The Consultative Committee for Space Data Systems

Recommendation for Space Data System Practices

# MISSION OPERATIONS REFERENCE MODEL

## RECOMMENDED PRACTICE

## CCSDS 520.1-M-1

## MAGENTA BOOK

July 2010

The Consultative Committee for Space Data Systems

**Recommendation for Space Data System Practices**

# MISSION OPERATIONS REFERENCE MODEL

**RECOMMENDED PRACTICE**

**CCSDS 520.1-M-1**

**MAGENTA BOOK**

**July 2010**

# AUTHORITY

| | |
|---|---|
| Issue: | Recommended Practice, Issue 1 |
| Date: | July 2010 |
| Location: | Washington, DC, USA |

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*, and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Space Communications and Navigation Office, 7L70
Space Operations Mission Directorate
NASA Headquarters
Washington, DC 20546-0001, USA

# STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommendations** and are not in themselves considered binding on any Agency.

CCSDS Recommendations take two forms: **Recommended Standards** that are prescriptive and are the formal vehicles by which CCSDS Agencies create the standards that specify how elements of their space mission support infrastructure shall operate and interoperate with others; and **Recommended Practices** that are more descriptive in nature and are intended to provide general guidance about how to approach a particular problem associated with space mission support. This **Recommended Practice** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommended Practice** is entirely voluntary and does not imply a commitment by any Agency or organization to implement its recommendations in a prescriptive sense.

No later than five years from its date of issuance, this **Recommended Practice** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Practice** is issued, existing CCSDS-related member Practices and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such Practices or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new Practices and implementations towards the later version of the Recommended Practice.

# FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Practice is therefore subject to CCSDS document management and change control procedures, which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems.* Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

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

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

Member Agencies

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

Observer Agencies

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

# DOCUMENT CONTROL

| Document | Title | Date | Status |
|---|---|---|---|
| CCSDS 520.1-M-1 | Mission Operations Reference Model, Recommended Practice, Issue 1 | July 2010 | Original issue |

# CONTENTS

# CONTENTS (continued)

# CONTENTS (continued)

# 1 INTRODUCTION

## 1.1 PURPOSE OF THIS RECOMMENDED PRACTICE

This Recommended Practice defines a Mission Operations (MO) reference model which provides a common basis for coordinating the development of CCSDS Recommended Standards for MO service specifications and serves as a reference to maintain the consistency of these Recommended Standards.

## 1.2 SCOPE

The scope of this Recommended Practice is the definition of all concepts and terms that establish a common basis for coordinating the development of CCSDS Recommended Standards for MO service specifications.

## 1.3 APPLICABILITY

### 1.3.1 APPLICABILITY OF THIS RECOMMENDED PRACTICE

This Recommended Practice serves as a guideline for the development of compatible Agency standards for MO systems. It is considered normative on the other Blue Book specifications produced by this working group. In this context, Mission Operations refers to end-to-end services between functions, resident on-board a spacecraft or based on the ground, that are responsible for mission operations.

### 1.3.2 LIMIT OF APPLICABILITY

This Recommended Practice is neither a specification of, nor a design for, real MO systems that may be implemented for the control and monitoring of existing or future missions.

## 1.4 RATIONALE

The primary goal of CCSDS is to increase the level of interoperability among Agencies. This Recommended Practice furthers that goal by establishing the basis for a set of MO Services to be used in the operation of ground as well as space-based assets.

## 1.5 DOCUMENT STRUCTURE

This Recommended Practice is organized as follows:

a) Section 1 provides purpose, scope, applicability and rationale of this Recommended Practice and lists the definitions, conventions, and references used throughout the document.

b) Section 2 provides the context of MO, presents the MO documentation structure, and shows how this Recommended Practice fits into that framework. It expands on the scope of this document to provide an overview.

c) Section 3 defines the MO system environment, data handled by a MO system, and introduces MO Services.

d) Section 4 defines an architectural model for the MO system. This architectural model comprises two views:

   1) The functional view defines the functionality that is performed by that component, without regard to the way this functionality is implemented in real systems. The functional view decomposes the layers into elementary components that transfer messages between the peer layers.

   2) The interaction view models the flow of messages inside each of the layers. The interaction view also shows the error return paths and conditions for each of the layers.

e) Section 5 defines the common characteristics of MO Services.

## 1.6 DEFINITIONS

**Software Component** (component)**:** a software unit containing the business function. Components offer their function as Services, which can either be used internally or which can be made available for use outside the component through Provided Service Interfaces. Components may also depend on services provided by other components through Consumed Service Interfaces.

**Hardware Component:** a complex physical entity (such as a spacecraft, a tracking system, or a control system) or an individual physical entity of a system (such as an instrument, a computer, or a piece of communications equipment). A Hardware Component may be composed from other Hardware Components. Each Hardware Component may host one or more Software Components. Each Hardware Component has one or more ports where connections to other Hardware Component are made. Any given Port on the Hardware Component may expose one or more Service Interfaces.

**Service:** a set of capabilities that a component provides to another component via an interface. A Service is defined in terms of the set of operations that can be invoked and performed through the Service Interface. Service specifications define the capabilities, behaviour and external interfaces, but do not define the implementation.

**Service Interface:** a set of interactions provided by a component for participation with another component for some purpose, along with constraints on how they can occur. A Service Interface is an external interface of a Service where the behaviour of the Service Provider Component is exposed. Each Service will have one defined ‘Provided Service

Interface', and may have one or more 'Consumed Service Interface' and one 'Management Service Interface'.

**Provided Service Interface:** a Service Interface that exposes the Service function contained in a component for use by Service Consumers. It receives the MAL messages from a Consumed Service Interface and maps them into API calls on the Provider component.

**Consumed Service Interface:** the API presented to the consumer component that maps from the Service operations to one or more Service Data Units(s) contained in MAL messages that are transported to the Provided Service Interface.

**Management Service Interface:** a Service Interface that exposes management functions of a Service function contained in a component for use by Service Consumers.

**Service System:** the set of Hardware and Software Components used to implement a Service in a real system. Service Systems may be implemented using one or more Hardware and Software Components.

**Service Provider** (provider)**:** a component that offers a Service to another by means of one of its Provided Service Interfaces.

**Service Consumer** (consumer)**:** a component that consumes or uses a Service provided by another component. A component may be a provider of some Services and a consumer of others.

**Service Data Unit (SDU)**: a unit of data that is sent by a Service Interface, and is transmitted semantically unchanged, to a peer Service Interface.

**Binding**: the access mechanism for a Service. Bindings are used to locate and access Service Interfaces. Services use bindings to describe the access mechanisms that consumers have to use to call the Service. The binding specifies unambiguously the protocol stack required to access a Service Interface. Bindings may be defined statically at compile time or they may use a variety of dynamic run-time mechanisms (DNS, ports, discovery).

**Service Connection** (connection)**:** a communications connection between a Consumed Service Interface and a Provided Service Interface over a specific Binding. When a component consumes the services of a provider component, this link is known as a Service Connection (connection).

**Service Extension**: addition of capabilities to a base Service. A Service may extend the capabilities of another Service with additional operations. An extended Service is indistinguishable from the base Service to consumers such that consumers of the base Service can also be consumers of the extended Service without modification.

**Protocol Stack:** the stack of Protocol Layers required for communication.

**Protocol Layer:** the implementation of a specific Protocol. It provides a Protocol Service Access Point to layers above and uses the Protocol Service Access Point of the layer below.

**Protocol Service Access Point (SAP):** the point at which one layer's functions are provided to the layer above. A layer may provide protocol services to one or more higher layers and use the protocol services of one or more lower layers. A SAP defines unambiguously the interface for a protocol that may be used as part of a Service Interface Binding specification.

**Protocol:** the set of rules and formats (semantic and syntactic) used to determine the communication behaviour of a Protocol Layer in the performance of the layer functions. The state machines that operate and the protocol data units that are exchanged specify a protocol.

## 1.7 CONVENTIONS

### 1.7.1 STYLE

Within this Recommended Practice, each formal statement stands in a paragraph by itself and is identified uniquely by a subsection number or by a combination of subsection number and list item number.

### 1.7.2 NOTES

Notes are not formally part of this Recommended Practice. They are isolated from the formal statements and are introduced by the word NOTE.

NOTE – This is an example of a note.

### 1.7.3 USE OF CAPITAL LETTERS

Names of system components, data units, and other elements of the reference model are shown with the first letter of each word capitalized.

### 1.7.4 DRAWING CONVENTIONS

#### 1.7.4.1 General

In figures illustrating this reference model, UML modelling diagrams are used (see reference [B1]). For the specification of the service model the following UML diagrams are used to represent specific views of the model.

### 1.7.4.2 Services and Service Extension

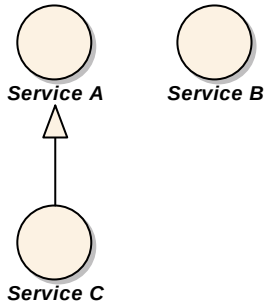


**Figure 1-1: Service Extension Drawing Convention**

Figure 1-1 shows three services (A, B, and C) and that Service C is an extension of Service A.

### 1.7.4.3 Components, Service Interfaces and Bindings

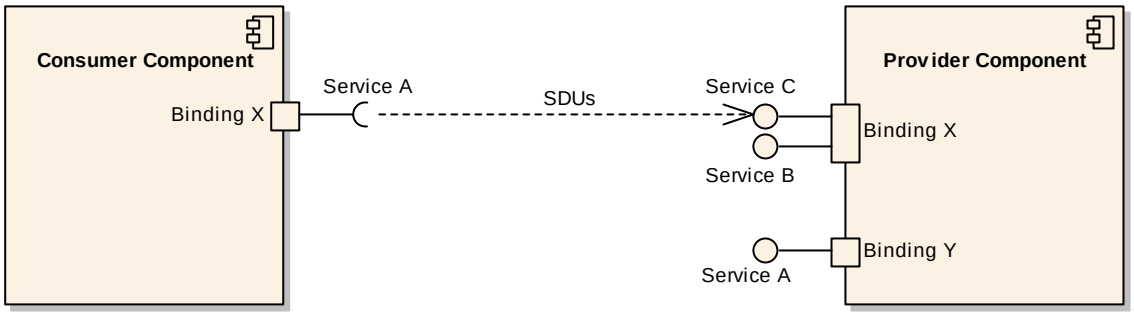


**Figure 1-2: Component Drawing Convention**

Diagram key:

- UML Components are Software Components.
- Small boxes on the edge of components are Bindings, the label next to these shows the actual binding in use.
- UML Provided Interfaces (Lollypops) are Provider Service Interfaces and must only be attached to Bindings.
- UML Required Interfaces (Cups on sticks) are Consumer Service Interfaces (that reference provider interfaces) and must only be attached to Binding.
- Service Connections are shown as dashed lines.
- SDUs travel over Service Connections.
- Consumer and Provider interfaces and Bindings must match.

Therefore, figure 1-2 shows two Software Components (‘Consumer Component’ and ‘Provider Component’), where ‘Consumer Component’ consumes ‘Service A’ through a specific Binding ‘X’. ‘Provider Component’ provides ‘Service A’ through a specific Binding ‘Y’, but also ‘Service B’ and ‘Service C’ through specific Binding ‘X’. Because ‘Service C’ is an extension of ‘Service A’, ‘Consumer Component’ is able to consume it using a ‘Service A’ consumer service interface, this connection is shown as a dashed line.

#### 1.7.4.4 Software and Protocol Layering

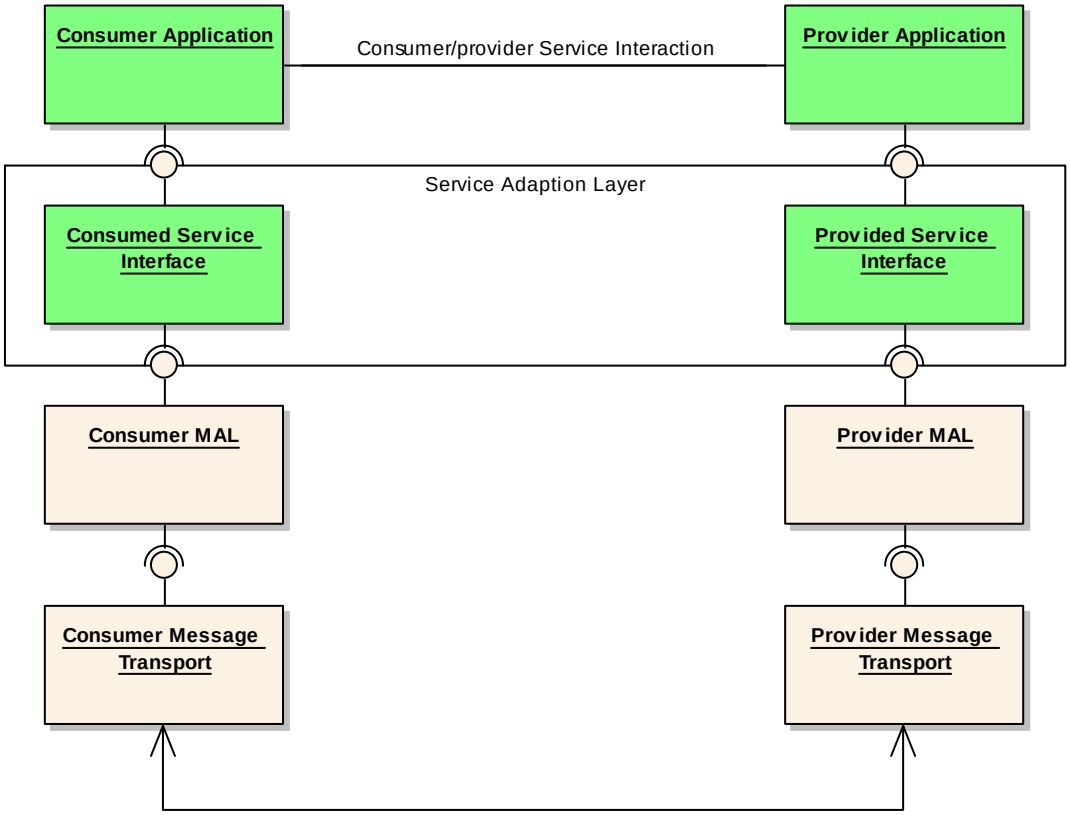


**Figure 1-3: Layering Drawing Convention**

Figure 1-3 shows a software and protocol stack for two Software Components. Protocol layers are shown as UML Objects, software layers are shown as green UML Objects. The UML assemblies (joined lollypop and ‘cup on stick’) show a Protocol Service Access Point.

Standard UML sequence diagrams are used to show the interaction sequence between layers and standard UML class diagrams are used to show structure relationships.

## 1.8 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommended Practice. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Practice are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Documents.

[1] T. Berners-Lee, R. Fielding, and R. Fielding. *Uniform Resource Identifier (URI): Generic Syntax*. STD 66. Reston, Virginia: ISOC, January 2005.

[2] *Mission Operations Message Abstraction Layer*. Recommendation for Space Data System Standards, CCSDS 521.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, [forthcoming].

[3] *Spacecraft Monitor and Control—Common Services*. Draft Recommendation for Space Data System Standards, CCSDS 521.1-R-1. Red Book. Issue 1. Washington, D.C.: CCSDS, September 2007.

NOTE – Informative references are listed in annex B.

# 2 MISSION OPERATIONS SERVICE CONCEPT

## 2.1 OVERVIEW

The services given in reference [B2] are based on a generic service pattern. This pattern covers not only the service interface but also includes the configuration data and history associated with the service. The basic service interface is illustrated in figure 2-1.

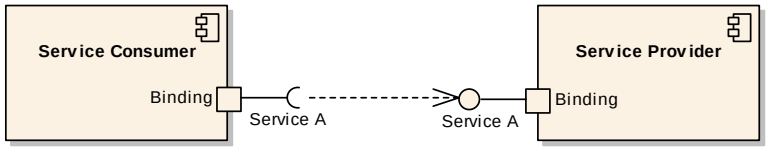


**Figure 2-1: Generic Service Model**

The pattern comprises four main components:

- The **Service Provider** is responsible for supporting the service functions.
- The **Service Consumer** is a user of the service functions, and is typically either a Human-Computer Interface, or another Software Component.
- The **Service Configuration** (not shown) specifies the entities that exist for a specific instance of the service. This specification must be available to both provider and consumer if they are to communicate effectively; however, for simple services it may be implicit for one or both components if the configuration is hard-coded into it.
- The **Service History** (not shown) maintains persistent storage of service history, such that a consumer can retrieve historical information for the service.

A most complex example of the pattern using multiple components is shown in figure 2-2.

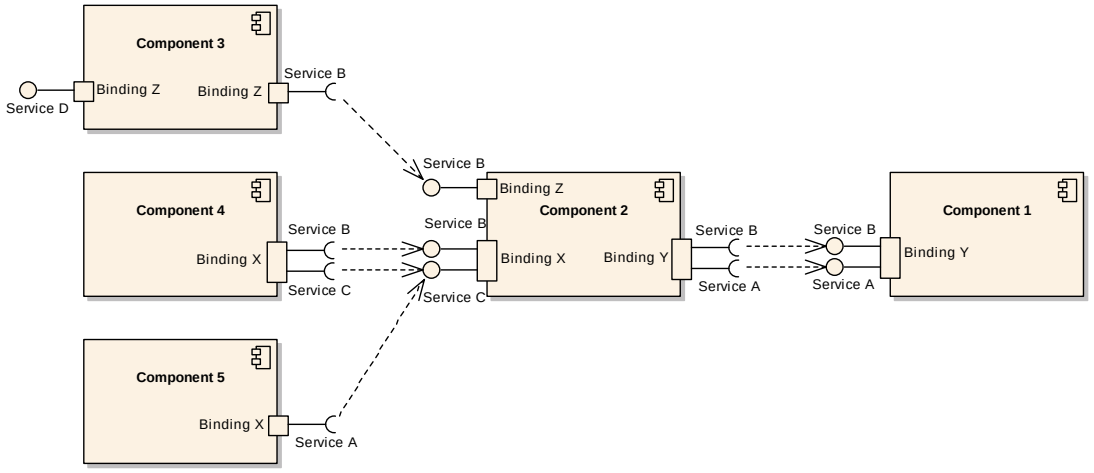


**Figure 2-2: Complex Service Model Example**

It shows how components can be linked together to provide more complex functions and how components can augment services provided by other providers.

