In addition to this, it is publishing its address information in a Directory service and using a Login service to provide authentication of its consumers. In this deployment it is co-located with the automation service and the system architects have decided that there is no need for authentication between it and the automation service provider:

[infrastructure diagram showing externally accessible directory and login service providers]

Finally, the automation and planning request services may be defined to use the COM Activity Tracking event pattern to report their execution progress, and so the planning service provider may use these to monitor the progress of the activities it has started in the automation service provider:

[activity tracking]

In each of these cases, the interface from the application layer to the service adaption layer is the same, no awareness of the message transport layer is required as that is hidden by the adaption layer and the MAL. Depending on the deployment in question you may need to interact with one or more support services providers (Directory, Login, etc) but in simple cases you may only need to interact with one or even zero of these.

In the next chapter we will look in more detail at the interface between the application itself and the service adaption layer.

## Relationship between applications and services

As shown in the previous chapter, the functional logic of an application lives above the Service Adaption Layer, so most applications do not need to directly interact with the MAL or the relevant message transport layer.

The Service Adaption Layer is the high-level service-specific API that is presented to an application that adapts to/from the more generic MAL API. It hides from the application layer the more generic nature of the MAL and provides an API that uses the operation names and structures specific to the relevant service specification.

In reality, this is represented in a programming language API and that API will be specific to each programming language, so it will be a different API in Java to that used by a mapping to C++. They may be similar but obviously they are not transferable; you cannot directly use a C++ API with a Java application. The API is also specific to the relevant service, so the API in Java for a Parameter service will be different to the API in Java for the Directory service.

Rather than publish standard APIs for each service in each supported language the MO approach is to define a mapping from the MO service model to the relevant programming language. So for example, R[X] defines a mapping from MO to Java whereas R[Y] defines a mapping from MO to C++.

The MAL specification also defines an XML notation for providing a machine readable version of the service specifications (the official service specification documents are auto-generated from this). As the mappings are well defined and encodable it is possible to implement code generators that take the service specification XML and auto generate the Service Adaption Layer for the relevant language, in fact these already exist as open source (see section XXX).

New language can be supported by defining a mapping from MO to that language, implementing that mapping into a code generator, and providing an implementation of the MAL and the chosen message transport layer. See section 8 for a further discussion on this aspect.

The use of generated APIs, that are defined by standards both in terms of programming language representation (language mapping), but also more importantly in terms of the service contract and therefore required behaviour of both consumer and provider (service specification) allows the specifics of the one side to be independent of the other. So the programming language of the service provider is completely disconnected from that of a service consumer, and the hardware choice of one is completely independent of the other. This separate of technology is vital when considering large space systems such as constellations, where many assets are being operated over a period of time where the hardware platform is likely to evolve. Having a standard specification for something like a parameter service allows applications to remain independent of the hardware and application being monitored, so whether that is the first generation of asset in the constellation or the latest the consumer applications are not affected:

[constellation deployment where the service specification is the constant in the system]

## Standard vs application-specific services

The set of services defined as part of the MO standards suite covers a broad range of scenarios, however it may not be possible for all aspects of monitoring and controlling a space mission to be covered. For the vast majority of cases these will be supported by use of the M&C Action service, for example defining an Action to switch on the power to a subsystem. However, there will always be situations when a new service may make more sense rather than the use of the Action service.

Using the Action service provides a simple mechanism for issuing requests to the remote system to perform some activity, for example switching something on or triggering a more complex operation such as spacecraft mode change. The COM Activity Tracking service allows the monitoring of the execution of that Action and the reporting of failures.

However, in some cases it may make more sense to define a new service when a more generic function that may be used in several places or between several missions, for example a common function like file transfer management (covered by File services, see section XXX). In this situation, where a common function is not covered by the standard MO services, a user defined service can benefit from all the layering and separation that the MO framework provides whilst staying closely focussed on the specific function it is covering.

An example of this would be when the set of operations required for a specific activity or function may require a specific pattern, for example where one operation returns an identifier that is later used in other operations. This kind of interaction is not possible using the M&C Action service whereas is straight forward when defining a service.

[more words here to finish this topic off]

## MO application development tools

Getting started using the MO framework and suite of standards may seem a little daunting; it is certainly the situation where a little help at the beginning can go a long way. To this end the member agencies of the working group have worked to make available the reference implementations, the code generators, and any standard libraries produced available as open source software.

In this section we briefly cover the software that is available to help with MO application development.

