The Consultative Committee for Space Data Systems

Draft Recommendation for Space Data System Practices

MISSION OPERATIONS REFERENCE MODEL

PROPOSED DRAFT RECOMMENDED PRACTICE

CCSDS 520.1-P-1.1

PROPOSED PINK BOOK

May 2026

The Consultative Committee for Space Data Systems

Draft Recommendation for Space Data System Practices

MISSION OPERATIONS REFERENCE MODEL

DRAFT RECOMMENDED PRACTICE

CCSDS 520.1-P-1.1

PINK BOOK

May 2026

AUTHORITY

Issue:	Proposed Draft Recommended Practice, Issue 1.1
Date:	May 2026
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:

https://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.
- State Space Corporation (Roscosmos)/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
CCSDS 520.1-P-1.1	Mission Operations Reference Model, Recommended Practice, Issue 1.1	April 2026	Current issue -Updates content and new notes in sections 2–3 -Updates to diagrams in sections 2–4, 5-2 -3.5.3 Service Addressing completely revised -3.8 MO Object Model completely revised -The description of the MAL Architecture has been updated and expanded

CONTENTS

Figure

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 the development of MO service specifications both for CCSDS Recommended Standards and for functional MO services developed outside of CCSDS for use within an MO system or for bridging to an MO system. It serves as a reference to maintain the consistency of these Recommended Standards and services.

1.2 SCOPE

The scope of this Recommended Practice is the definition of all concepts and terms that establish a common basis for the development of MO service specifications for CCSDS Recommended Standards. It may be referenced by MO services developed outside of CCSDS.

1.3 APPLICABILITY

1.3.1 APPLICABILITY OF THIS RECOMMENDED PRACTICE

This Recommended Practice serves as a guideline for the development of compatible services for MO systems, applicable to CCSDS standard MO services and custom MO services. It is considered normative on Blue Book specifications aiming to be MO compliant. 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. MO services may be implemented on the ground, on-board a spacecraft, or both.

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 and data handled by an 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 provides sequences for the basic support interactions required for a compliant MO service implementation.

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 can be made available for use outside the component through Provided Service Interfaces.

NOTE – Components may also depend on services provided by other components through Consumed Service Interfaces.

Hardware Component: A complex physical entity (such as a spacecraft or a control system) or an individual physical entity of a system (such as an instrument, a computer, or a piece of communications equipment).

NOTE – 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.

NOTE – 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 Application Program Interface (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 (SDUs) 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.

Domain: A means of applying hierarchical taxonomy to mission operation related data and messages. The term domain is used with different meanings in specific mission and deployment scenarios in order to be able to distinguish between data and messages from different sources. For instance, in a constellation, the domain may be used to distinguish the data from individual spacecraft of the constellation. In a different context the domain may be used to distinguish between different applications, subsystems, and payloads.

MO Object: Representation of a complex data type with two specific characteristics: MO Objects have a unique and an immutable identity and can be referenced unambiguously. MO Objects are used to represent and formally specify the complex data model in service specifications.

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. A component may be a provider of some Services and a consumer of others.

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.

Interface Binding Signature (binding): A 'signature' that results from a service user and service provider implementing the proper stack of interface protocols in order to bind to service elements. This signature may involve Transmission Control Protocol/Internet Protocol (TCP/IP), Hypertext Transfer Protocol (HTTP), or some set of CCSDS space-communication protocols.

NOTE – Bindings may be 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 Capability Set: A grouping of the service operations that remains sensible and coherent, and also provides a Service Provider with an ability to communicate to a consumer its capability. The specification of services is based on the expectation that different deployments require different levels of complexity and functionality from a service. To this end, a given service may be implementable at one of several distinct levels, corresponding to the inclusion of one or more capability sets.

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.

Authentication: The process of verifying the identity or other attributes claimed by or assumed of an entity (user, process, or device), or to verify the source and integrity of data. (Reference [5].)

NOTE – This definition does not include the protection against duplication or modification of data to avoid confusion between data authentication and integrity.