## 2.2 PATTERNS OF INTERACTION

An operation of a service is invoked through the exchange of a set of messages between a service provider and consumer and forms a pattern of interaction. Analysis of the services given in reference [B2] shows that there are limited numbers of these patterns of interaction that can be applied to all currently identified services.

Standardising a pattern of interaction, which defines the sequence of messages passed between consumer and provider, makes it possible to define a generic template for an operation of a service.

The Message Abstraction Layer (MAL—reference [2]) defines this limited set of generic interaction patterns (templates) that must be used by services defined in the MO service framework. Each operation of a service is defined in terms of one of the MAL interaction patterns.

By stating that a given operation is an instance of a predefined pattern, the service specification can focus on the specifics of that operation and rely on the standard pattern to define the messaging rules.

For example, if an operation named ‘sendCommand’ were defined and if it were to be stated that it is an instance of a pattern called ‘SUBMIT’, then the ‘sendCommand’ operation could be separated into two parts: 1) the pattern of messages that are exchanged (the ‘SUBMIT’ pattern) and 2) the meaning of those messages and what ‘sendCommand’ does. If the pattern is defined as a standard (‘SUBMIT’), the service specification that defines ‘sendCommand’ need define only the meaning of the messages and what the operation does. The MAL defines this standard set of patterns.

## 2.3 MESSAGE ABSTRACTION

To provide implementation language and message transport independence, all operations of a service must be defined by a language/platform/encoding-agnostic specification. The MAL defines this set of basic data types, and how they must be used to build up the messages that make up the operations of a service, as an abstract API. This abstract API then has only to be mapped once, in a MO standard, to a specific implementation language or transport encoding to apply to all services that are defined in terms of the MAL.

In addition to the patterns of interaction and the abstract API, the MAL provides support for the following:

- concepts such as domain, session, and zone (see 3.5);
- facilities such as access control (authentication and authorisation) and Quality of Service (QoS) (see 3.6 and 3.7, respectively).

## 2.4 MISSION OPERATIONS SERVICES

Whilst the MAL provides message abstraction and generic concepts such as access control and QoS, above this exists the Mission Operations Services (MO Services). The MO Services comprise:

- the Common Object Model (COM);
- the Common Services;
- the Functional Services.

All three are defined in terms of the MAL. The COM provides a generic service template which is used to define the Common and Functional services. This ensures a common approach and simplifies the task of defining each service.

The term Mission Operations Services is used to collectively refer to both Common Services and Functional Services.

The Common Services are:

- Directory

  Service publish and lookup.

- Login

  Operator login.

- Configuration

  Service configuration management.

- Interaction

  Operator interaction.

- Retrieval

  Historic archive retrieval and management.

- Replay

  Replay session management and control.

The layering of the message transport, MAL, MO Service Adaption Layer and the service provider and consumer applications is shown in figure 2-3 (each layer builds upon the layers below).

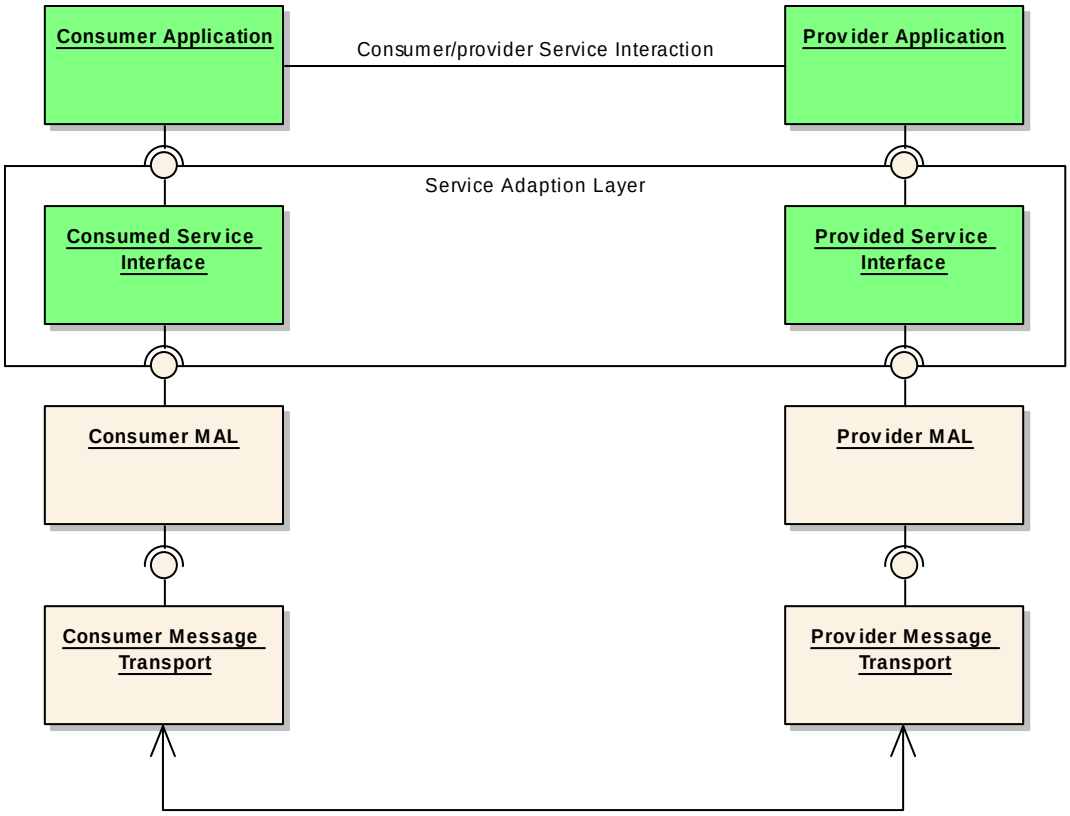


**Figure 2-3: Service Stack View**

The MO Service Adaption Layer is responsible for converting between the simple interaction pattern interface of the MAL to a higher service specific interface used by the applications.

NOTE – A benefit of implementing multiple services over a message abstraction layer is that it is easier to bind them to different underlying technologies and protocol encodings. All that is required is an 'adapter' layer between the MAL and the underlying protocol to enable all services over that technology. This can be either an implementation of the MAL that is bound to a specific technology or an implementation that supports multiple technologies. Hence the same service can be implemented over ground-based network technologies and middleware, or it could even be carried across the space link itself.

The services, in the form of standard language-specific APIs, themselves provide the 'plug-and-play' interface for applications, allowing them to be integrated and deployed wherever is appropriate for the mission.

## 2.5 MISSION OPERATIONS FRAMEWORK

There is one and only one abstract definition of the MAL that is to be used as the means to define the interactions among services. Each Service has one and only one definition for the service, in terms of what that service is, what it does, and what operations it offers. The service specifications use the MAL and may use the COM. They may also reference other supporting services.

The service specifications and the MAL are abstract in their definition; they do not contain any specific information on how to implement them for a particular programming language or transport encoding.

Moving from the abstract to the implemented system necessitates two other specifications. One is the Language Mapping that states how the abstract MAL and Service specifications are to be realised in some specific language; this defines the API in that language. The second is the transport mapping from the abstract MAL data structures into a specific and unambiguous encoding of the messages and to a defined and unambiguous mapping to a specific data transport. It is only when these mappings are defined that it is possible to implement services that use the MAL interface and use the transport bindings to exchange data.

The MAL and service specifications are supplemented by a set of standard MO specifications for implementing the MAL in specific programming languages and also for mapping the MAL to a specific message encoding and transport.

NOTE – Only the MAL specification needs to be mapped to a specific implementation language, encoding or transport. The service specifications are defined in terms of the MAL and therefore the same mapping applies to these services unmodified.

Of the Recommended Standards produced for the MO framework specification, each book falls into one of the following four categories:

a) Language Mapping

   One book exists for each mapping from the MAL to the specific implementation language.

b) MAL Specification

   Only one book exists defining the MAL.

c) Service Specifications

   Only one book exists for each service specification. The service specification may use aspects of the COM specification, in which case this book would also be required.

d) Transport Mapping

   One book exists for each mapping from the MAL to the specific transport and encoding.

Language-mapping Recommended Standards define a standard mapping of the MAL to a specific implementation language. This mapping provides a standard API for application developers to develop against, allowing the reuse of both applications and also MAL implementation.

Service Specification Recommended Standards define the high-level application level service. They use the abstract notation defined in the MAL specification and detail the operations, behaviour, and required consumer behaviour for the service. Existing CCSDS application service specifications and data standards are expected to be referenced by these standards where applicable, such as Navigation data standards (references [B8], [B9], and [B10]) and implementations are expected to use relevant CCSDS standards also where applicable such as Spacecraft Onboard Interface Services (SOIS) standards (reference [B11]).

Transport-mapping Recommended Standards define technology mappings to specific transports, such as CCSDS Space Link Extension (SLE) Services (references [B3] and [B4]), CCSDS File Delivery Protocol (CFDP) (reference [B5]), CCSDS Asynchronous Messaging Service (AMS) (reference [B6]), and message encodings such as XML, ASCII, and CCSDS Space Packets (reference [B7]). These mappings allow system engineers to choose a message transport and encoding appropriate for a specific deployment.

To provide a working implementation of a service, one book of each category must be selected and used, i.e., the MAL specification, a programming language mapping for the MAL, a transport and data encoding mapping for the MAL, and one or more fully defined service specifications.

## 2.6 INTEROPERABILITY, APPLICATION PORTABILITY, AND DEPLOYMENTS

The MAL is defined in a language- and protocol-agnostic manner as it only standardises the message exchange at an information level; it leaves the language used to implement it, the encoding mechanism, and the transport used open to be selected in the system implementation phase.

This abstraction in the specification of the MAL allows two types of flexibility to be provided: firstly the separation of encoding and transport allows flexibility in deployment (allows the changing of encoding and transport), and secondly the choice of language allows portability of an application with a specific implementation of the MAL (allows reuse of software across missions).

Using the book numbering from 2.5:

a) language mapping;

b) MAL specification;

c) service specifications;

d) transport mapping.

For two agencies to interoperate they must standardise on the transport and encoding selected (books selection ‘bcd’ must match). The choice of implementation language (book selection ‘a’) at each agency is hidden from the other by the MAL and therefore not required for entity interoperability:

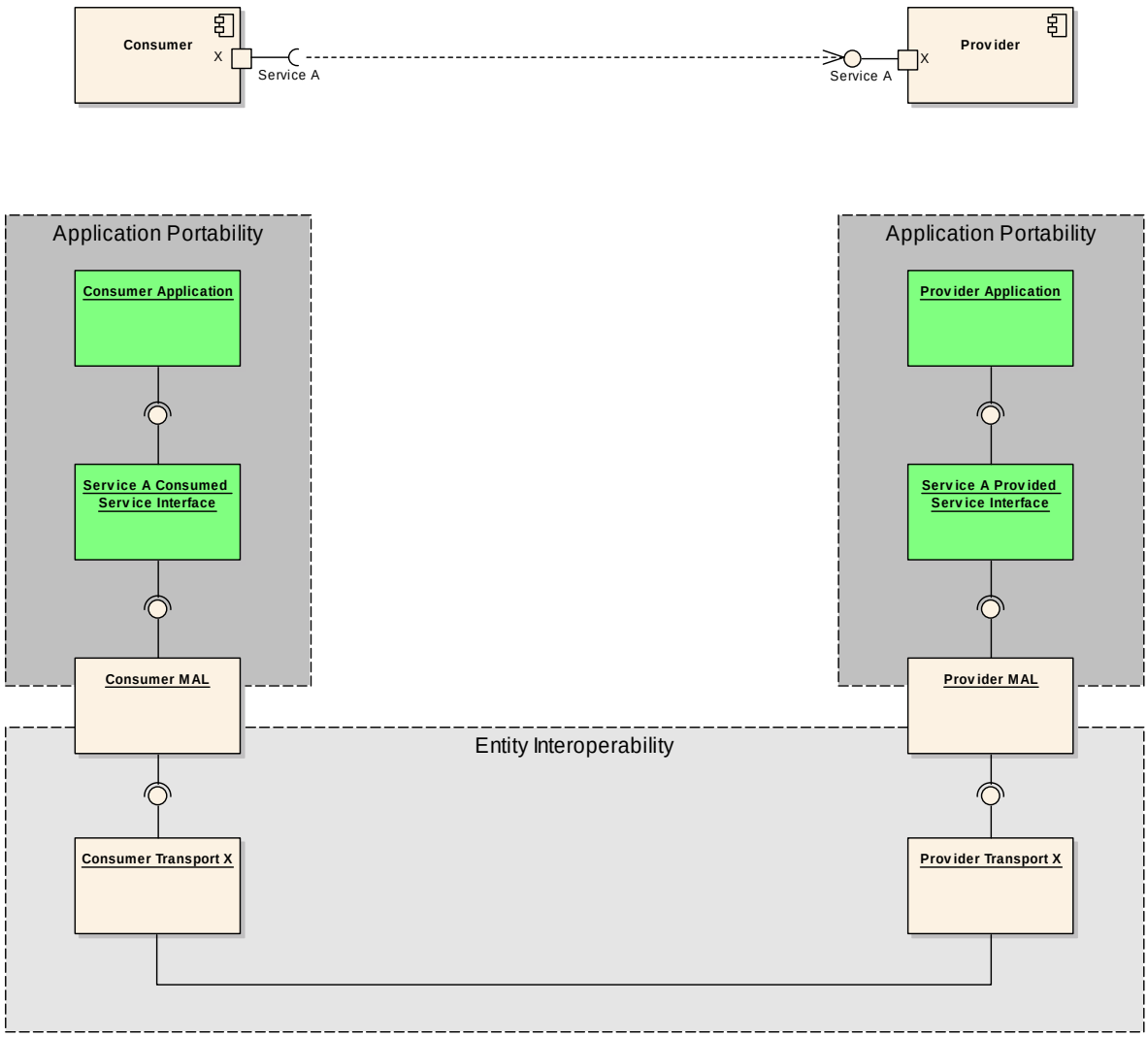


**Figure 2-4: Example Entity Interoperability**

Figure 2-4 shows a consumer and provider interacting over a single transport. The two components are required to standardise on the transport and encoding to interoperate but the choice of implementation language is separate for each component.

The key benefits of this approach are:

– support for heterogeneous implementations;

– ability to change the transport infrastructure within a system, without major re-work to the application-level software; only the mapping to the transport encoding needs to be redone.

The separation of information interoperability (MAL and higher layers) and protocol interoperability (encoding and transport) permits creation of a protocol-matching bridge or Gateway that allows translation from one encoding/transport choice to another:

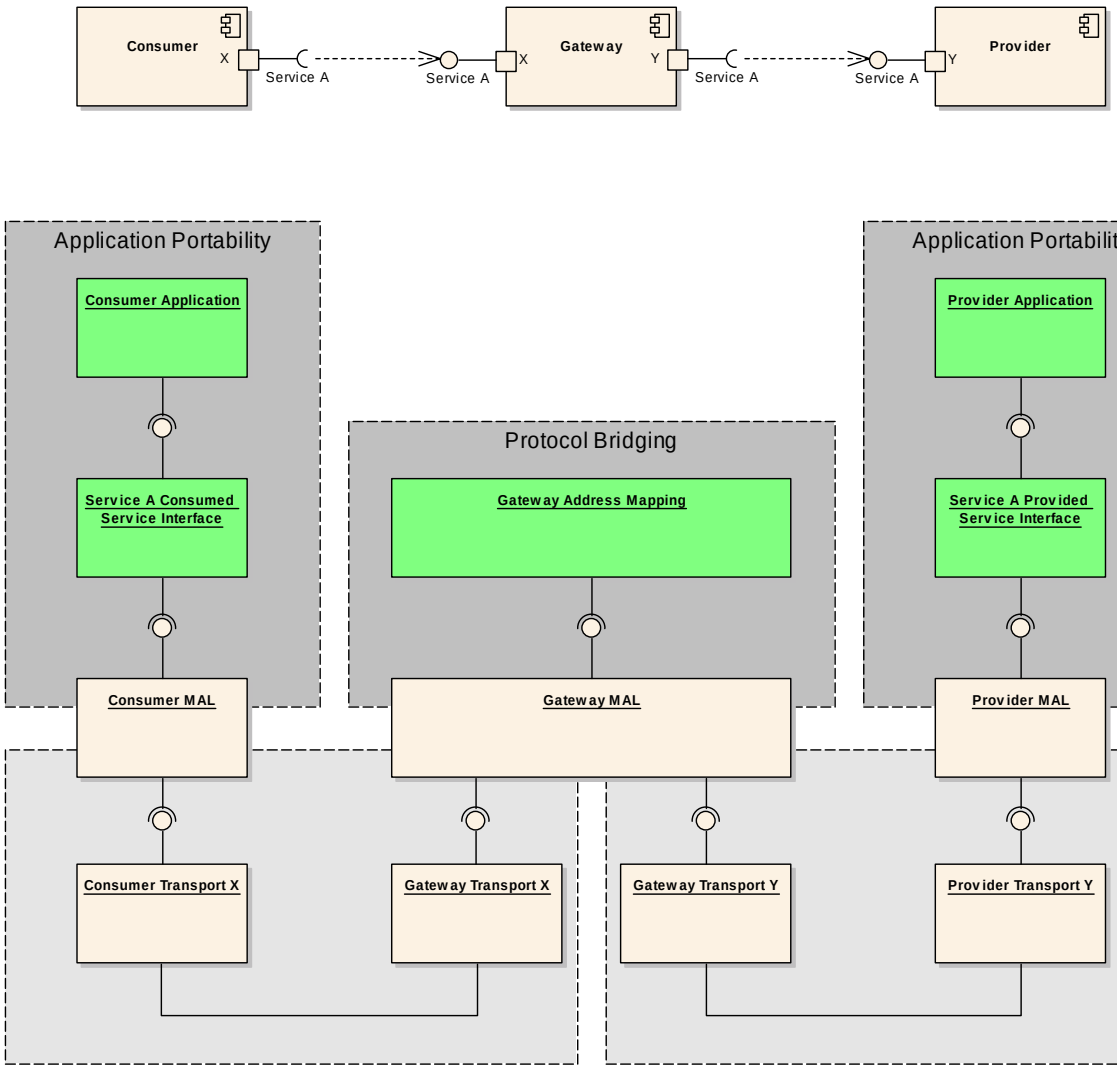


**Figure 2-5: Protocol Bridge Example**

Figure 2-5 shows that the consumer and provider components are still fully interoperable even though they utilise different transport encodings. An implementation of the MAL may be fixed to one specific encoding and transport, but the MAL specification permits this to still be interoperable with other implementations using a different transport/encoding through the use of the protocol bridge. It should be noted that a custom transport/encoding can be

used, for example, to utilise existing infrastructure. All that is required is a mapping from the MAL to that transport.