## MO service XML editor

When developing new service specifications, writing the service specification directly into XML is not an easy task. To assist with this task ESA has developed an open source editor, as an Eclipse plugin, that provides a visual editor for the development of service specifications:

[screen shot]

The editor allows the service designed to visually create the service specification using its GUI rather than having to hand edit the MO XML. It also wraps the ESA API and documentation generator to allow you to easily export the XML into any supported language without having to leave the editor.

## API and documentation code generator

ESA has produced and open sourced its code generator that reads in a supplied MO XML service specification and generates standard Java APIs and (if desired) MS Word based documentation. It is this generator that is used to assist in the development of the official CCSDS MO service standards.

This small application uses the idea of generator plugins to provide the mapping and so is easily extended to other languages. CNES has produced a plugin for it that is able to generate to a bespoke C interface.

As the range of languages is increased so will the set of available code generators.

## Open source Java implementation

CCSDS procedures require any new Blue book standard, so this applies to the MAL and all message transport mappings, to have two independently implemented implementations. For the MAL specification this was performed by ESA and CNES, both of whom have made available their implementations as open source (link to location?).

ESA is also committed to make available any reference implementation of the message transport layer mappings and message encodings to fully support development of applications using MO.

## Standard Java libraries in Maven Central

As part of ESA’s support to the MO standardisation work, whenever a new service specification is standardised ESA uses its Java code generator to auto generate the relevant service API, compiles it and makes it available for download as a precompiled Java JAR file with associated source code and Java documentation. This facility is hosted in something called Maven Central (URL to searchmaven).

ESA’s Java reference implementation of the MAL and the standard message transport layer adapters are also available in Maven Central.

Most modern Java build systems (Ant, Maven, Gradle, plus many others) can directly download these JARs from Maven Central as part of the build process, therefore incorporating these components into your application is trivial.

## Example application source

[Can we provide links to GitHub locations of example applications?]

# Integrating MO Systems

Integration of MO systems is concerned not only with integrating MO compliant systems with each other but also the integration of MO systems with non-MO systems such as existing infrastructure or systems where MO is only used for interoperability needs. This also includes deployments where MO is not used by any system directly but as an interoperability conduit between cooperating organisations:

[diagram showing in brief the types of integration mentioned above]

Each of the scenarios shown above are examined and discussed in the following sections.

## Integrating MO systems

There are three aspects to consider when integrating MO compliant systems, firstly the chosen communication protocol, and secondly the information being shared between the cooperating organisations, and finally how is that access provided.

The first aspect, communications protocol, is by far the simplest. Any decision made can easily be changed later, or multiple protocols supported in parallel, due to MO’s separation of functional layers from communication layers. There are still many aspects of the messaging technology to consider, such as security, performance, etc but these can be reviewed in isolation. See chapter 8 for further discussion of this topic.

The second aspect to consider is what to integrate and how far that integration should go. For example, providing Parameter information to another organisation is possible using the M&C Parameter service (see section XXX), but what capabilities of that service should be exposed, will there be a need for historic parameter value access, is this done via the COM Archive service, is access to the Common Configuration service required? A more detailed discussion of this topic is given in section XXX.

The final aspect is how access is provided. It is entirely possible to give direct access to the service providers in question, and in many situations that is perfectly valid. For example, when integrating system, both residing inside the same organisation, or where they originate from different organisations but are being integrated to form a large system for a single purpose:

[diagram showing direct integration]

However, in many situations that level of integration is either not possible or not desirable. For example, where one organisation is providing information to another, it may be critical that there is a separation of systems from the internal service providers to external organisations for reasons of security, reliability, or performance. In this scenario one possible solution is provide a small façade system between the internal service providers and the external organisation:

[diagram showing façade deployment, including separating into separate security domains]

The façade may map directly onto the internal service provider, but it also may also be a mirror only being updated with select information at a slower rate. This would allow the façade to be compromised without any impact on the more critical internal service providers. It is also possible, and probably quite likely, that the façade provides a protocol bridge capability isolating both internal and external protocol choices from either side.

The integration approach taken will depend on a wide range of factors which are likely to vary significantly between uses. The MO architecture is explicitly designed to permit this flexibility, with integration decisions isolated from applications through the MO framework.

## MO at the edge

When integrating non-MO systems with MO systems, or when MO is being used as an interoperable medium for interchange between non-MO systems, there are two main approaches that can be used: bridging applications or message translators.