Data integrity: The property that data has not been changed, destroyed, or lost in an unauthorized manner. (Reference [5].)

Data origin authentication: The corroboration that the source of data received is as claimed. (Reference [5].)

NOTE – Data origin authentication is data integrity with authentication of the source.

Authorisation: The granting of rights, which includes the granting of access based on access rights. (Reference [5].)

Confidentiality: The property that information is not made available or disclosed to unauthorized individuals, entities, or processes. (Reference [5].)

Non-Repudiation: Assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the information. (Reference [5].)

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 NOMENCLATURE

1.7.2.1 Normative Text

The following conventions apply for the normative specifications in this Recommended Practice:

a) the words 'shall' and 'must' imply a binding and verifiable specification;

b) the word 'should' implies an optional, but desirable, specification;

c) the word 'may' implies an optional specification;

d) the words 'is', 'are', and 'will' imply statements of fact.

NOTE – These conventions do not imply constraints on diction in text that is clearly informative in nature.

1.7.2.2 Informative Text

In the normative sections of this document, informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

– Overview;

– Background;

– Rationale;

– Discussion.

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 [1]). 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

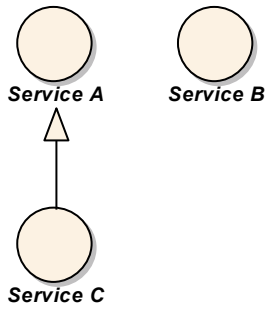


Figure 1-1: Service Extension Drawing Convention

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

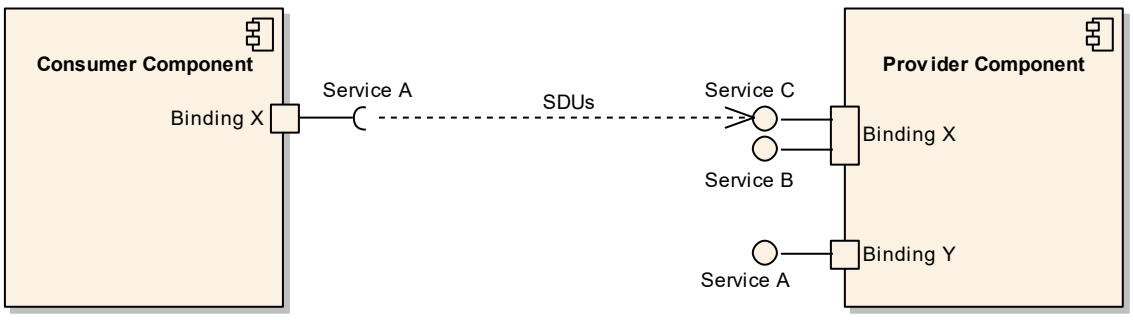


Figure 1-2: Component Drawing Convention

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.
- Interface Binding Signature is the stack of protocols that define an interface.
- Consumer and Provider interfaces and Binding Signatures 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