The protocol bridge concept can also be used to provide service extension, where one service provider extends the capabilities of another either through the provision of extra information (e.g., statistical services) or extra operations:

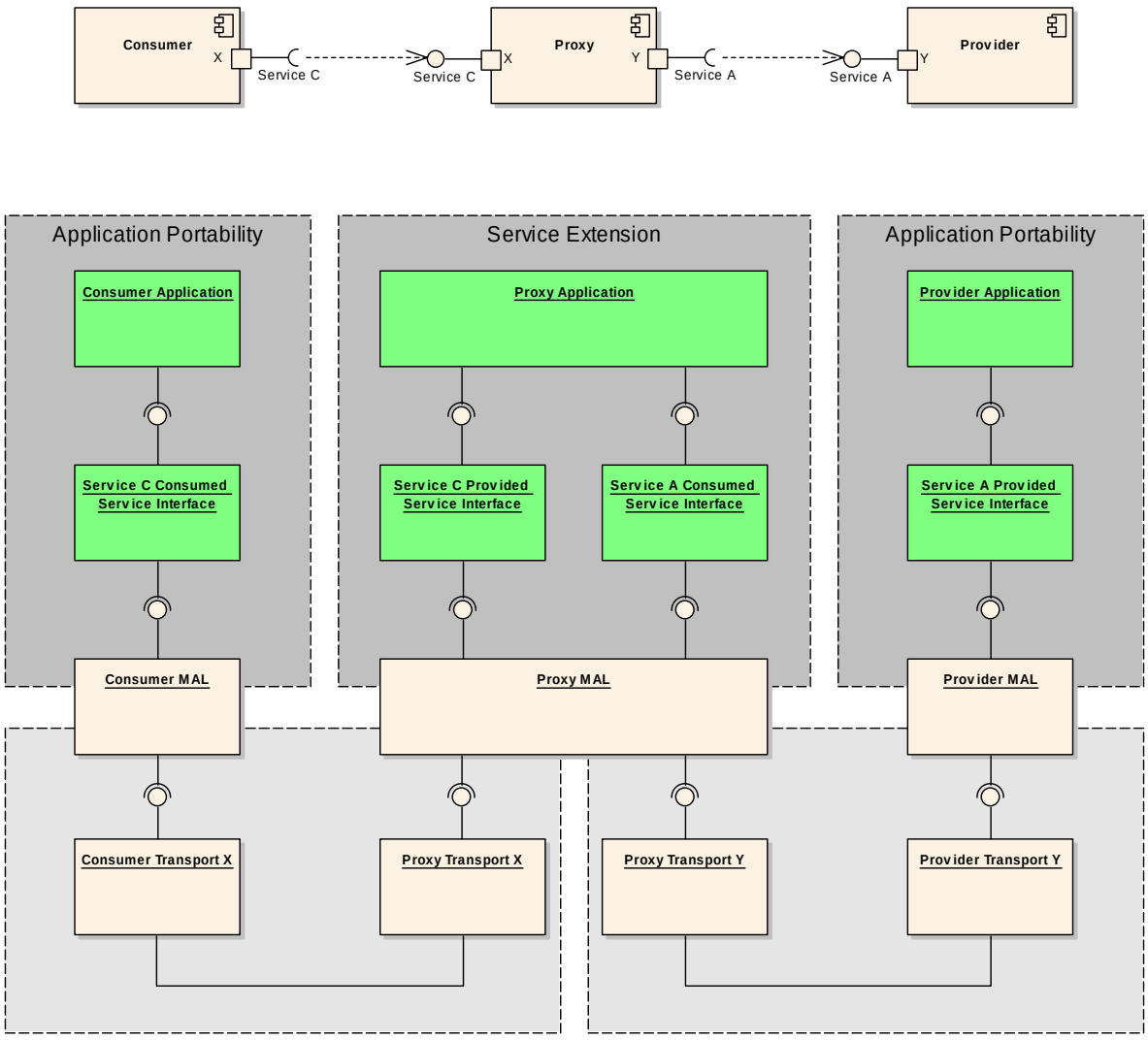


**Figure 2-6: Service Extension Example**

Figure 2-6 shows the Proxy component acting as a consumer of Service A from the Provider component and as a provider of Service C to the Consumer component. A proxy provides a way of exposing a service to external consumers where direct visibility would not be desirable (for example for security reasons). In the example the Proxy component is also acting as a protocol bridge; however, this is not required.

# 3 MO CONTEXT AND CONCEPTS

## 3.1 OVERVIEW

This section defines the context of a MO service deployment and extends the concepts outlined in section 2.

## 3.2 SERVICE CONTEXT

A deployment of a MO service is defined as consisting of two components, a service provider of the MO service and a consumer of that service:

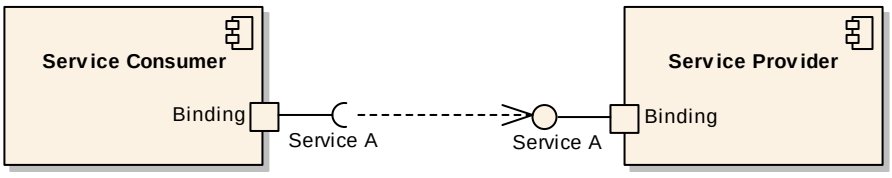


**Figure 3-1: Basic MO Service Deployment**

a) The MO service concept does not define or limit the locations of the two components. As long as the two components can communicate with each other using the relevant communications transport, then they may interact.

b) The service provider can be located wherever is appropriate for the deployment, be it on ground inside a control system or at another ground facility, at the ground station itself, onboard an orbiting spacecraft or even onboard a deep space craft or lander. The same applies for the service consumer.

c) The two components are also not required to be in separate locations. The concept supports ground-to-ground, onboard-to-onboard and space-to-space deployments as well as the more traditional ground-to-space deployments.

d) A service consumer interacts with a service provider through the exchange of service messages (referred to as messages from this point onwards). (See 3.4 for further explanation.)

NOTE – It should be noted that other components of a system may prevent communications, but the concept does not limit it by definition.

## 3.3 SERVICE DECOMPOSITION

The MO service consumer and provider components are decomposed as follows, with equivalent layers on both sides:

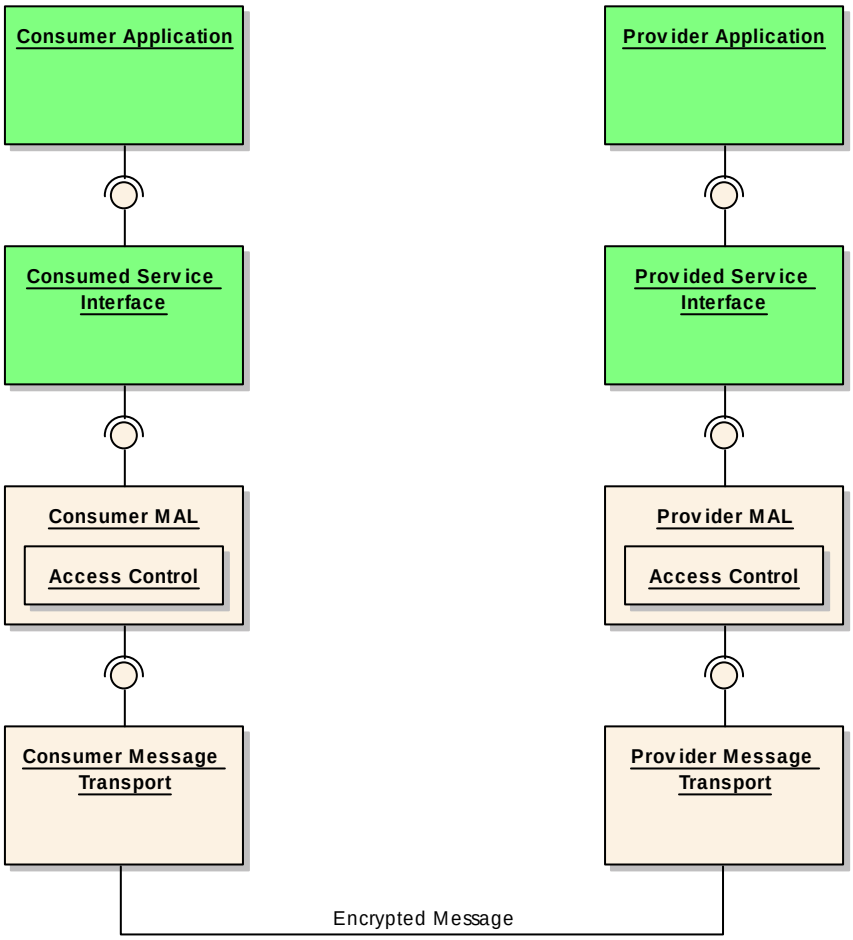


**Figure 3-2: Layered MO Service Decomposition**

a) **Consumer Application**. Consumes the services provided by the service provider.

b) **Consumed Service Interface**. Responsible for converting from the messages provided by the MAL abstract service below it to the service interface exposed to Consumer Application.

c) **Consumer MAL**. Provides the standard messaging service that is used by the Service Interface to communicate with its service provider peer. All messages transported by the MAL are filtered through the Consumer Access Control component via a standard abstract interface.

d) **Consumer Access Control**. Implements a standard service interface defined by the MAL and provides access control to services from the consumer point of view. It can

reject messages or augment with security credentials any messages that pass through the MAL. The actual access control policy in place is deployment specific.

e) **Consumer Message Transport**. Implements a standard message transport interface defined by the MAL for the transport of messages from a source to a destination. It is responsible for converting the message from the language-specific representation to the wire-level representation required for that transport. Combines both the relevant messaging component (most probably a COTS) and an adaptation from the MAL interface to that software.

f) **Provider Application**. Implements the service-specific behaviour of the relevant MO service specification.

g) **Provided Service Interface**. Responsible for converting from the messages provided by the MAL abstract service below it to the service interface exposed to a provider application.

h) **Provider MAL**. Provides the standard messaging service that is used by the Service Interface to communicate with its service consumer peer. All messages transported by the MAL are filtered through the Provider Access Control component via a standard abstract interface.

i) **Provider Access Control**. Implements a standard service interface defined by the MAL and provides access control to services from the provider point of view. It can reject messages or augment with security credentials any messages that pass through the MAL. The actual access control policy in place is deployment specific.

j) **Provider Message Transport**. Implements a standard message transport interface defined by the MAL for the transport of messages from a source to a destination. It is responsible for converting the message from the language-specific representation to the wire-level representation required for that transport. Combines both the relevant messaging component (most probably a COTS) and an adaptation from the MAL interface to that software.

NOTE – Whilst the above layers must logically be present in both the consumer and provider, there is no requirement for them to physically be layered using software APIs, etc. For example, it is perfectly legitimate for a service consumer or provider to hardcode all features of the layers as long as all MO Services, MAL and transport requirements are adhered to.

## 3.4 SERVICE MESSAGES

Service messages (messages) are exchanged between a service consumer and provider to initiate and report progress of the required service operation.

a) A message is an abstract entity that is passed from one component to another component in the MO service model.

b) Its in-memory representation is dependent on the programming language in use and the point in the stack; for example, it may be in one form in the MO Service Adaption Layer and a completely different one in the Transport Layer.

c) The MAL specification and the Programming Language mapping define the representation required for that language at the interfaces between the layers.

d) The on-the-wire representation is dependent on the transport and encoding used by that transport. Conceptually the model is only concerned with information exchange; as long as the on-the-wire representation preserves all information (or it can be reconstructed by the Transport Layer), then on-the-wire presentation is only a concern for the Transport Layer.

e) The Transport layer is responsible for the conversion from the in-memory message form to an on-the-wire Protocol Data Unit (PDU) for that transport. It is the Transport that provides the message-level interoperability between two entities.

f) A single message may be split into several physical PDUs that can be transported over different Transports. As long as the message can be rebuilt by the receiver, this is permitted. It is the responsibility of the relevant Transport layer to coordinate this splitting and recombination. This is expected to be used when summary information of a message is sent by one technology and the payload of a message is large and sent via another technology (such as files). In this case it would also be the responsibility of the Transport layer to coordinate the separate transport technologies.

## 3.5 SERVICE DEPLOYMENT

### 3.5.1 GENERAL

**3.5.1.1** Services may be deployed in many different configurations; the arrangement of these is a deployment consideration.

**3.5.1.2** A service provider may support one or more services; however, it is also possible that another service provider supports the same services (e.g., for redundancy).

**3.5.1.3** A single service can have many service consumers. For example, many operators could wish to display data from the same service. Conversely, a single service consumer may be associated with multiple services, potentially from multiple service providers. For example, an overall mission timeline may require data from multiple sources.

**3.5.1.4** The MAL addresses the multiplicity of service instances within its design to allow such deployments by identifying each deployed service with a *session*, *domain*, and *network zone* (see 3.5.2, 3.5.3, and 3.5.4, respectively).

**3.5.1.5** Each MAL message contains fields that hold the session, domain, and network zone that it belongs to.

**3.5.1.6** The arrangement and definition of the session, domain, and network zone is a deployment consideration and outside the scope of this specification.

NOTE – In a real-world deployment of a service there may be many instances of that service, in many service providers, and many service consumers.

### 3.5.2 SESSION

**3.5.2.1** For a given service deployment it may be possible to observe both current (live) data and also data replayed from stored history. In a given system it may be possible to observe both live and historical data in parallel, and it may also be possible to observe data originating from a simulator or test configuration in parallel to that originating from the live operational system.

**3.5.2.2** The entities being controlled in the real, simulated, or test cases (and monitored in both these and historical replay cases) may be the same, and in order to distinguish these parallel operational scenarios it is necessary to partition data by operational *session*.

**3.5.2.3** While partitioning can be achieved physically, in a distributed network environment it is required that services are defined in such a way that *session* is explicit to avoid any possibility of confusion, and to enable data to be combined in a single system.

**3.5.2.4** The data delivery of a session has two attributes, the epoch and the rate. Services are expected to operate at the correct rate for real operations using the current epoch; however, a simulation might be able to use a different epoch and possibly rate. Replay of a session may be run at a faster or slower rate than real-time, for example, a replay of the real-time session's history at a slower speed than originally received.

**3.5.2.5** In the context of MO, the term *session* is used to refer to a coherent data source, relating to one of the following:

a) the operational system in live real-time;

b) the operational system in replay;

c) a simulation of the operational system.

NOTE – Multiple *sessions* may exist in parallel (particularly for cases c with a).

### 3.5.3 DOMAIN

**3.5.3.1** A service does not always simply relate to the control of a single spacecraft. Many existing space agencies and missions require the control of multiple remote assets such as spacecraft fleets and constellations, ground stations, etc. In order to ensure that unique referencing of entities and data items is possible, the concept of a hierarchy of system components or *domains* is required.

**3.5.3.2** A domain is defined as a logical namespace that the entities modelled by services belong to. It provides a *namespace for operational data*, such as telemetry monitoring parameters and actions.

**3.5.3.3** A domain is used to scope the frame of reference of monitoring and control, for example agency>mission>spacecraft>subsystem. However, no explicit relationship between Agency, mission, or other is enforced; it is a deployment decision to define the domains in use for a particular system.

**3.5.3.4** A domain identifier for MO Services is defined as a list of identifiers, each of which narrows the preceding domain, reading left to right, where the leftmost is the most significant.

NOTE – This is the reverse of an Internet address (ccsds.org) which reads right to left, rightmost being most significant.

For example, Action C1234 'Heater C On' for a specific Agency/Mission/Craft becomes:

AgencyY.MissionA.SatB.C1234

*or even*

AgencyY.MissionA.SatB.HeaterC.ON

which cannot inadvertently be sent to AgencyY.MissionX.SatY and executed.

**3.5.3.5** Within a specific configuration, the *domain* may be implicit to allow generic multi-domain operations to be defined and to ensure that the specification of operations is not unduly verbose.

NOTE – The support for multi-domain facilities is not a mandatory requirement of the service specification and it should be noted that the services are designed to cover single domain infrastructures as well as multi-domain.