### Bridge approach

The bridge application approach is similar to the façade deployment shown in section 6.1: an application bridges (or adapts) from one system to the other. Each side sees what they expect to see from a system compliant to their interfaces, and the façade is also able to provide separation from the technologies and security concerns of each system as outlined above:

[diagram showing bridge application approach, show both client and server being bridged]

This approach is no different to when integrating any two systems of different technologies, the advantage of bridging to MO is that you only need to do this once to support any MO compliant system.

### Message translation

Discuss how a message-based approach is possible, by auto-genning to formats, and translating on the fly. Outline how this approach is identical to tradition ICD based interfaces.

## Integrating services

Most application services defined in MO utilise some aspect of the COM services and service patterns such as Event and Activity Tracking. The Common services also provide some infrastructure services that are designed to be used in these scenarios.

The following sections provide a brief discussion of each of these services and the aspects to consider when integrating service providers and consumers.

### COM Services

#### Archive

The use of the Archive service is required by many service specifications for the storage of historic information as well as the configuration of service providers. Providing access to some aspect of an Archive service may be fundamental to the success of the integration but, depending on the actual use, it may not be required at all. For example, if one system is providing only live Parameter values to an external system, then there is no need to have external access to an Archive service provider.

If historical access is required then some kind of Archive service access is most likely required. In this situation it needs to be carefully considered whether direct access to an internal archive is appropriate or whether some kind of federated archive capability is more appropriate:

[diagram showing federated archive deployment]

#### Event

The COM event service pattern is used by many services to distribute asynchronous change information. It is also fundamental to services such as COM Activity Tracking.

It is possible no events need to be distributed when certain services are used in specific ways, for example, use of the Parameter service is a request/response pattern where the service consumer polls the service provider for current values only when required.

However, if event distribution is required, the simplest approach is to forward the events onto the external system. It may be sensible (or even required) to filter events to avoid overloading external systems or passing on sensitive information.

[diagram showing event relay/filtering]

#### Activity tracking

Activity tracking is an event-based pattern that is used to report transfer of a request from a consumer to a provider and then execution of that request in the provider. The main use case for activity tracking is in the situation where the request is handled by a provider external to the originating system (local requests are expected to be reliable enough to not need activity tracking although it can be there).

Activity tracking uses the Event service and therefore it is a requirement for it to be present if activity tracking is required.

[diagram showing activity tracking running over event service]

### Common services

#### Directory

The Common Directory service is a key service for supporting system integration. Whilst it is possible to use fixed connection details for services, or share these using other mechanisms, this is a cumbersome approach that limits flexibility of deployment. By using a Directory service, at a fixed and well-known address, to hold provider capability and connection addressing details external service consumers can quickly and reliability locate the service provider they require and also be flexible to change in the system.

[diagram showing service address lookup]

The Directory service is a key service that integrators should strongly consider supporting.

#### Login

The Login service provides facilities for authenticating consumers when you wish to use the MO access control concept to provide authentication and authorisation capabilities. It is possible to provide this functionality by other means, for example by using fixed and prearranged communication links for each external system and relying on other technologies such as secure sockets to provide communication encryption. However, when the system in question is expected to support multiple external consumers or a Directory service is being used to support dynamic service provider addressing, a login service can simplify the task of authenticating the external systems and for providing authorisation information.

In this use case there is a link between the login service exposed to the external system and the mechanism used to bridge to the external system. If it is MO to MO then the login service provider needs to connected to the access control component of either the service provider or the façade application in between. For the bridging of non-MO to MO systems then the provider must link the provided login service to the mechanism used to provide authentication tokens to external systems and used for authorisation.

[diagram showing basic login and access control relationship]

#### Configuration

The Configuration service is a key enabler for distribution of service configuration information. In the integration scenario this means allowing consumers to access the list of configured information for each service they wish to interact with, for example in the Parameter service this is the list of Parameters available and their definitions.

Services often provide operations to retrieve the service configuration information directly, but these operations are designed for specific service object lookup and do not usually support retrieval of the complete configuration of a service provider, the Configuration service is designed specifically for that purpose allowing service consumers to obtain the complete configuration of a service provider with minimal overhead.

In combination with the Login service (and Directory service for address lookup) it is possible to create a deployment that allows external systems to quickly retrieve the configured service information available to them. The use of the Login service allows the Configuration service provider to filter the returned information, only returning the information the relevant consumer can access:

[diagram showing configuration service working in conjunction with a login service to provide a filtered list of config data]

## Integrating application services

Integrating two systems using MO is no different than existing approaches, the big difference being that in most cases the actual interfaces themselves are already defined in CCSDS specifications.