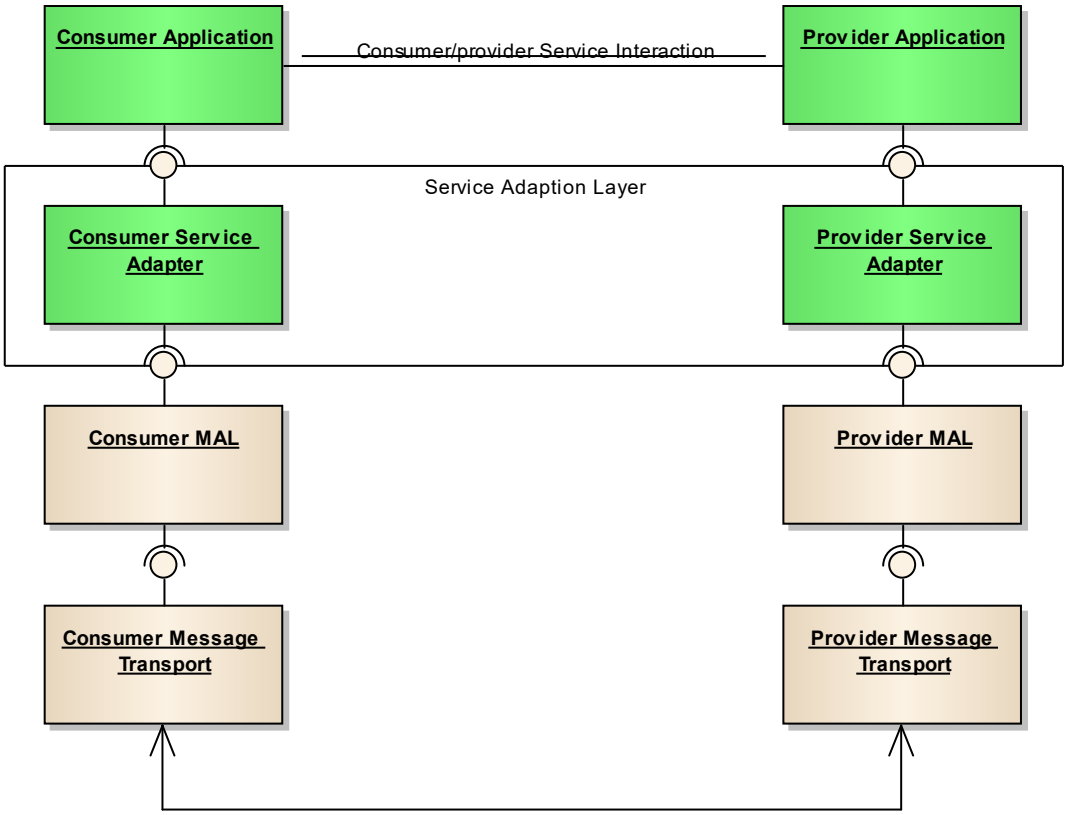


Figure 1-3: Layering Drawing Convention

Figure 1-3 shows a software and MO 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] *Unified Modeling Language (UML)*. Version 2.2. Needham, Massachusetts: Object Management Group, February 2009.

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

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

[4] *Mission Operations Common Services*. Recommendation for Space Data System Standards, CCSDS 522.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 2020.

[5] *Information Security Glossary of Terms*. Recommendation for Space Data System Practices, CCSDS 350.8-M-2. Magenta Book. Issue 2. Washington, D.C.: CCSDS, February 2020.

[6] *CCSDS Cryptographic Algorithms*. Recommendation for Space Data System Standards, CCSDS 352.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, August 2019.

[7] *CCSDS Authentication Credentials*. Recommendation for Space Data System Standards, CCSDS 357.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, July 2019.

[8] *Security Guide for Mission Planners*. Report concerning Space Data System Standards, CCSDS 350.7-G-2. Green Book. Issue 2. Washington, D.C.: CCSDS, April 2019.

NOTE – Informative references are listed in annex B.

2 MISSION OPERATIONS SERVICE CONCEPT

2.1 OVERVIEW

The services given in reference [B1] 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.