### 3.5.4 NETWORK ZONE

**3.5.4.1** Network Zone indicates to a consumer how 'local' a service provider is, and is distinct from domain. Domain does not specify physical location or network connectivity. There may sometimes be a coincidental mapping through practicality, but it is not universal.

**3.5.4.2** The definition of the Network Zones is deployment specific; there are no predefined Network Zones. For interoperability the Network Zones in use in a particular deployment shall be pre-agreed between the involved Agencies.

**3.5.4.3** All network traffic in a distributed system can be affected by a pipe-line delay and data link capacity. In the case of offline services, service providers may be restricted by firewall access, link capacity, and link latency. An everyday example is email collection over

a dial-up modem to a remote and protected email server. The mail protocol will try to deliver mail regardless of the ability of the link to support the delivery.

**3.5.4.4** For the purposes of service provision, a system's architecture can be physically modelled in terms of network zones. A service provider specifies which network zone it resides in. When looking up a service in a service directory, a service consumer can specify which zone is preferred. Typically, a service consumer might prefer or be configured to use a local provider, i.e., one that resides in the same network zone as itself.

**3.5.4.5** The network zone is used as part of the lookup of services and also is contained in the header fields of the messages exchanged when interacting with a service provider.

### 3.5.5 SERVICE ADDRESSING

**3.5.5.1** Each service, for a specific domain, session, and network zone is located by an address. This address, represented by a Universal Resource Indicator (URI), allows a consumer to locate and communicate with it.

**3.5.5.2** The URI address itself is not required to contain any reference to the domain, session, and network zone, as the service provider component may support several services. This information is contained in the message header of all messages.

**3.5.5.3** The URI syntax follows that of the Internet Standard 66 (reference [1]), namely:

```
<scheme name> : <hierarchical part> [ ? <query> ] [ # <fragment> ]
```

**3.5.5.4** Each message transport specification shall define the scheme name it uses so that an implementation of the MAL will be able to identify which transport to use for a specific URI. A transport may define multiple scheme names if there is a choice of encodings.

NOTE – A transport is not required to encode URIs as strings if a more efficient alternative is available to it (such as tokenisation or Ids); however, the URI must arrive at the destination with the correct value.

## 3.6 SECURITY AND ACCESS CONTROL

### 3.6.1 OVERVIEW

To ensure that only authorised operational clients have access to service functions, it is critical that some form of authentication, both client and server, is an integral part of the MAL. To avoid the need for a client to support multiple authentication methods, it is highly desirable that all service capabilities use the same mechanism and that client authentication is required only once per client 'login' even if multiple services are used (this does not preclude the ability of an implementation to challenge a security client; however, the mechanism for this is outside the scope of this specification).

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

It is not part of this specification to detail any applicable security methods or standards (that is a deployment decision); however, the MAL supports a generic security and authorisation concept that allows the appropriate mechanism to be used. This concept is similar to that of the MAL's hiding the transport protocol used.

The MAL also does not impose any specific set of access restrictions regarding what operations and services may be accessed by whom but provides a framework of messages and patterns that can be restricted using an appropriate policy for a particular domain, implementation, or agency. For example, a simple spacecraft that is operated by a small enterprise may require only very simple access control provided at the login level, whereas a multi-craft agency with many operators will require a much finer grain of access control.

### 3.6.2 ASPECTS OF SECURITY

Security is typically separated into:

- Data and Data Origin Authentication

  Corroboration of the source of information that is contained in a message.

- Authorisation

  Conveyance, to another entity, of official sanction to do or be something.

- Confidentiality

  Keeping information secret from all but those who are authorised to see it.

- Integrity

  Detecting that information has not been altered by unauthorised or unknown means.

- Non-Repudiation

  Preventing the denial of previous commitments or action.

a) In the MO concept it is assumed that confidentiality is provided by the lower Transport layer and is transparent to the MAL and above. The effect of this is that once a message rises above the Transport layer all encryption will have been removed.

b) Alternative methods must be employed to support the case where confidentiality is required all the way to the payload. One possible mechanism relies on a custom encoding scheme that encodes specific messages privately and uses the normal message-handing functionality to transfer the encrypted information.

c) Authentication and authorisation are the main areas of concern for MO. Non-repudiation and integrity are supported by certain authentication solutions and therefore are possible only with specific message encodings.

d) Authorisation is not possible without authentication (one cannot authorise an operation if one does not know from whom that operation originated) so authentication is mandatory if authorisation is required in a deployment.

e) Therefore there shall be three modes of access control supported:

   1) Nothing

      An open system in which anyone can perform any operation. The system can only log operations performed but not by whom.

   2) Authentication Only

      A closed system in which clients must log in but once in they can perform any supported operation. The system can log who performed what.

   3) Authentication and Authorisation

      A closed system in which everyone must log in and with different levels of access. The system can restrict who performs what.

f) It is a deployment decision which mode of access control a specific system uses.

### 3.6.3 AUTHENTICATION AND AUTHORISATION

Authentication is a function of the specific protocol in use. For example, digital signatures are used to 'sign' a specific message, and through some feature specific to the authentication mechanism, the receiver of that message can confirm that the source actually generated the message. A digital signature is derived from a specific binary representation (or encoding) of that message, and because of this the digital signing can be performed only at the encoding stage. Different encodings may support different authentication technologies and also different ways of representing authentication signatures.

a) Authorisation requires authentication; any process that attempts to restrict access to specific functionality must be able to determine where a specific message originated before it can attempt to perform authorisation. Authorisation is performed by checking that a specific operation is being performed by an authorised consumer. The specific checks are application/deployment specific, but it is expected that it will involve a lookup of some configuration based on the information passed as part of the message.

b) Authorisation shall be performed in the MAL. Presented to the MAL by the lower Transport layer is a generic message with authenticated consumer credentials (which may contain time information for ensuring 'freshness'). It is the responsibility of the MAL to confirm, using an implementation-dependent mechanism through a standard

MAL service interface, that the specific consumer is authorised to perform the operation. If the operation is permitted, then the MAL passes the message to the higher layer as normal; if not, then a standard error message is returned to the consumer indicating an authorisation failure.

c) The Login service provided by the Common Services (reference [3]) provides the mechanisms by which an operator provides his or her security credentials to the system; from this point the MAL performs authentication and authorisation using these verified credentials.

### 3.6.4 AUTHENTICATION AND AUTHORISATION SEQUENCE

Figures 3-3 and 3-4 show the logical sequence for the sending of a message from a service consumer application to a receiving service provider application.

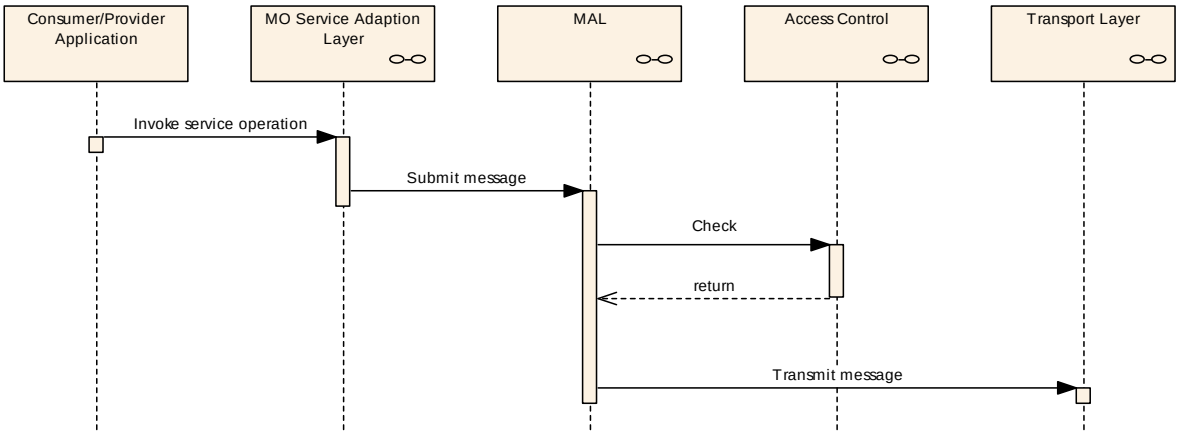


**Figure 3-3: Authentication and Authorisation Sending Sequence**

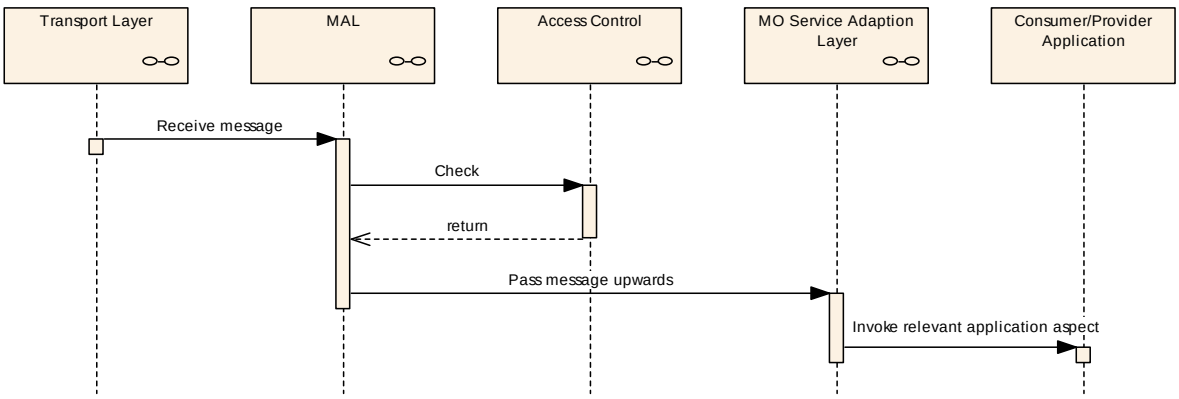


**Figure 3-4: Authentication and Authorisation Reception Sequence**

a) The consumer application invokes a language specific API on the MO Service Adaption Layer. The MO Service Adaption Layer creates a MAL message and passes that to the MAL via the abstract MAL interface.

b) Every operation that is invoked by an application-level service on the MAL must have supplied with it a consumer identifier (the operation identifier is part of the standard MAL message header). The meaning of that consumer identifier is dependent on the security system used for the deployment. This identifier must allow the MAL Access Control implementation to perform a lookup for authorisation purposes.

c) The MAL passes the message with the consumer identifier to the consumer Access Control implementation. This performs any required authorisation checks at the sending side and replaces the consumer identifier with its technology dependent consumer security credentials. It then passes this back to the MAL.

d) The MAL passes the message and consumer credentials down to the Transport layer for encoding to the wire protocol and transportation.

e) The Transport layer performs any actions required for the encoding process, maybe using the consumer access control implementation for digitally signing the encoded message (deployment specific), and then encrypts (if required) the message for transportation.

f) The receiving transport removes any encryption. It then authenticates the message using a deployment specific process—any authentication failure should generate the appropriate MAL error message—and then decodes the message and passes it with the consumer credentials up to the provider MAL implementation.

g) The provider MAL implementation passes the message to its Access Control implementation, which checks any authorisation requirements with its internal permission rules, either rejecting the message or passing the message back to the MAL; it also converts the credentials back into a consumer id.

h) For the case where no authorisation is required, the MAL shall always accept any messages passed to it.

i) For the case when no authentication is required, an empty consumer identifier should be used. It is expected that an implementation of the MAL will be configured to accept this empty consumer identifier and not perform any authorisation of it.

NOTE – The API between the MAL and the higher application layer is a standard API that by definition contains no encryption. The use of a standard API and no encryption exposes a potential security weakness, which can be mitigated by the use of a non-standard hardcoded API between the two, although this approach then removes the flexibility of the standard API. This is a deployment decision.

### 3.6.5 AUTHENTICATION CHALLENGES AND SECURITY HANDOVER

During the lifetime of a security session it may be required by the deployment security concept to periodically challenge the operator to resupply his or her security credentials.

a) The mechanism for reacquiring these credentials is outside of the scope of the specification, as it is driven by the system rather than the operator. It is a concept that shall be supported in the MAL language mappings.

b) Also, there may be a case for the operator to either change his or her current role or to completely change the operator through operations handover. An operation in the Common Login service triggers this handover, and the security implementation in the MAL shall support this if required in the deployment.

c) It shall also be supported that an authentication challenge can be triggered as a result of the handover request; for example, the new operator supplies his or her credentials and then the system challenges the existing operator to supply his or her credentials before the handover takes place.

d) It is required that during a handover there shall be no loss of messages unless the new security credentials restrict them. For example, any ongoing telemetry reception shall not be affected by the handover unless the new security credentials do not permit access to the data source.

### 3.6.6 SECURITY BRIDGE

**3.6.6.1** The protocol bridge concept (see 2.6) can also be extended to an authentication bridge. For example, a control centre can use a Mission Control System (MCS) as a proxy and have fine-grained security/authentication inside on the LAN and then put in place a bridge to step across to the coarse authentication used over the space link, i.e., control centre level:

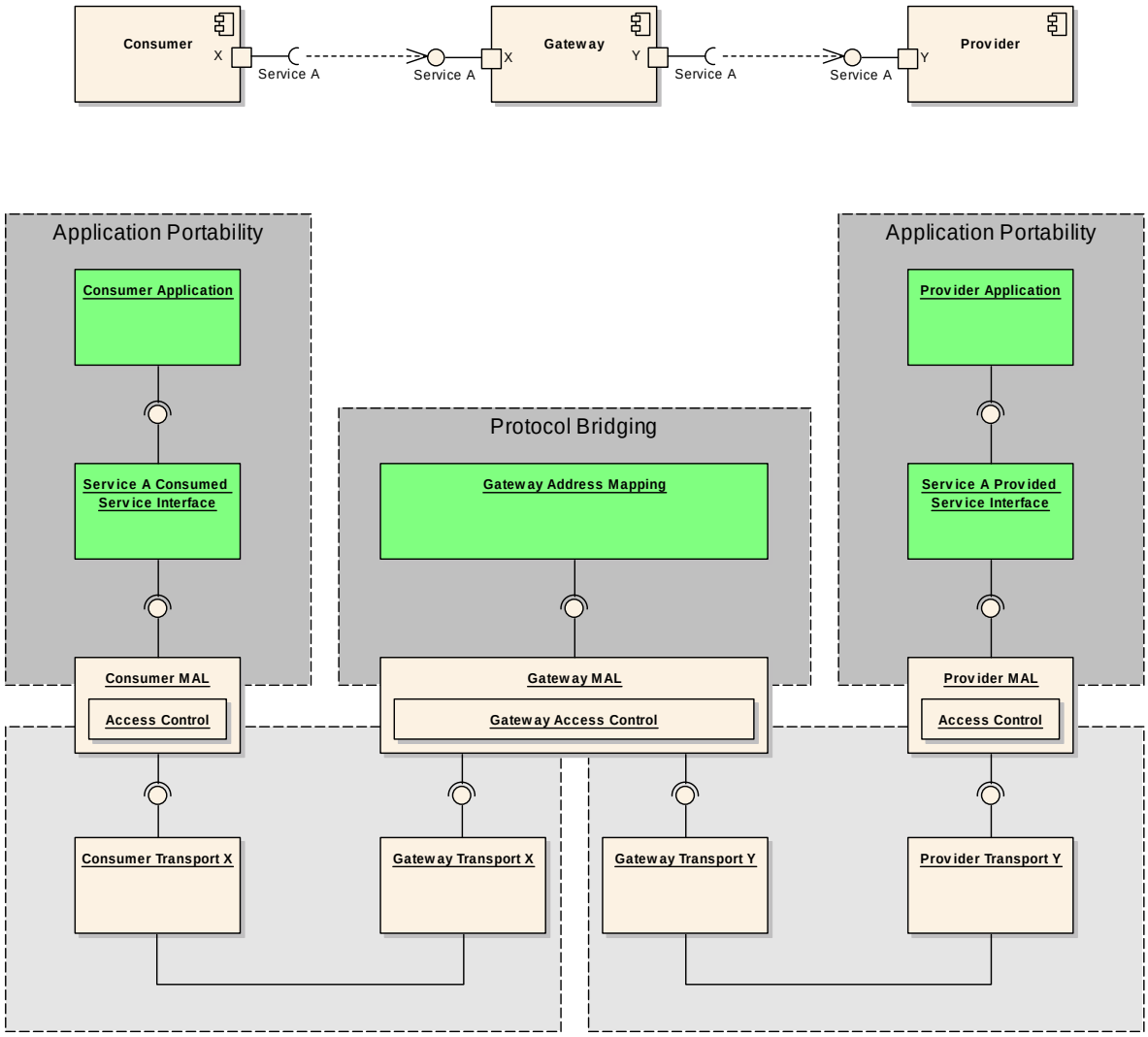


**Figure 3-5: Security Bridging**

**3.6.6.2** By extending the capabilities of the bridge component above the MAL layer it is possible for a bridge component to filter the messages, according to an implementation-specific internal security model, between the two security areas.

## 3.7 QUALITY OF SERVICE

### 3.7.1 OVERVIEW

Many things can happen to messages as they travel from source to destination, resulting in the following issues:

– Dropped Messages

The messaging layer might fail to deliver (drop) some messages if they arrive when the buffers are already full. Some, none, or all of the messages might be dropped, depending on the state of the network, and it is impossible to determine what will happen in advance. The receiving component must ask for this information to be retransmitted, possibly causing severe delays in the overall transmission.

– Duplication

  Multiple instances of the same message can be received when the network is designed to adopt a multiple-path forwarding strategy. The receiver has to detect when such a condition occurs so that just the first instance of the message is taken, while other instances are discarded.

– Delay

  It might take a long time for a message to reach its destination, because it gets held up in long queues, or takes a less direct route, or is being transmitted over long distances. Alternatively, it might follow a fast, direct route. Thus delay is very unpredictable.