The basic procedure is:

* Review the list of required capabilities
* Map requirements to MO services
* Review each service to see which capabilities are required
* Review required capabilities to determine which other support (COM/Common) services are required

At this point you will have a list of services, the selected capabilities inside those services, and the support services required. If you are providing the services, or a subset of them, a decision needs to be made on how to support it, either directly or via a façade. This can be different for each service.

At this point you made be able to decide on how the configuration of each of the services is to be shared, in some cases where the configuration is small it can via traditional ICDs, however if it is more complex or of a more dynamic nature it may make more sense to consider using the Configuration service.

If a traditional approach of bespoke, non-standard, interfaces had been used then it would have required the development and specification of these. The use of standard CCSDS specifications removes the need for this expensive, difficult, and error prone task.

Finally, it is then a matter to implement and test as normal. This is, of course, only required for the first use of the interface, future integrations can reuse the existing work and much reduce the effort required, possibly down to zero.

# Managing and Administering MO Systems

In this chapter we review how the concepts from chapter 3 impact the responsibilities of managers, administrators and, to a certain extent, operators. There is focus on use of, rather than on the engineering of, the MO standards. We will look at the impact of some of the support services, such as COM activity tracking and archive service, but also where security aspects of MO are specifically addressed, as this is another topic of frequent concern amongst those new to the technology.

Specifically, these areas are:

* Security and user management
* Audit trails
* Archiving and backup
* Technology migration

In the following sections, for each one above, there is a discussion how it is affected by MO and what MO does to help with the issue.

## Security and user management

[not sure what to write here, any ideas would be good!]

## Auditing and audit trails

[not sure what to write here, any ideas would be good!]

## Archiving and backup

The central service to consider for archiving and backup is the COM archive service. If the COM archive service is not used then the relevant functionality behind your service providers should be considered, however this is outside of the scope of this document.

The advantage of the COM archive service is that the interaction with it and the functional services is fully defined in each service specification. Depending on your usage of the services you use you may not actually require an archive service, however if you are using one then you have a single capability to consider.

The COM archive service does not require any specific archive implementation or a single access point, so how it is implemented in your deployment is a deployment decision.

In terms of things to consider, normal backup policies should be followed, the only consideration above that is to ensure that backup occurs when the archive is in a consistent state however this is no different for most archives. For example, ensure that the backup does not occur in the middle on an update operation.

## Technology longevity and migration

One of the biggest issues with the long timescales often encountered with space missions is technology longevity and the resultant need to migrate to new technologies both in software and hardware.

Dependencies on out of date operating systems, technologies and other software is a large concern for space missions bring with it a large overhead in terms of cost and maintenance but also security when needing to interact with other organisations.

For hardware, being unable to move to more modern and faster hardware without reengineering systems is also a large concern as older hardware becomes more expensive and harder to keep running.

One of the key design goals of MO is to separate out the many aspects of a system, to separate the functional aspects from the language used to implement it and to separate the technologies used to communicate from those functional aspects:

[brief, simple, stack diagram showing separation of implementation from technologies but also separation of implementation between consumer and provider]

MO’s ability to separate these different aspects means that elements of a system are cleanly separated from each other:

* Application migration

Upgrading applications in the system can be performed without affecting other existing applications allowing easy update of components of a system.

* Hardware migration

Separation from hardware supports migration to newer hardware.

* Technology migration

Independence from communications technologies allows these to be changed without affecting functional components but also being able bridge between technologies as well as support multiple ones concurrently allows easy migration to newer, possibly more secure, technologies.

Finally, it should be noted that due to MOs bridging ability technology migration can be performed in isolated areas without requiring large replacement if this is not desired.

# MO Communications

Chapter length estimate: 4-6 pages. Although it seems somewhat specific, the topic of communications seems to be of particular interest to some and this interest does not always align with other stakeholder interests. It is for this reason that it is proposed to split this out as a dedicated chapter. The intention here is to focus specifically on the way the various layers of the architecture interact to allow communications in a distributed system. The main topics would include interactions, with concrete protocol/binding examples, and illustrative examples which relate communications to physical space and ground segment topologies. It is also likely that some aspects of distributed systems are covered, such as service discovery, which effectively allows the dynamic binding of distributed nodes.

Chapter summary:

* Detailed look at the architecture and the way communications works
* Overview of different communications bindings
* Distribution and service discovery
* Practical example illustrated with a physical space-ground topology

Detailed comms architecture

Focus on the layer diagram showing the interaction between Service Adaption Layer, MAL and Message Transport Layer.