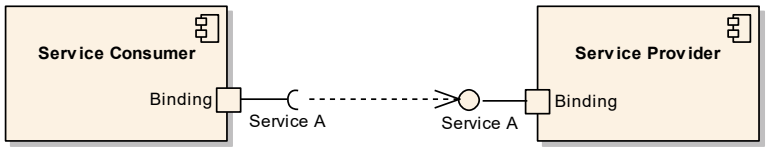


Figure 2-1: Generic Service Model

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

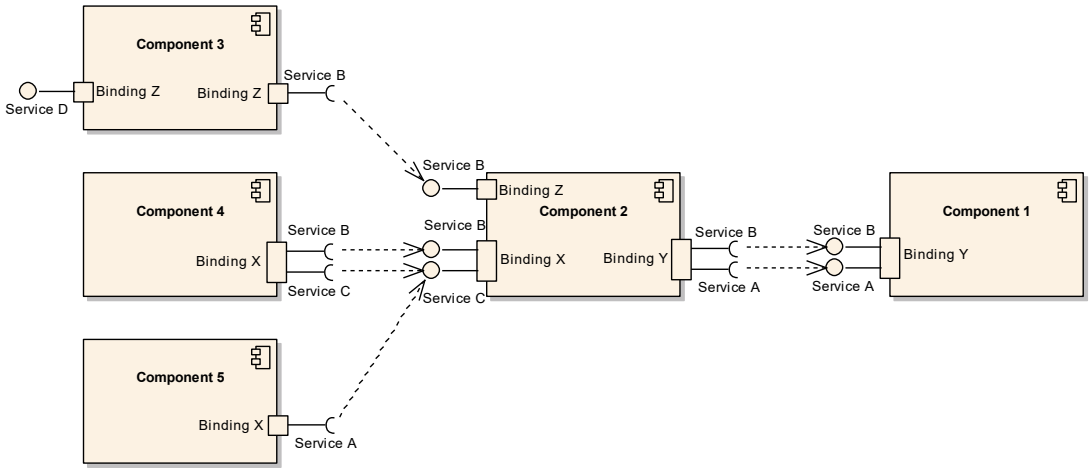


Figure 2-2: Complex Service Model Example

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

Services are made up of sets of operations. An individual 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 [B1] 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 [3]) 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 the function of ‘sendCommand’. 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 a set of basic data types, an object model, and how they must be used to build up the messages that make up the operations of a service, as an abstract API. Implementations must map this abstract API to a specific implementation language, yielding a concrete API for this language. Different implementations may use different APIs written using different languages. In order to be interoperable, the same underlying transport bindings and encodings must be used by the implementations, either directly or by means of a protocol bridge. These mappings then 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:

– domain concept (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 MO Objects and access control, above this exists the Mission Operations Services (MO Services). The MO Services comprise:

– the Common Services (reference [4]);

– the Functional Services (e.g. Monitor & Control Services—reference [B2]).

Both are defined in terms of the MAL. The MAL also provides a generic object model which is used to define the Common Services and which may be used to define the Functional services. Relying on the object model simplifies the task of defining each service by providing a standard way of expressing complex data models. This ensures a common approach in handling certain aspects (e.g. versioning) of the service objects. However, when defining new or custom Functional Services it is not required to adopt the object model or Common Services.

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.

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).