– Jitter

  Messages from source will reach the destination with different delays. This variation in delay is known as jitter and can seriously affect the quality of streaming audio, video and also onboard systems that rely on messaging being deterministic.

– Out-of-Order Delivery

  When a collection of related messages is routed through a network, different messages may take different routes, each resulting in a different delay. The result is that the messages arrive in a different order from the one in which they were sent.

– Error

  Sometimes messages are misdirected, or combined together, or corrupted, while en route. The receiver has to detect such errors and, just as if the message had been dropped, ask the sender to repeat itself.

– Time Coupling

  If a destination is not present on the network when a message arrives for that destination, unless the transport holds the message until the destination becomes available, that message will be dropped. This relationship is called Time Coupling. Tight Time Coupling requires that both sender and receiver be present at the same time. Loose Time Coupling allows both to be present at non-overlapping times.

Quality of Service (QoS) abilities are specific to the messaging transports employed but also to a particular implementation of an application. Different transports may overcome some of the QoS issues listed above, but in the case that one does not, then, depending on the requirements of an application, the application software or the MAL implementation itself may have to provide functionality to overcome an issue.

There may, of course, be other driving factors on the QoS requirements of an implementation, such as the cost of developing reliable messaging transports compared to the likelihood and impact of message loss. For example, with a direct RPC style connection over a LAN, most of the main QoS issues are relatively low, especially for non-critical operations or functions.

This need for varying degrees of QoS support requires that there exist more than one level of QoS provision from the MAL specification.

### 3.7.2 SUPPORTED QOS LEVELS

**3.7.2.1** The following QoS levels exist in the MO Service Concept:

a) Best Effort

- makes a single attempt to deliver a message to its destination but cannot ensure that it will be delivered successfully;
- provides messages without errors but may be duplicated;
- does not necessarily preserve the order of messages;
- does not provide Loose Time Coupling.

b) Assured

- builds on Best Effort;
- ensures delivery of messages to its destination;
- does not duplicate messages;
- preserves the order of messages between a single provider and consumer.

c) Queued

- builds on Assured;
- provides support for Loose Time Coupling.

d) Timely

- builds on Assured;
- provides time guarantees (delay and jitter within specified limits).

NOTE – The QoS levels are interaction-pattern independent; each pattern can be deployed over any QoS level.

**3.7.2.2** A compliant implementation shall support at least one of the QoS levels; however, it is not required to support all levels.

**3.7.2.3** It shall be possible from a client application to determine which levels are supported by a MAL implementation over a specific transport.

### 3.7.3 QOS PRIORITIES

A service provider can also support a number of priority levels that apply across all QoS levels it supports.

a) The provider shall respond to a higher-priority message before a lower-priority message regardless of which QoS level is in use.

b) These shall be numbered from zero (lowest) to the number of levels minus one (highest). For example, if a provider supports four priority levels, they are numbered zero to three.

### 3.7.4 QOS NEGOTIATION

A given service provider need not offer all QoS levels and in fact will probably only offer a single level, as the specifics of a QoS level often have application implementation implications.

a) The set of QoS levels a service provider supports is a factor of two items, the QoS levels that the provider application supports and the QoS levels that the chosen transport supports.

b) Service providers inform service consumers of their QoS support level and the number of priorities through the Common Directory Service (reference [1]). What QoS levels a service provider publishes in the directory will be driven by what it can support and what transports are available to it.

c) A given service provider also specifies the number of priority levels it supports, which may be only one (in which case all messages are of the same priority).

d) A service consumer selects the QoS level and priority by sending those values in the relevant fields of the message header of the initial message.

NOTE – The priority and QoS level of a connection from a consumer to a provider is fixed and cannot change during the lifetime of that link. However, this does not preclude a consumer closing the link and reopening at a different level or having multiple links open concurrently.

### 3.7.5 QOS FAILURE

A QoS level provides certain guarantees, depending on chosen level, about messages passed on that link. However, it is possible that at some point, possibly because of network issues, those constraints will be violated.

a) The sender of a message shall be informed of these error conditions using the standard error reporting mechanisms provided for in the patterns and using standard error codes defined by the MAL specification.

b) It is also possible that a specific message transport can report transport-specific errors to a message sender when errors are detected by it outside of the normal message exchange of an interaction pattern. In this case the error shall be reported using an implementation language-specific mechanism (detailed in the relevant language mapping) asynchronously.

c) It is also possible that a specific transport is able to detect the loss of connection between a provider and consumer. In this case the transport shall send an appropriate standard error code to the application using the asynchronous error reporting mechanism outlined above.

NOTE – When a protocol bridge is in use (see 2.6) there is an issue with what QoS can be provided to a consumer. The multiple transports in use when using bridges may affect the QoS offered by a provider, with both consumer and provider being affected by the link. Methods to mitigate this are currently outside this specification.

### 3.7.6 QOS PROPERTIES

For a specific connection there are certain properties that must be defined before a connection can be made. These properties are associated with the service provider and must be communicated to the consumer by some means, possibly through configuration but most likely via the Common Directory Service.

The following table provides the list of standard QoS properties; however, more may be defined or required by a particular implementation or message transport:

**Table 3-1: Standard QoS Properties**

| Property identifier | Permitted values | Default value | Description |
|---|---|---|---|
| TIME_TO_LIVE | Non-negative integer values | 0 | The time allowed by the consumer for delivering the messages to the provider. If the message cannot be delivered before then, it must be dropped and a delivery-timed-out error must be returned to the consumer, if the interaction pattern allows it. Values are in milliseconds, a value of '0' means no timeout. |
| LARGE_MESSAGE | Boolean | False | Set by the application layer when creating a message to transmit to a destination. Signals to the lower layers that the message being transmitted is considered 'large' by the application. Is expected to be used by Transports that can optimize large data transfers. The definition of 'large' is deployment specific. |

## 3.8 COMMON OBJECT MODEL

### 3.8.1 OVERVIEW

The COM provides a standard service template for MO Services to utilise. It defines an entity model that MO Services extend and an associated set of operations for the management and observation of that model. Defining this standard service entity model allows not only the specification of standard operations but also the definition of a standard historical archive and the associated services that support the archive.

### 3.8.2 COMMON MODEL STRUCTURE

**3.8.2.1** Services that utilise the COM must adhere to the basic structure that is shown in figure 3-6. This model is composed of four conceptual objects, of which three (Definition, Occurrence, and Status) are represented in structures used by the COM:

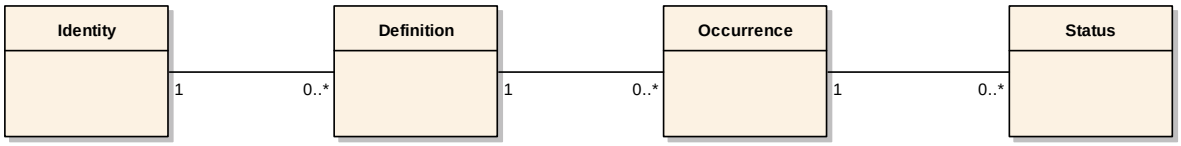


**Figure 3-6: COM Structure**

a) The first conceptual object is the identity of the entity. The entity identity exists throughout the history of the archive and cannot change. For example, for a telemetered parameter, that would be the parameter name.

b) The second conceptual object is the definition of the entity. The definition of an entity may evolve over time, so there is a one-to-many relationship between the entity identity and its definition. For example, this would be the definition of the parameter containing information such as type and description.

c) The third conceptual object is the occurrence of the entity. The occurrence object holds attributes of the entity that exist at creation. There may be many occurrences for each definition of an entity, but each occurrence can have only one definition, so there exists a one-to-many relationship between definition and occurrence. For some types of entity there may only ever be one occurrence for each definition, which is the case with telemetered parameters. This does not violate the relationship, but it must be noted during the specification of the relevant service. For other types of entity there may be many occurrences concurrently for a definition. For example, with a telecommand, each sending of that telecommand (based on a specific definition) is an occurrence of that telecommand and is distinct from other occurrences. The occurrence would contain such information as the argument values of the telecommand.

d) The final conceptual object is the status of the entity. There may be many status updates for each occurrence of an entity, but each status is related to only a single occurrence, so there exists a one-to-many relationship between occurrence and status. For example, the current value of a parameter or the current verification state of a telecommand occurrence is a status attribute. For some types of entities there may not actually be a status. For example, an alert occurrence has no alert status as the occurrence does not change over time. This does not violate the relationship, but it must be noted during the specification of the relevant service.

e) At any one point in time an entity can have zero to *n* active definitions, each definition may have zero to *n* occurrences, and each occurrence has zero to one active statuses.

**3.8.2.2** The Definition, Occurrence, and Status object are represented as abstract structures in the COM. A service specification must extend these abstract structures for it to be possible to use the COM operations. For example, for telecommands the structures used to represent the definition, any occurrences, and the evolving status of those occurrences, must extend the abstract structures defined by the COM formal specification.

### 3.8.3 COMMON OBJECT MODEL UPDATES

Changes to the model are communicated to clients by the distribution of updates. The COM defines a base update message, one for each component of the model, which all service updates must extend.

a) There exist update structures for Definition, Occurrence, and Status. These structures extend the basic MAL update structure so that they may be used with the Publish/Subscribe pattern.

b) Updates to the COM are either full updates, where a new copy of the object is sent, or partial, where just part of the object is sent.

c) The model defines three complete update messages, one for Definition, one for Occurrence, and one for Status. These shall be used to distribute a complete copy of the relevant object. For example, the definition of a new telecommand would be sent as a MAL update which would be extended by a complete Definition update message containing the new telecommand Definition structure.

d) Partial updates are specified by extending the relevant COM update structure with a service-specific structure containing the changed information. It is a service specific detail how this is defined and what each partial update actually modifies.

### 3.8.4 COMMON MODEL ARCHIVE STRUCTURE

**3.8.4.1** Services that utilise the COM must adhere to a basic structure. This structure allows the definition of a common archive structure for the provision of historic retrieval and replay services.

**3.8.4.2** As stated in 3.8.2, the Definition, Occurrence, and Status objects are represented as abstract structures in the COM. A service specification must extend these abstract structures for it to be possible to use the archive and any associated historical services. For example, for telecommands the structures used to represent the definition, any occurrences, and the evolving status of those occurrences must extend the abstract structures defined in the COM specification for any COM compliant archive to be able to hold these service-specific structures.

**3.8.4.3** As changes are made during the lifetime of the items this information is distributed to consumers using the COM update structures and the normal MAL interaction patterns. These updates can also be stored in a COM compliant archive:

**3.8.4.4** By storing these updates in an archive, any historical replay/retrieval operations can correctly reflect the history of the items.

# 4 MO ARCHITECTURE MODEL

## 4.1 OVERVIEW OF MO ARCHITECTURE MODEL

### 4.1.1 PURPOSE AND SCOPE OF ARCHITECTURE MODEL

The purpose of the Architecture Model is to provide the functional view of the layers of the MO Service Stack. These functional concepts are elaborated in the context of the MO system environment introduced in section 3.

This Architecture Model provides an abstract model of an MO system. This abstract model is refined in two ways: one to provide a functional view of a MO system, and the second to provide an interaction view of each layer.

The functional view defines the functionality that is performed by that layer, without regard to the way this functionality is implemented in real systems. The functional view decomposes the layers into elementary parts that transfer messages between the peer layers.

The interaction view models the flow of messages inside each of the layers. The interaction view also shows the error return paths and conditions for each of the layers.

### 4.1.2 MODELLING TECHNIQUE

The functional views of the architecture model are presented as UML object diagrams.

The interaction views of the architecture model are presented as UML sequence diagrams, which model the interaction of the objects.

## 4.2 TOP-LEVEL ARCHITECTURE MODEL

### 4.2.1 OVERVIEW

Figure 4-1 details the overall top-level model for the MO Service Stack. Each of the identified layers is detailed in following subsections.

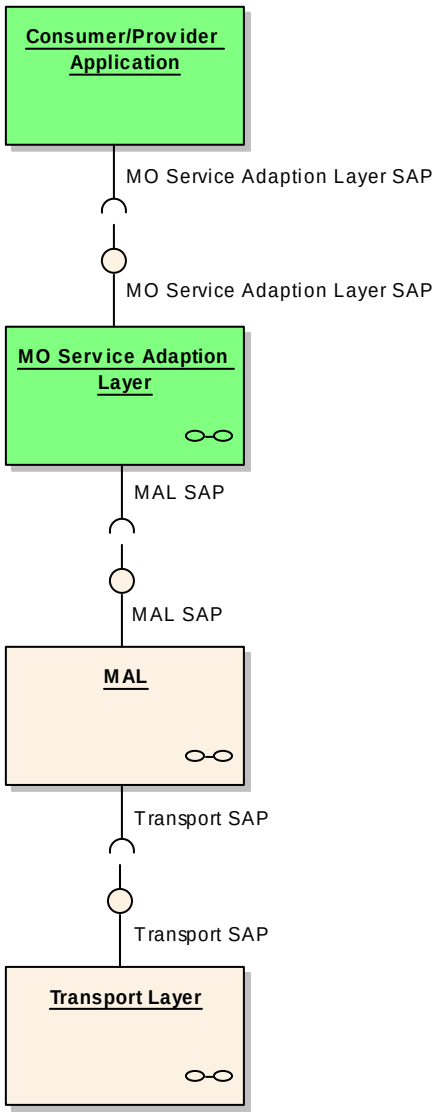


**Figure 4-1: Top-Level MO Service Architecture**

Each object represents a layer in the MO Service Stack. Each layer provides an abstract service interface that is consumed by another layer.

### 4.2.2 HIGH-LEVEL SENDING SEQUENCE

The high-level sending sequence down the stack is shown in figure 4-2.

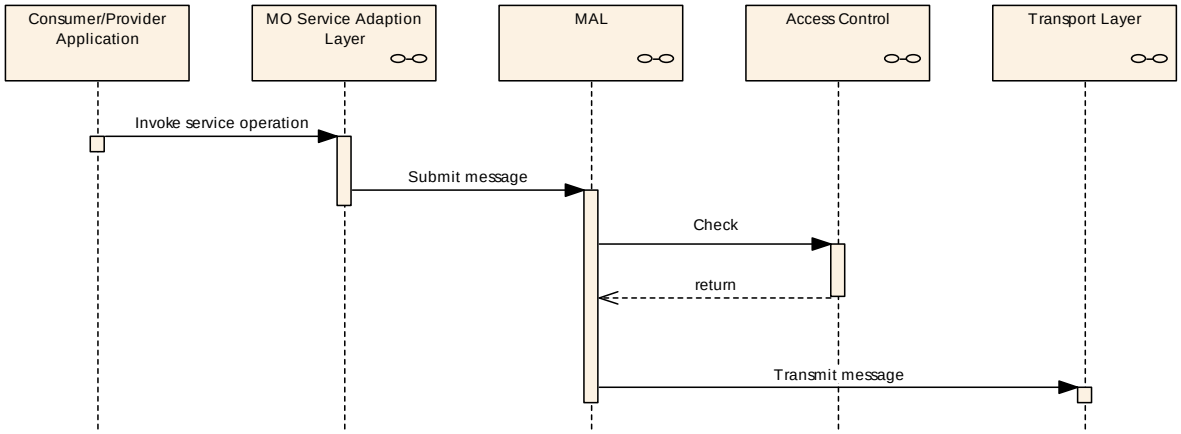


**Figure 4-2: High-Level Sending Sequence**

The sequence is as follows:

a) The Application invokes the relevant service operation provided by the MO Service Adaption Layer. This operation may be the start of a message exchange by a service consumer (do some operation) or by a service provider in response to an existing operation request (here is the result).

b) The MO Service Adaption Layer converts that operation invocation to a message and passes that to the MAL using its abstract layer interface.

c) The MAL first checks that the message may be sent using the Access Control object. The rules of the Access Control object are deployment specific, it is a deployment decision as to what must be checked before a message is sent.

d) Finally the message is passed to the Transport layer for encoding into PDUs and transmission to the destination.

### 4.2.3 HIGH-LEVEL RECEPTION SEQUENCE

The high-level reception sequence up the stack is shown in figure 4-3.

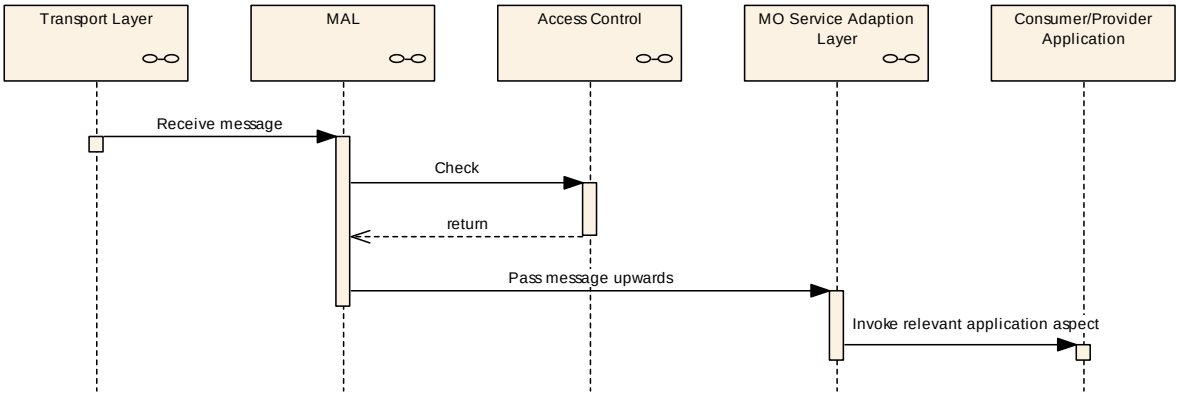


**Figure 4-3: High-Level Reception Sequence**

a) The Transport layer receives a message from another Transport Layer in another component (step not shown).

b) The message is decoded and any transport-specific checks are performed before it is passed upwards to the MAL.

c) The MAL first checks that the message may be received using the Access Control object. The rules of the Access Control object are deployment specific; it is a deployment decision as to what must be checked before a message is received.

d) The MAL then passes the message up to the MO Service Adaption Layer.

e) The MO Service Adaption Layer converts that message into a relevant operation invocation to the Application (consumer or provider).