[drill down diagram]

Illustrate the use of standard APIs allows the clean separation of these layers

Discuss how Message transport layer can be swapped out without affecting any layer above and also how this allows you to support heterogenous network and HW deployments.

Show this in the context of when using MO only for interoperability

Explain how this can also be used for not only IPC but also in-process communications

Explain about merging layers is possible to increase performance, remove overhead, but at the cost of flexibility.

Overview of different comms bindings

Give outline of currently available bindings and their expected usage

Distribution and service discovery

Discuss how distribution, especially when used dynamically, really benefits from a Directory service.

Illustrate the importance of this for interoperable situations. Mention how this is very standard practice, considering web addresses and DNS.

Discuss how Directory service also being an MO service has benefits (technology isolation, and multiple technology support).

# Overview of the Standards (2-3 pages)

This chapter outlines the general approach taken to further MO documents and aims to provide the reader with a guide to the rest of the standards, allowing them to decide which to read next.

The MO suite of standards is generally split into three areas, firstly MO architecture books, secondly technology mapping specifications, and finally the service specifications:

[top level document organisation diagram]

Some documents span more than one area, for example the COM specification defines some aspects of the MO architecture whilst also specifying services such as COM Archive.

[Mapping of the standards onto topics and stakeholders]

## Architecture documents

The architecture documents form the foundation of the MO service approach. They provide the structure for applications and the foundation and notation used to specify services and technology mappings.

### Mission Operations Reference Model

This Recommended Practice defines a reference model for the MO standards and applications using them. It expands on chapters XYZ of this document covering these aspects in greater depth.

See R[x] for further reading.

### Mission Operations Message Abstraction Layer (MAL)

The MAL Recommended Standard defines the MAL in conformance with the reference model, formally defining required behaviour of both consumers and providers when using the MAL. It also defines the basic data model used by the MAL and services specified using the MAL.

See R[x] for further reading.

### Mission Operations Common Object Model (COM)

The COM Recommended Standard defines the standard object model for MO Services to utilise. The specification is split into two parts, the first specifies a standard data object model, and the second specifies some support services and service patterns.

See R[x] for further reading.

## Technology mappings

MO technology mappings come in two forms, either a mapping to a programming language or a mapping to a message transport technology.

For a mapping to a programming language, the specification will define how to transform from a service specification to a high-level API in that language, how the MAL is represented, and possibly how message transports should be represented. It should also define how the data types of the Mal are mapped to the language.

For a message transport technology mapping, the specification will define how the interaction patterns of the MAL are mapped to the transport technology, any limitations caused by the mapping to the technology (for example certain interaction patterns may not be supported), and how the MAL data types, must be encoded. The technology mapping specifications, when they define an encoding, usually do it in such a way that it can be used by other message transport mappings. For example, the ZMTP transport mapping does not define an encoding but can be used by one from another specification such as the TCP/IP specification.

Each technology mapping has an expected usage, this does preclude its used outside of that, but the design of each one has been with a specific domain in mind:

[UML deployment:

diagram showing deployment and where the various technology mappings are expected to be used

Space packet on spacelink and beyond

TCP/IP in local fast connections

ZMTP inside MOC

HTTP outwards]

Each of the current technology mappings are outlined in the following sections, more may be available and have been defined after publication. Other mappings may also be available by organisations that have been defined outside of CCSDS.

### Java API

The Java API recommend practice provides a mapping from MO XML, which holds a service specification, to the Java programming language. It also defines a standard API to the MAL, to the MAL access control component, and finally a standard message transport layer API:

[UML diagram showing MAL API standard interfaces]

These standard APIs allow, for example, different implementations of the message transport layer to be changed without requiring any changes to the layers above. The Java API also supports multiple message transports to be accessed concurrently in the same application, assuming the MAL layer above has been implemented with that support.

The ESA code generator generates Java service API code compliant to this specification (see section XXX).

See R[x] for further reading.

### C++ API

The C++ API recommend practice provides a mapping from MO XML, which holds a service specification, to the C++ 11 programming language (or C++98 with external libraries). Similar to the Java API, it also defines a standard API to the MAL, to the MAL access control component, and finally a standard message transport layer API.

See R[x] for further reading.

### Space Packet Transport Binding and Binary Encoding

This Recommended Standard defines the binding between the MAL and the Space Packet Protocol specified in R[x]. This binding allows MO Services to use the Space Packet Protocol as a messaging technology and therefore be transported over anything that can transport/route Space Packets. It also defines two binary encodings, one fixed and one variable length.