## 4.3 MO SERVICE ADAPTION LAYER ARCHITECTURE

### 4.3.1 GENERAL

The MO Service Layer is tasked with converting from the service-specific operation-based interface to the message interface of the MAL. Logically it contains both the Consumed Service Interface used by a consumer application and the Provided Service Interface used by the provider application:

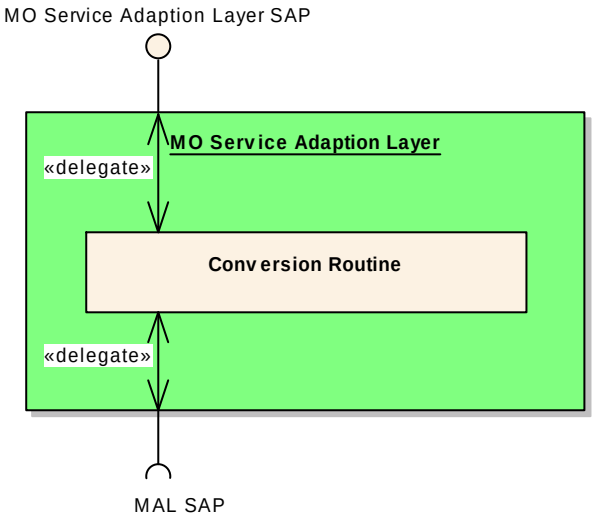


**Figure 4-4: MO Service Adaption Layer Architecture**

a) The layer presents to the Application layer a service-specific interface. This is expected to be represented as a language-specific API.

b) It uses the abstract layer interface of the MAL to send messages and receive messages.

NOTES

1 The COM is not shown here because the COM is a service template and is extended by service specification.

2 Once the service specifications are cast into a programming language representation the COM is just part of the normal service API.

3 The behaviour of this layer is the same regardless of whether a service is defined in terms of the COM or not.

### 4.3.2 MESSAGE SENDING SEQUENCE

Figure 4-5 shows the message sending sequence for the MO Service Adaption Layer. It includes the error case as an alternative fragment.

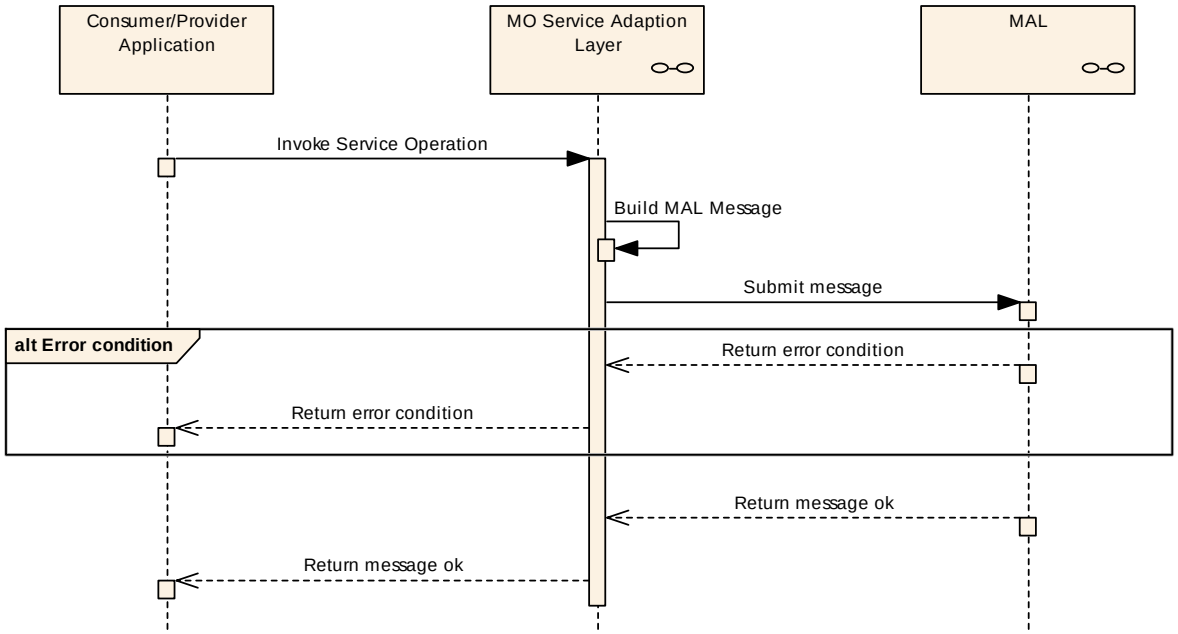


**Figure 4-5: MO Service Adaption Layer Sending Sequence**

a) The Application layer invokes a service-specific operation on the MO Service Adaption Layer. This may be an initial message from a consumer starting an interaction or from a provider in response to a received message.

b) The MO Service Adaption Layer constructs a message that represents the service-specific operation.

c) The MO Service Adaption Layer then submits the message to MAL using its abstract interface.

d) The MAL either returns an error condition due to a failure to send the message or a confirmation that the message was sent successfully.

NOTES

1 This does not imply that the sent message was received or accepted by the destination but merely that the layers below the MO Service Layer accepted the message.

2 Depending on the interaction pattern being used there may not be any further messages received for that operation from the destination.

3 The error condition applies to the SEND interaction pattern for the case where errors are raised by the MAL or Transport layer itself.

### 4.3.3 MESSAGE RECEPTION SEQUENCE

Figure 4-6 shows the message reception sequence for the MO Service Adaption Layer. It includes the error case as an alternative fragment.

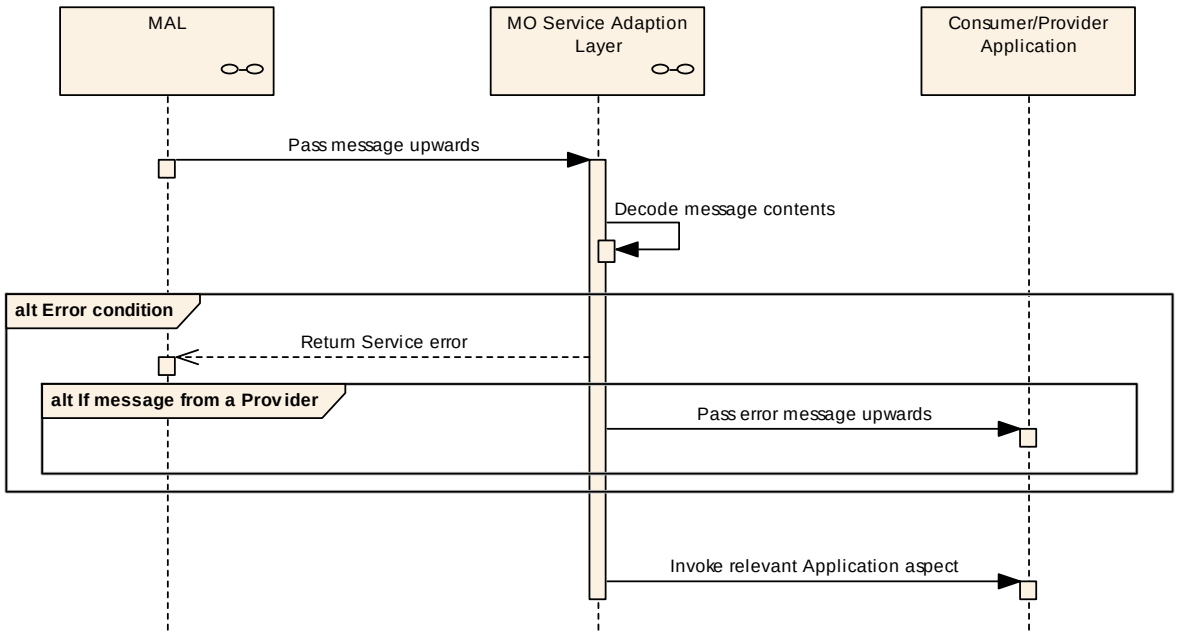


**Figure 4-6: MO Service Adaption Layer Reception Sequence**

a) The MO Service Adaption Layer receives the message from the MAL.

b) The MO Service Adaption Layer decodes the contents of the message into its internal representation to map to the relevant Application layer.

c) If there is an error during the decoding the MO Service Adaption Layer returns the relevant error to the MAL. The MO Service Adaption Layer returns the error by using the relevant interaction pattern error message.

d) If the pattern being used does not support error messages (for example SEND pattern), then the MO Service Adaption Layer has no way of returning the error and shall just drop the message.

e) If the message is from a provider (determined by examining the interaction pattern stage fields of the message), then the error message is also sent to the transaction source (usually the consumer).

f) If there was an error, then the sequence ends at this point.

g) If no errors are detected, then the MO Service Adaption Layer invokes the relevant Application layer.

## 4.4 MESSAGE ABSTRACTION LAYER ARCHITECTURE

### 4.4.1 GENERAL

**4.4.1.1** The MAL forms the central layer of the MO stack. It is responsible for coordinating the flow of messages between a consumer and provider, and also provides the conceptual interface layer between the application service based view and the physical transport based view:

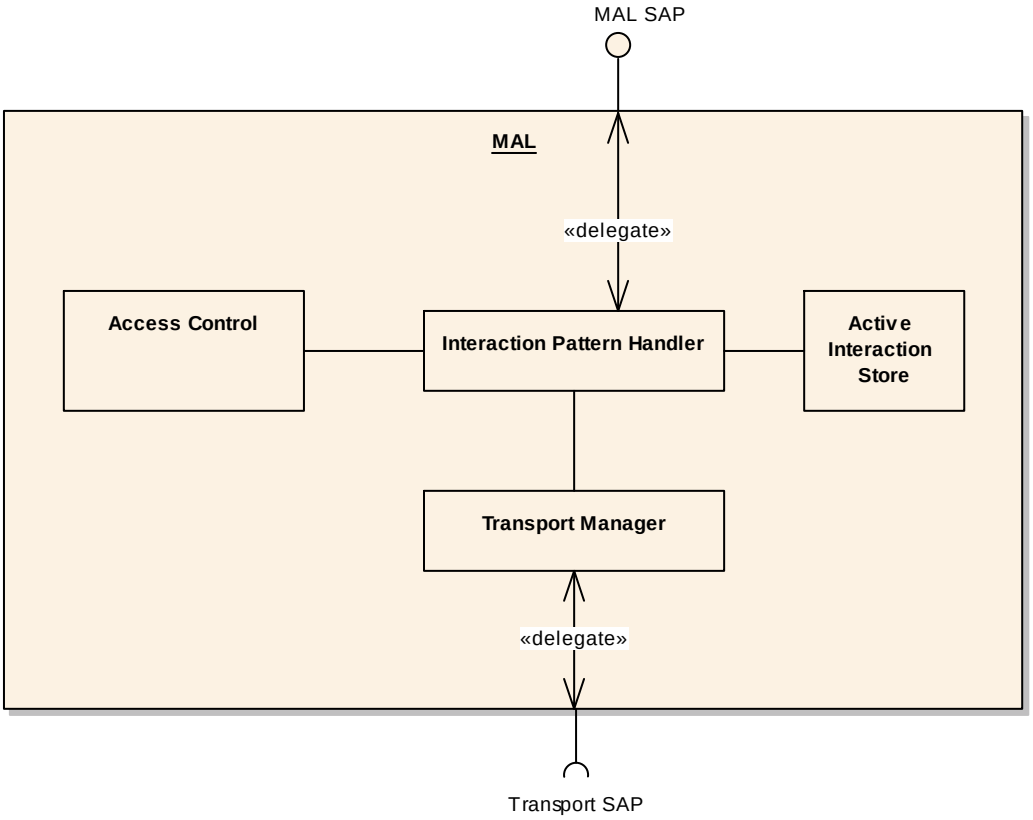


**Figure 4-7: MAL Architecture**

**4.4.1.2** There are several parts to the MAL that are shown in the above figure. It is not intended that these will be actual parts of an implementation; however, it is expected that they will be represented in behaviour.

a) The MAL Interaction Pattern Handler is responsible for managing the flow of messages between its internal parts, the Access Control part, and between the connected MO stack layers.

b) The Active Interaction Store is used by the MAL for recording which interactions are currently active so that response messages can be routed to the correct upper layer.

c) The Transport Manager allows there to be separation between the message handling components of the MAL (detailed above) and the Transport layer below. It is only something that is required if a MAL implementation is capable of using multiple transports concurrently.

### 4.4.2 MESSAGE SENDING SEQUENCE

Figure 4-8 shows the message sending sequence for the MAL. It includes the error cases as alternative fragments.

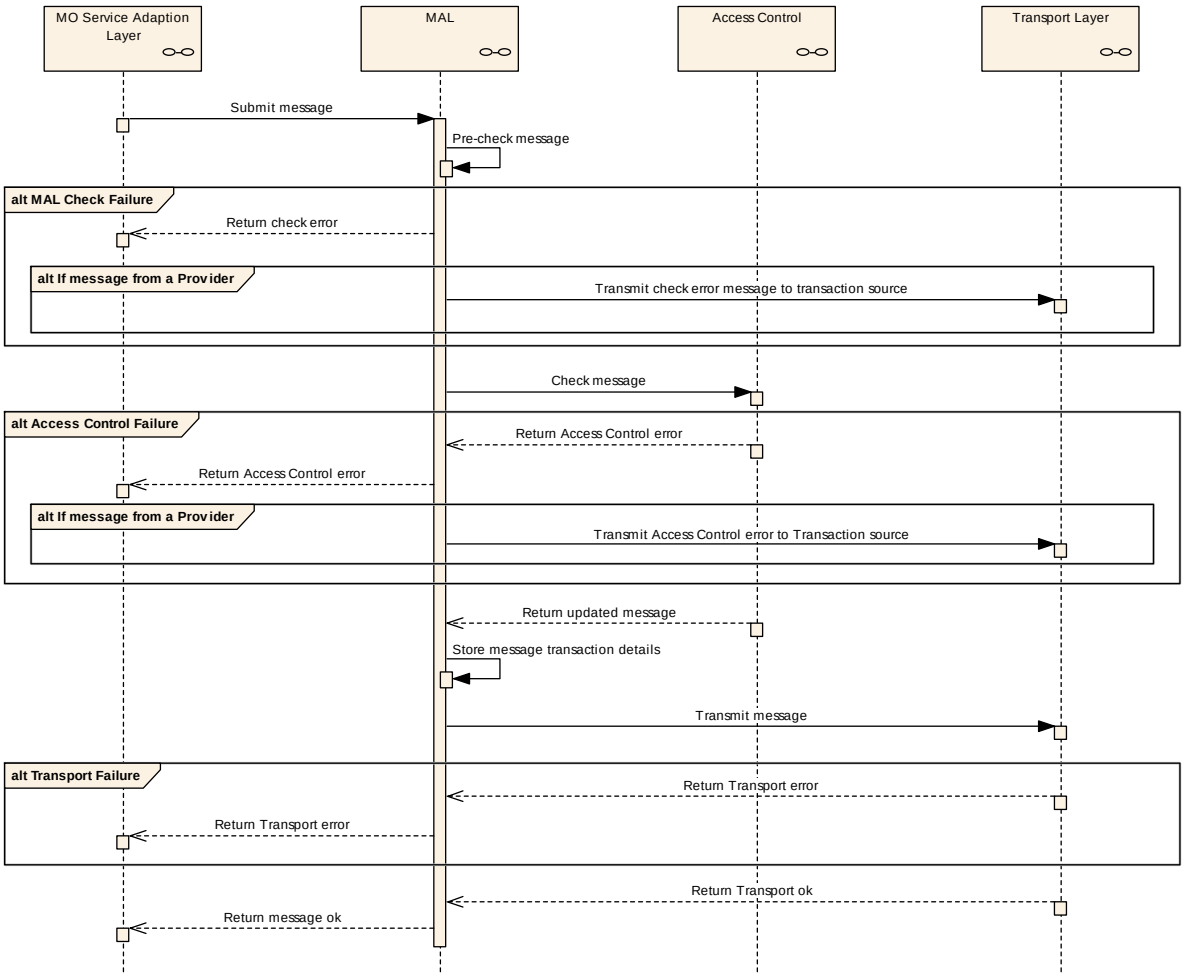


**Figure 4-8: MAL Sending Sequence**

The basic outline of this sequence from the high-level sequence (figure 4-2) is that the MAL receives message from the MO Service Adaption Layer and passes them down to the relevant Transport layer.

a) The MO Service Adaption Layer submits the message to the MAL using the MAL's abstract interface.

b) The MAL checks the contents of that message to ensure that it contains all the required information for the MAL itself. If it fails these checks, then an error is returned to the MO Service Adaption Layer.

c) If the message is from a provider (determined by examining the interaction pattern stage fields of the message), then the error message is also sent to the transaction source. The transaction details are removed from the Transaction Store.

d) If there was an error, then the sequence ends at this point.

e) The MAL then submits the message to the Access Control part using its abstract interface. The rules that the Access Control part applies are deployment specific.

f) If the Access Control part rejects the message, then it returns an error to the MAL. The MAL creates the relevant Access Control error message and passes this back to the MO Service Adaption Layer.

g) If the message is from a provider (determined by examining the interaction pattern stage fields of the message), then the error message is also sent to the transaction source. The transaction details are removed from the Transaction Store.

h) If there was an error, then the sequence ends at this point.

i) If the Access Control part accepts the message, then it returns an updated (with updated security credentials, if applicable) message to the MAL.

j) The MAL then stores the transaction details if this is the first message being sent from the consumer. If it is a return message from a provider, then the transaction details are removed from the store if it is the final message for that interaction pattern.

k) The MAL then passes the message to the Transport layer for transmission to the destination.

l) If the Transport encounters an error during its attempt to send the message, or part of the message, then it returns an error to the MAL. If the sender is a consumer, then the MAL uses the Active Transaction Store to build a return error message to be returned to the sending application. If it is a provider, then an application error is returned to the provider.

m) If there was an error, then the sequence ends at this point.

n) If there are no issues during sending by the Transport it returns a success message to the MAL which passes that back to the MO Service Adaption Layer.

### 4.4.3 MESSAGE RECEPTION SEQUENCE

Figure 4-9 shows the message reception sequence for the MAL. It includes the error cases as alternative fragments.

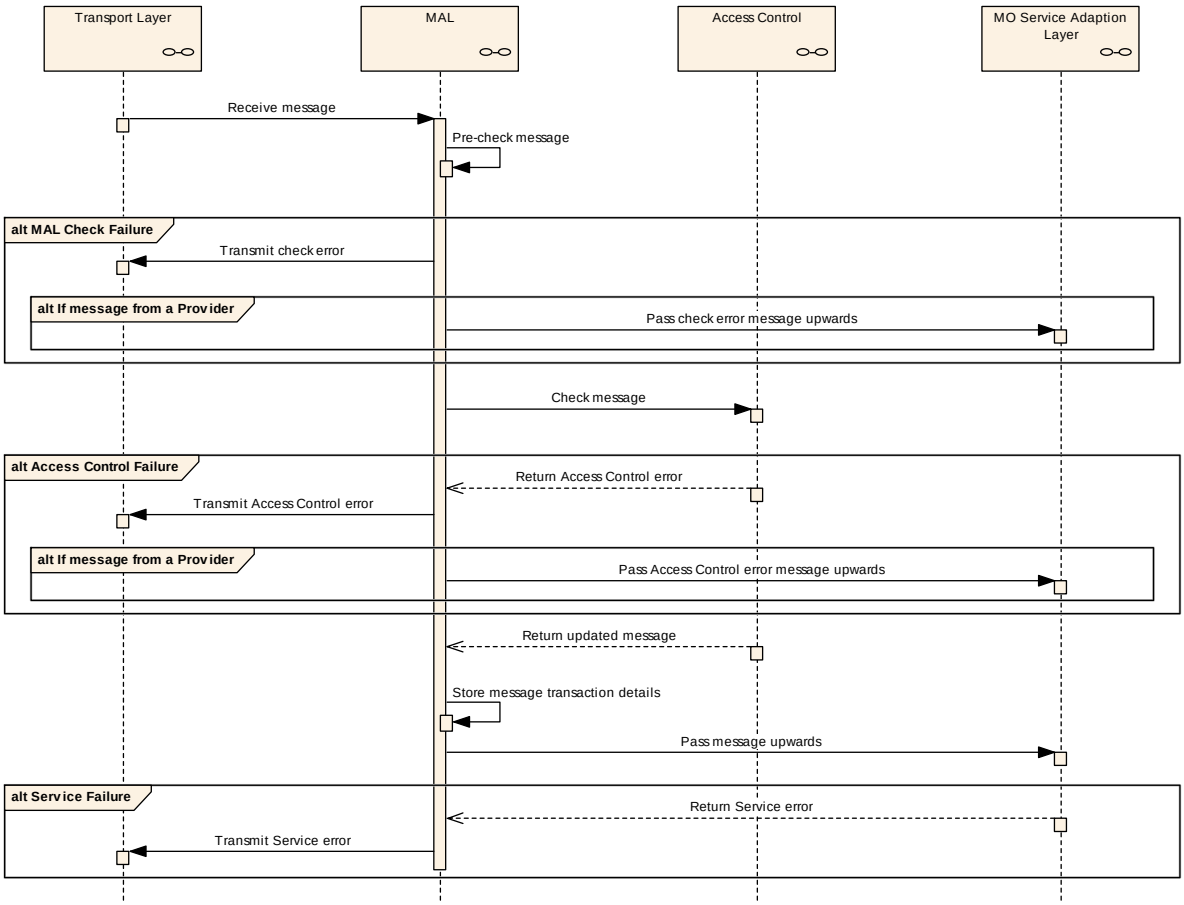


**Figure 4-9: MAL Reception Sequence**

The basic outline of this sequence from the high-level sequence (figure 4-3) is that the MAL receives messages from the Transport layer and passes them up to the relevant MO Service Adaption Layer.

a) The Transport layer passes the message to the MAL using the MAL's abstract Transport interface.

b) The MAL checks the contents of that message to ensure that it contains all the required information for the MAL itself. If it fails these checks, then an error is transmitted to the Transport layer unless the message is itself an error message.

c) If the message fails the check and is from a provider (determined by examining the interaction pattern stage fields of the message), then an error message is sent to the transaction source. The transaction details are removed from the Transaction Store.

d) If there was an error, then the sequence ends at this point.

e) The MAL then submits the message to the Access Control part using its abstract interface. The rules that the Access Control part applies are deployment specific.

f) If the Access Control part rejects the message, then it returns an error to the MAL. If the message is not an error, then the MAL creates the relevant Access Control error message and passes this to the Transport layer for transmission.

g) If the message is from a provider (determined by examining the interaction pattern stage fields of the message), then the Access Control error message is also sent to the transaction source. The transaction details are removed from the Transaction Store.

h) If there was an error, then the sequence ends at this point.

i) If the Access Control part accepts the message, then it returns the message (with updated security credentials, if applicable) to the MAL.

j) The MAL then stores the transaction details if this is the first message being sent from the consumer. If it is a return message from a provider, then the transaction details are removed from the store if it is the final message for that interaction pattern.

k) The MAL then submits the message to the relevant MO Service Adaption Layer component.

l) If the higher layer encounters an error during its processing off the message, then it returns an error to the MAL. If the sender of the message is a consumer and the sent message is not an error, then the MAL uses the Active Transaction Store to build a return error message to be transmitted to the sending application. If it is a provider, then an application error is returned to the provider. The sequence then ends.

### 4.4.4 ACCESS CONTROL MESSAGE PROCESSING SEQUENCE

The Access Control is conceptually part of the MAL and is used by the MAL to provide Authentication and Authorisation checks on a message. However, it provides an important facility and therefore it is expanded here.

The implementation rules and checks made by Access Control are completely deployment specific; however, the interface used by the MAL and the behaviour of the components in regards to that interface are part of the MAL standard (reference [2]).

Figure 4-10 shows the message processing sequence for the Access Control part of the MAL. Only a single sequence is presented for this as it is identical regardless of whether a message is being sent or received. It includes the error cases as alternative fragments.

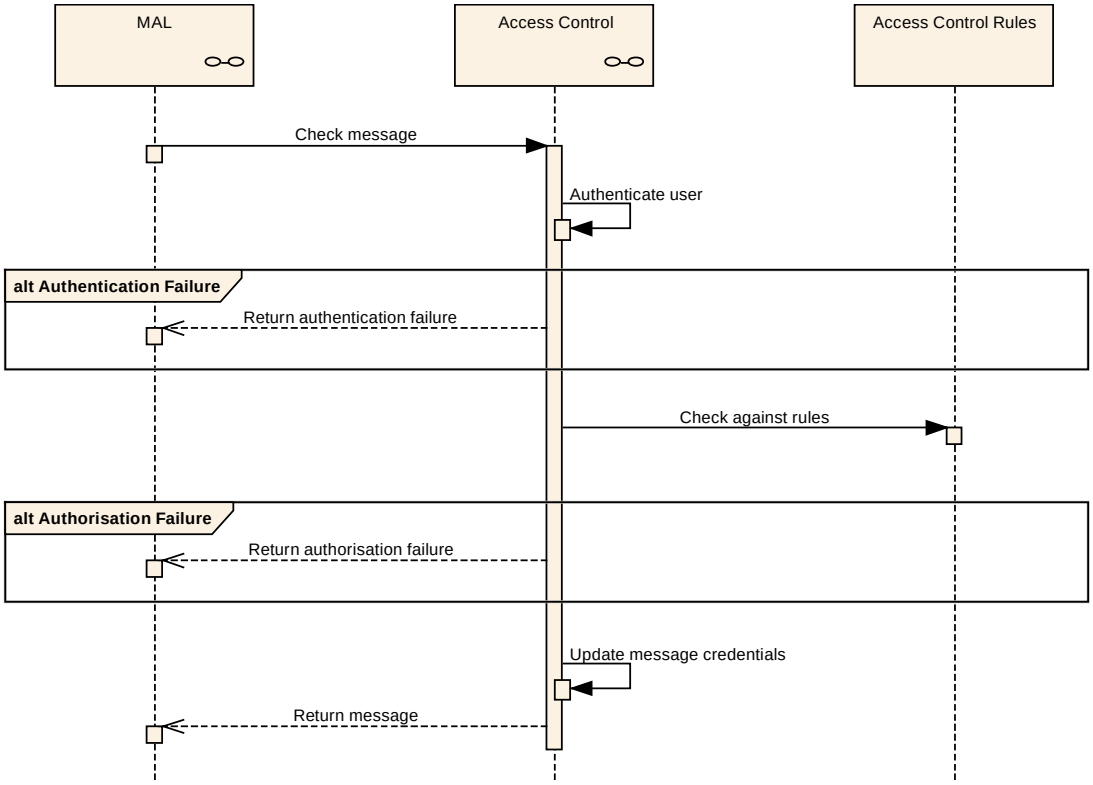


**Figure 4-10: Access Control Processing Sequence**

a) The MAL submits a message to the Access Control part for checking. It includes whether the message is being received or sent by the MAL.

b) The first check of the Access Control part is to authenticate the user credentials of the message.

c) If this check fails the Access Control part shall return an Authentication failure message to the MAL and the sequence shall end at this point.

d) The Access Control part shall then check that the message is authorised to be sent/received. The actual checks involved at this point are deployment specific.

e) If this check fails the Access Control part shall return an Authorisation failure message to the MAL and the sequence shall end at this point.

f) If both authentication and authorisation checks pass, then the Access Control part shall update the message user credentials (if appropriate) and return the updated message to the MAL.

## 4.5 TRANSPORT LAYER ARCHITECTURE

The Transport layer is responsible for taking the abstract message from the MAL and transmitting it to the destination:

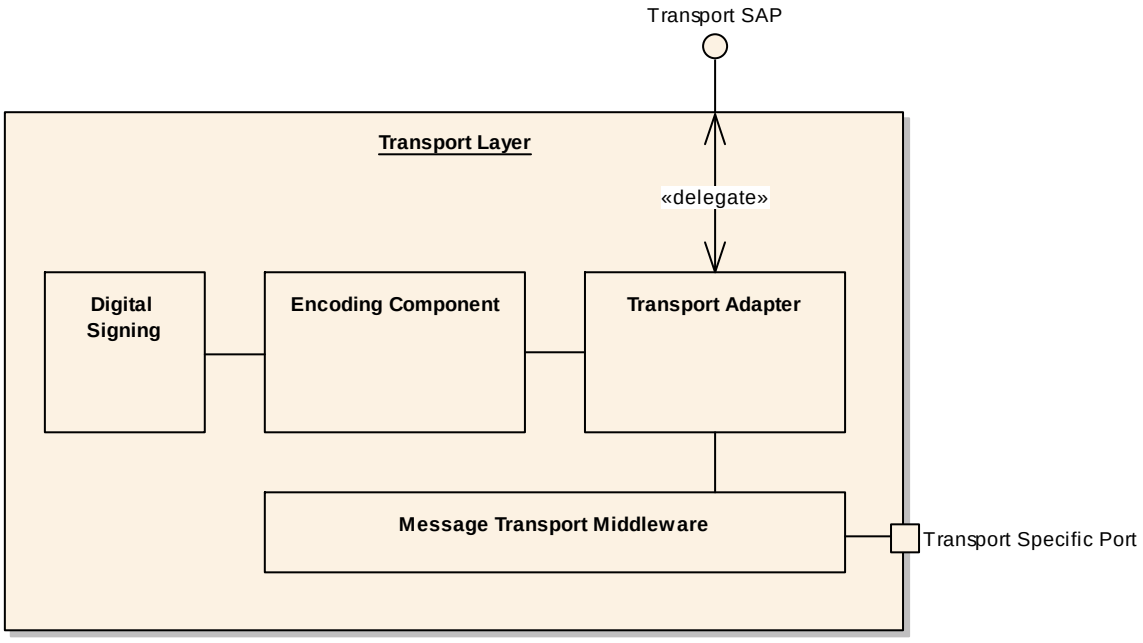


**Figure 4-11: Transport Layer Architecture**

There are several parts to the layer that are shown in the above figure. It is not intended that these be actual parts of an implementation; however, it is expected that they will be represented in behaviour.

a) The Transport Adapter is responsible for managing the flow of messages between the internal components of the Transport component.

b) The Encoding Component is responsible for the conversion from the abstract message format of the MAL into the on-the-wire representation used by that transport. It has an association with the digital signing part if this is applicable for the security deployment in use.

c) The digital signing part may not actually reside in the layer but is shown like this because the two functions are related.

d) The Message Transport Middleware is the component that is responsible for the actual transmission of the encoded message to the destination.

NOTES

1 Splitting of an abstract message into multiple fragments is a Transport layer issue and is not shown in this document.

2 If multiple separate technologies are to be used by the Transport layer to transmit the fragments, then the coordination of these separate technologies is an implementation detail and is not shown here.

3 Recombination of the message fragments is a responsibility of the Transport layer.

### 4.5.1 MESSAGE SENDING SEQUENCE

Figure 4-12 shows the message sending sequence for the Transport layer. It includes the error cases as alternative fragments.

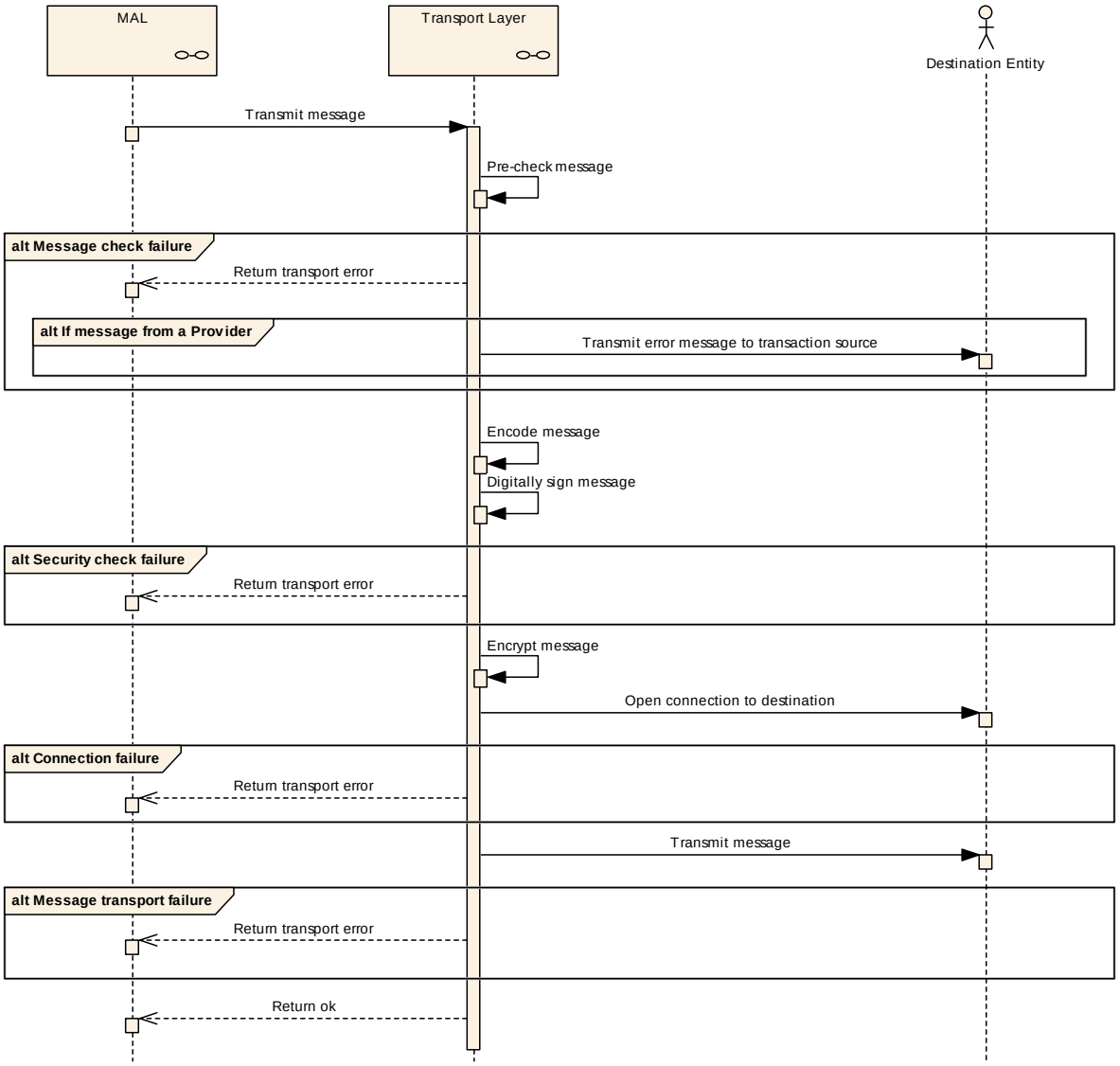


**Figure 4-12: Transport Layer Sending Sequence**

The basic outline of this sequence from the high-level sequence (figure 4-2) is that the Transport layer receives messages from the MAL and passes them to the relevant destination.

a) The MAL submits the message to the Transport layer using the layer's abstract interface.

b) The Transport layer checks the contents of that message to ensure that it contains all the required information. If it fails these checks, then an error is returned to the MAL.

c) If the message is from a provider (determined by examining the interaction pattern stage fields of the message), then the error message is also transmitted to the transaction source.

d) If there was an error, then the sequence ends at this point.

e) The Transport layer then encodes the message and digitally signs the encoded message, if applicable. The need for a digital signature and the method used is deployment specific.

f) If the digital signing part rejects the message, then it returns an error using an implementation specific mechanism. The Transport layer creates the relevant Access Control error message and passes this back to the MAL as a transport failure message.

g) There is no need to send the error to the transaction source if the message is from a provider, as the signing process has failed, and therefore it would automatically be rejected by the destination transport.

h) If there was an error, then the sequence ends at this point.

i) The Transport layer then opens a connection and transmits the encoded message to the destination. The actual process at this point is transport-specific.

j) If the Transport encounters an error during its attempt to transmit the message, then it returns an error to the MAL. The sequence then ends.

k) If there are no issues during sending by the Transport it returns a success message to the MAL.