The Space Packet mapping is intended for situations where MAL messages are crossing into the space segment.

See R[x] for further reading.

### Binding to TCP/IP Transport and Split Binary Encoding

This Recommended Standard specifies a binding of the MAL to TCP/IP and another binary encoding for MAL data types called split binary. Split binary encoding optimises the encoding of MAL messages by splitting out Boolean fields into a separate bit field therefore reducing the overhead of single bit fields without the difficultly of bit packing.

The TCP/IP mapping provides a very efficient and fast transport but only supports point to point connections so is best suited to specific consumer/provider pairs that require high bandwidth.

See R[x] for further reading.

### Binding to ZMTP Transport

This Recommended Standard defines the binding between the MAL and the ZeroMQ Message Transport Protocol (ZMTP). This binding allows MO Services to use ZMTP as messaging technology in all situations where this may be required. The encoding of the message body itself is not specified in this Recommended Standard but any MAL encoding, specified in other books, can be used for encoding the body of the messages.

The ZMTP transport binding is intended in situations where many applications may want to participate, where transport level message routing and broadcasting is desirable.

See R[x] for further reading.

### Binding to HTTP Transport and XML Encoding

This Recommended Standard specifies the binding between the MAL and the Hypertext Transfer Protocol (HTTP) and defines an XML encoding for MAL data types.

The HTTP transport binding is expected to be used in situations where external, lower bandwidth, connections are expected to be used. For example where one organisation is providing a service to another, external, organisation.

See R[x] for further reading.

## Service specifications

In the following sections the current and future service specifications are outlined. Where they are published documents the relevant document reference is given.

### Common

The Common services provide infrastructure services support the mission operations services, specifically Directory, Login, and Configuration.

See section 4.4 of this document and R[x] for further reading.

### Mission Operations Monitor & Control Services

The M&C service provides the basic ability to monitor and control a remote entity. It provides three basic classes of information:

* Actions:
  + Allow executable tasks to be invoked and their evolving status to be monitored: spacecraft telecommands are an example of an action.
* Parameters
  + Provide status monitoring capability but also may have their value set.
* Alerts
  + Provide a mechanism for asynchronous notification of operationally significant events.

In addition to the services provided for the above, it provides the following services:

* Check
  + Supports the definition of checks that are applied to parameter values; includes check types such as “limit check”, “constant check”, and “delta check”.
* Statistic
  + Supports the association of parameters to defined statistical evaluations (e.g., min, max, mean, standard deviation) and report the resultant statistics evaluations periodically.
* Aggregation
  + Provides the capability to acquire several parameter values in a single request or be delivered periodically.
* Conversion
  + A functional extension of the other services is to add the engineering unit conversion capability.
* Group
  + Defines the concept of a group that allows the other services to refer to sets of objects by a group identifier rather than each one individually.

See R[x] for further reading.

### MDPD

The MDPD specification contains a set of services which provide controlled access to mission data for the community of users who do not have direct access to a system’s monitoring and control facilities. It uses the concept of a data product and supports both batch and stream data distribution to allow delivery of any type and format of data to authorized users.

See R[x] for further reading.

### Planning

[To do]

See R[x] for further reading.

### Automation

S[To do]

ee R[x] for further reading.

### File services

The File system and transfer management services provide two capabilities for interacting with remote file stores; firstly, management of that file store (list, rename, move, delete, etc), and secondly management of transfer of files to/from remote file store. It does not define any specific mechanism for the actual fire transfer, that is delegated to what ever is appropriate for the deployment ((S)FTP on the ground and CCSDS CFDP in space for example).

### Others…

# Future directions

Chapter length estimate: 4-6 pages.

This section gives examples of future directions for the MO standards and also some examples of how they may be deployed in future scenarios:

* The idea of an enterprise MO bus between Agencies for more rapid service deployment
* MO onboard

# Annex/Appendix: Glossary of MO Terms

This is a complete list of MO-specific terminology, included for reference. For each term, there will be a brief description and a chapter/section reference indicating where to go for more information.

# Annex/Appendix: Relationship with ECSS PUS

This is the question we are most often asked about, especially (but not exclusively) in Europe. The ECSS PUS is very well known around the world and the relationship between MO and PUS can be very tricky to explain, especially to someone who is deeply familiar with PUS. It is therefore worthwhile spending a few pages describing this, and is placed in an appendix so that it sits outside the main flow of the document.