### 4.5.2 MESSAGE RECEPTION SEQUENCE

Figure 4-13 shows the message reception sequence for the Transport layer. It includes the error cases as alternative fragments.

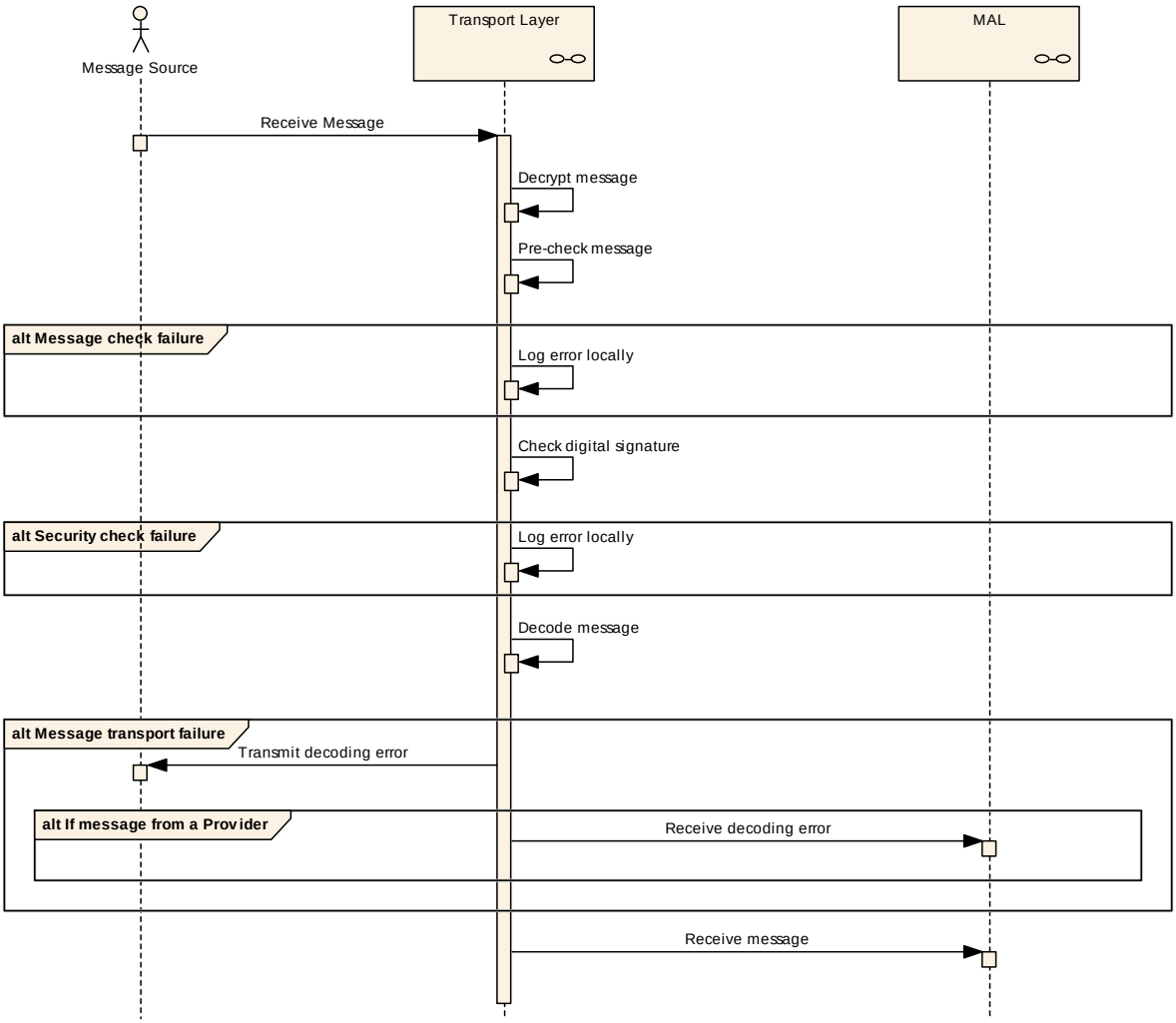


**Figure 4-13: Transport Layer Reception Sequence**

The basic outline of this sequence from the high-level sequence (figure 4-3) is that the Transport layer receives a message and passes it up to the MAL.

a) The message source passes the message to the Transport layer using whatever mechanism is appropriate for that messaging technology.

b) The Transport layer decrypts the message and then checks to ensure that it contains all the required information for the layer itself. If it fails these checks, then an error may be logged locally. No error message is passed upwards or returned to the source, because it may be a spoof message.

c) If there was an error, then the sequence ends at this point.

d) The layer then checks the digital signature, if applicable. If the digital signature part rejects the message, then it returns an error using an implementation-specific mechanism to the Transport layer. The Transport layer then may log that error locally. No error message is passed upwards or returned to the source, because it may be a spoof message.

e) If there was an error, then the sequence ends at this point.

f) The message is then decoded from the on-the-wire representation to the abstract MAL representation.

g) If there is a problem in the decoding of the message, then the Transport layer creates the relevant decoding error message and transmits this back to the message source.

h) If the message is from a provider (determined by examining the interaction pattern stage fields of the message), then the error message is also passed to the transaction source.

i) If there was an error, then the sequence ends at this point.

j) If there are no issues, it passes the decoded message to the MAL.

# 5 MO SERVICE INTERACTIONS

## 5.1 OVERVIEW

This section provides sequences for the basic support interactions required for a compliant MO service implementation.

## 5.2 SECURITY AND LOGIN

Figure 5-1 shows the basic procedure for a service consumer for interacting with a security service.

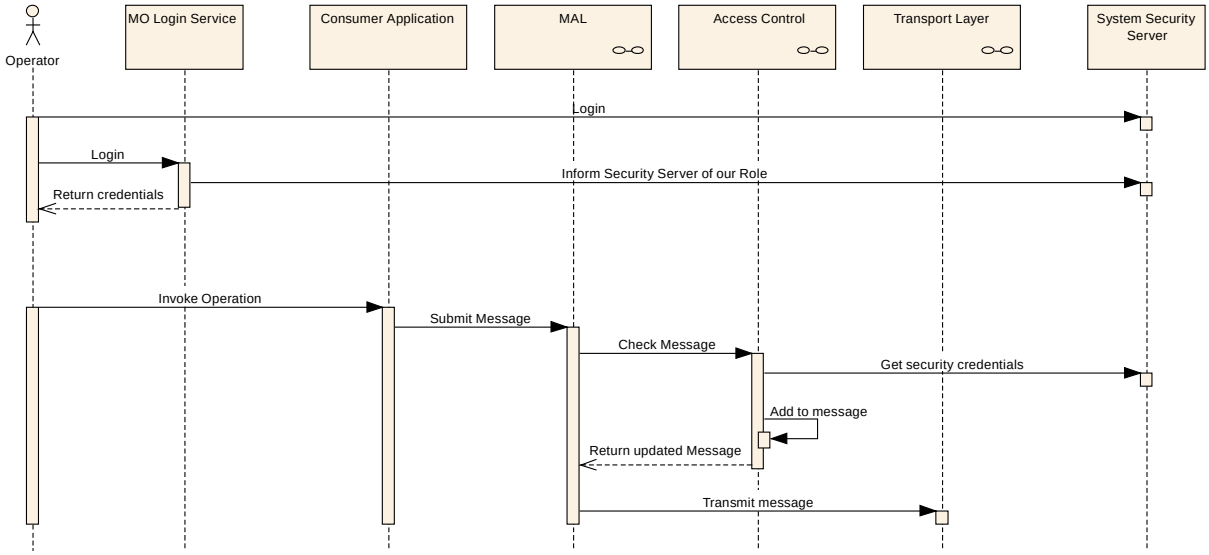


**Figure 5-1: Login Sequence**

The basic outline of this sequence is that the operator logs into the relevant operating system as normal and then logs into the MO service system to provide a relevant role:

a) The operator logs into the operating system as normal using the relevant technology and account credentials.

b) The operator then uses a deployment specific MO Login service implementation to inform the system of his or her role.

c) The MO Login service performs the actions necessary for that specific security implementation.

d) The MO Login service returns the security credentials appropriate for that implementation and deployment. These should be used for all further messages submitted to the MAL implementation.

e) It is deployment-specific what is contained in the returned security credentials and whether it is required that these then be used in other messages passed by a consumer to its MAL implementation. For example, it is possible that a specific implementation is able to determine the correct credentials from the system without any further information.

f) The operator then uses a consumer application to invoke a MO service operation. This creates and submits a message to its MAL implementation.

g) The MAL implementation submits the message to its Access Control implementation. The Access Control implementation is specific to the security implementation and deployment in operation.

h) The Access Control implementation then updates the contained security credentials, if appropriate, before returning the updated message to the MAL.

## 5.3 SECURITY CHALLENGE

To improve system security it may be required that for a specific security deployment the system shall be able to challenge the operator to re-enter his or her security credentials. The actual challenge process is outside of the scope of the MO service concept; however, as shown in figure 5-2, it is something that can be supported in the messaging patterns.

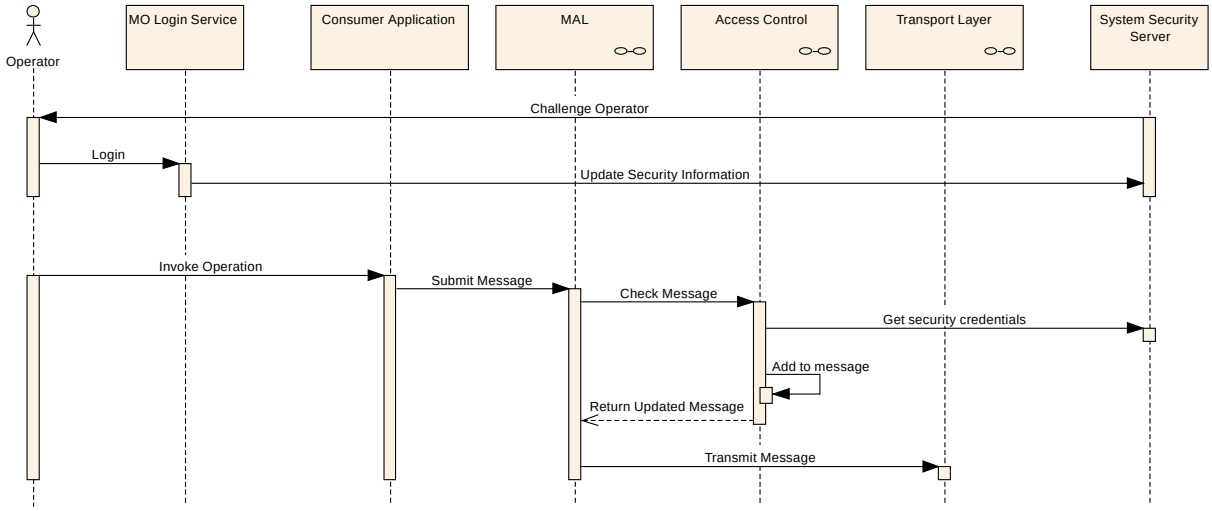


**Figure 5-2: Security Challenge Sequence**

The basic outline of this sequence is that the operator is challenged to revalidate his or her credentials by the security system deployed. These updated credentials are then used by the MO service system in future messages:

a) The operator is challenged by the security system to re-enter his or her security credentials. This is a deployment specific operation.

b) In this example the MO Login service is used to re-enter the operator's security credentials.

c) In the message exchange sequence, where the Access Control implementation updates the message credentials, the new credentials are used from this point in time onwards.

d) It is an implementation and deployment detail how the new credentials are passed to the Access Control implementation in use at that time.

## 5.4 INITIAL COMMUNICATION

For a consumer application to send a message to a provider, there needs to be a communications link between the two. How this link is opened and maintained is transport dependent.

Figure 5-3 details the sequence for the MAL and above layers.

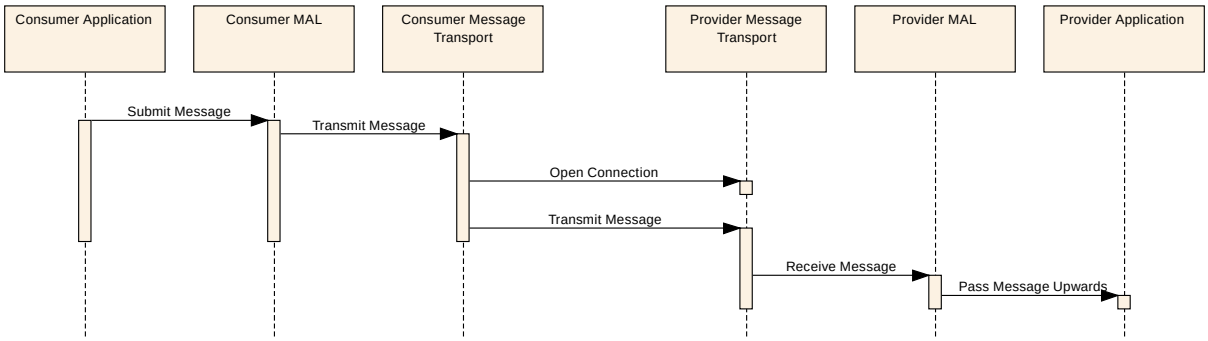


**Figure 5-3: Initial Communications Sequence**

The basic outline of this sequence is that the consumer application sends an initial message to the provider:

a) The consumer application submits a message to the MAL for transmission to the provider. This message is the first message of the required interaction pattern and operation. There are no special messages for the opening and negotiation of communications between a consumer and provider.

b) The MAL updates the message as appropriate and passes it to its message transport.

c) The Message Transport performs the required functionality for that particular transport to pass the message to the provider application.

d) The provider transport passes the message upwards to its MAL implementation. This is the first message to have been received by this provider from this consumer.

e) The Provider MAL performs the required processing of that message (such as access control) and passes the message upwards.

f) The Provider Application processes the message appropriately.

From the above it can be seen that no special initial messages are defined or sent; communications are opened using the initial message sent by a consumer. This does not mean that a specific service cannot require that a particular operation must be invoked before another, as that is a service-specific behaviour; however, the MO service concept uses the initial message exchange to open a communications link.

# ANNEX A

# DEFINITION OF ACRONYMS

# (INFORMATIVE)

| | |
|---|---|
| **AMS** | CCSDS Asynchronous Message Service |
| **API** | Application Programming Interface |
| **ASCII** | American Standard Code for Information Interchange |
| **BLOB** | Binary Large OBject |
| **LAN** | Local Area Network |
| **MAL** | Message Abstract Layer |
| **MCS** | Mission Control System |
| **MO** | Mission Operations |
| **QoS** | Quality of Service |
| **RPC** | Remote Procedure Call |
| **SDU** | Service Data Unit |
| **SM&C** | CCSDS Spacecraft Monitor & Control |
| **XML** | eXtensible Markup Language |
| **URI** | Universal Resource Identifier |

# ANNEX B

# INFORMATIVE REFERENCES

# (INFORMATIVE)

[B1] *Unified Modeling Language (UML)*. Version 2.2. Needham, Massachusetts: Object Management Group, February 2009.

[B2] *Mission Operations Services Concept*. Report Concerning Space Data System Standards, CCSDS 520.0-G-3. Green Book. Issue 3. Washington, D.C.: CCSDS, [forthcoming].

[B3] *Space Link Extension—Return All Frames Service Specification*. Recommendation for Space Data System Standards, CCSDS 911.1-B-3. Blue Book. Issue 3. Washington, D.C.: CCSDS, January 2010.

[B4] *Space Link Extension—Forward CLTU Service Specification*. Recommendation for Space Data System Standards, CCSDS 912.1-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, December 2004.

[B5] *CCSDS File Delivery Protocol (CFDP)*. Recommendation for Space Data System Standards, CCSDS 727.0-B-4. Blue Book. Issue 4. Washington, D.C.: CCSDS, January 2007.

[B6] *Asynchronous Message Service*. Recommendation for Space Data System Standards, CCSDS 735.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, [forthcoming].

[B7] *Space Packet Protocol*. Recommendation for Space Data System Standards, CCSDS 133.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

[B8] *Orbit Data Messages*. Recommendation for Space Data System Standards, CCSDS 502.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 2009.

[B9] *Tracking Data Message*. Recommendation for Space Data System Standards, CCSDS 503.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 2007.

[B10] *Attitude Data Messages*. Recommendation for Space Data System Standards, CCSDS 504.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 2008.

[B11] *Spacecraft Onboard Interface Services—Subnetwork Packet Service*. Recommendation for Space Data System Standards, CCSDS 851.0-M-1. Magenta Book. Issue 1. Washington, D.C.: CCSDS, December 2009.

NOTE – Normative references are listed in 1.8.