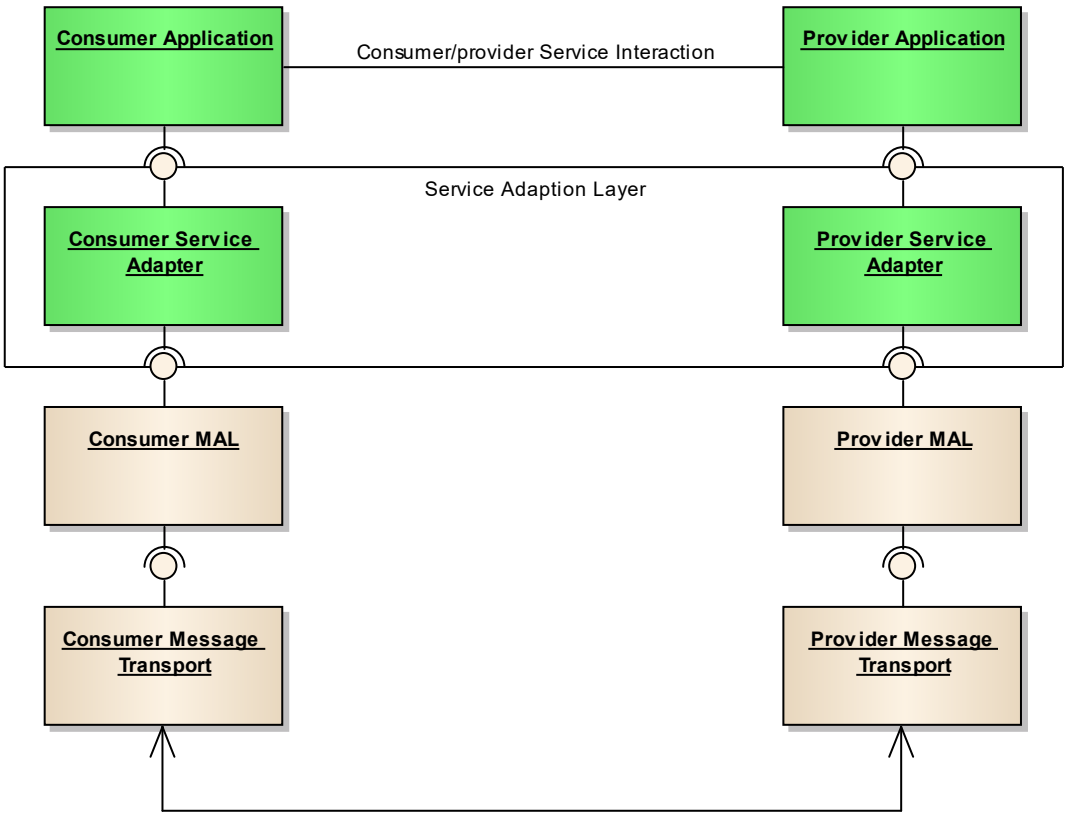


Figure 2-3: Service Stack View

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. The Consumer Service Adapter adapts the MAL API to the consumed service interface; the Provider Service Adapter adapts the MAL API to the provided service interface.

NOTE – A benefit of implementing multiple services over a MAL 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 object model. 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 mapping the MAL to specific message encodings and transports. Mappings to specific implementation languages are not standardized because they are not needed for interoperability.

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) MAL Specification

   Only one book exists defining the MAL.

b) Service Specifications

   Only one book exists for each service specification.

c) Transport Mapping and Encoding

   One book exists for each mapping from the MAL to the specific transport and encoding. Transport and encoding do not always have to be defined in the same book. Some

encodings can be used with multiple transports and some transports allow multiple encodings.

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 [B7], [B8], and [B9]) and implementations are expected to use relevant CCSDS standards also where applicable such as Spacecraft Onboard Interface Services (SOIS) standards (reference [B10]).

Transport-mapping Recommended Standards define technology mappings to specific transports. As of the date of publication these are mappings to CCSDS Space Packets (reference [B3]), TCP/IP (reference [B4]), HTTP (reference [B5]), ZeroMQ Message Transport Protocol (ZMTP) (reference [B6]), and message encodings for Binary (reference [B3]), Split Binary (reference [B4]), and XML (reference [B5]). These mappings allow system engineers to choose a message transport and encoding appropriate for a specific deployment. Protocol bridging may be used to interconnect deployments that use different transports or encodings.

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

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: first, the separation of encoding and transport allows flexibility in deployment (allows the changing of encoding and transport), and second, the choice of language allows portability of an application with a specific implementation of the MAL (allows reuse of software across missions).

NOTE – Whether encoding and transport may be selected separately from each other depends on the concrete encodings and transports used.

For two agencies to interoperate they must standardise on the services that one agency provides to the other and on the transport and encoding selected. This means books chosen from category ‘service specifications’ (as defined in subsection 2.5) must match as well as books selected from category ‘transport mapping’. Standardisation on the ‘MAL specification’ is achieved readily because only one book exists in this category. The choice of implementation language

at each agency is hidden from the other by the MAL and therefore not required for entity interoperability as long as the same transport and data encodings are chosen on both sides:

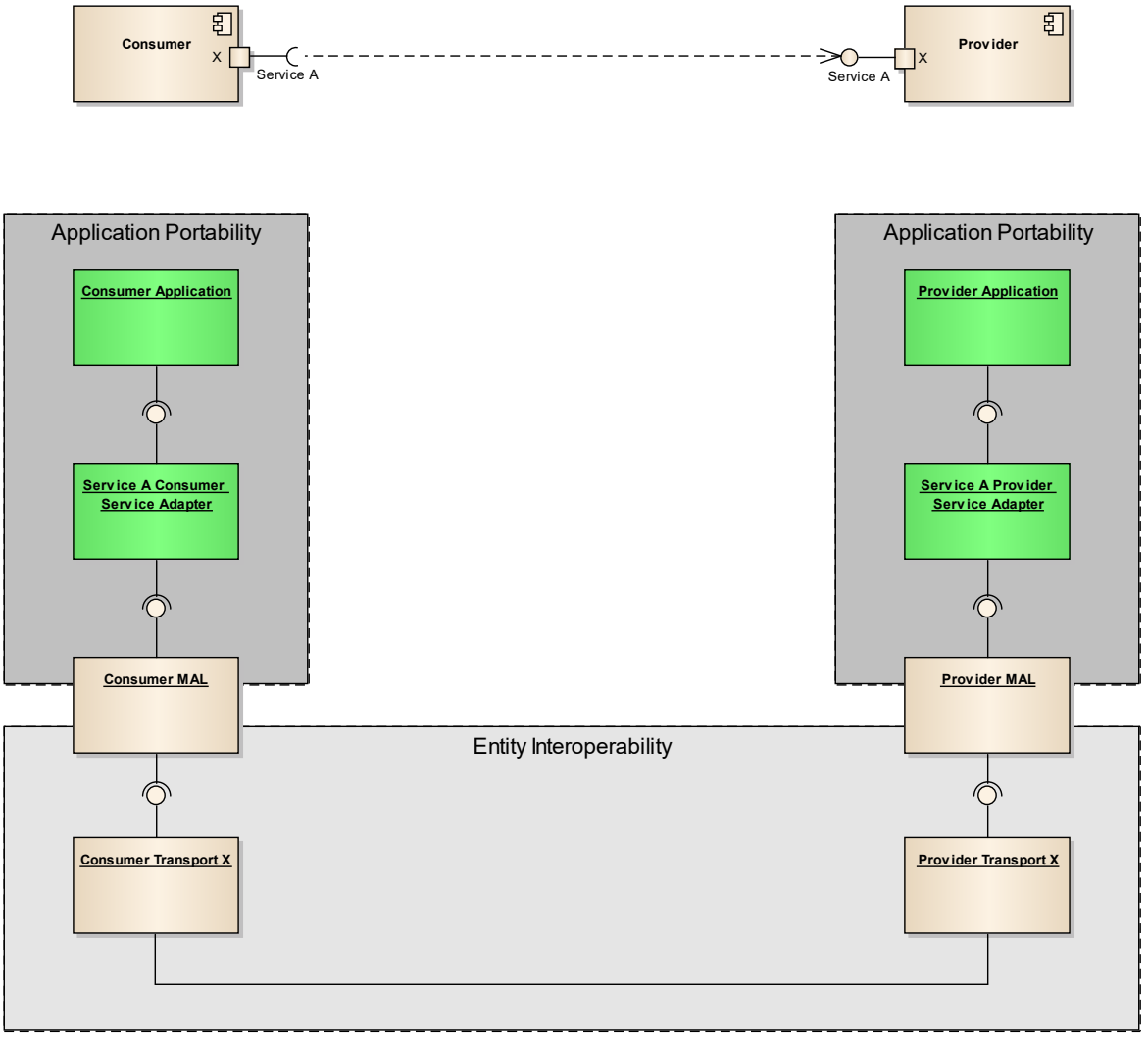


Figure 2-4: Example Entity Interoperability

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:

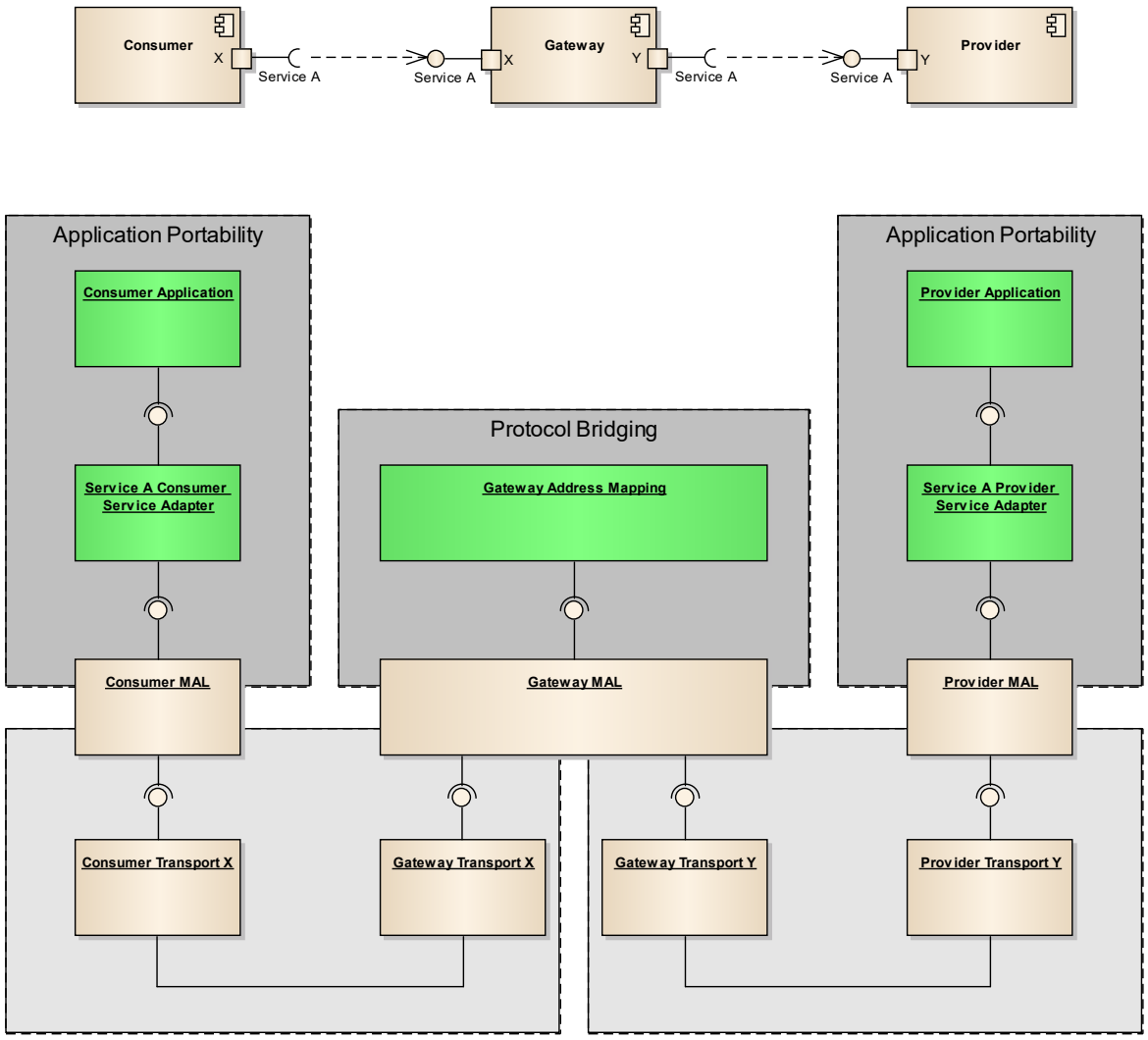


Figure 2-5: Protocol Bridge Example

Figure 2-5 shows that the consumer and provider components are still fully interoperable even though they utilise different transports or 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. A protocol bridge maps between different transports, encodings and addresses. 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, in which one service provider extends the capabilities of another either through the provision of extra information (e.g., statistical services) or extra operations:

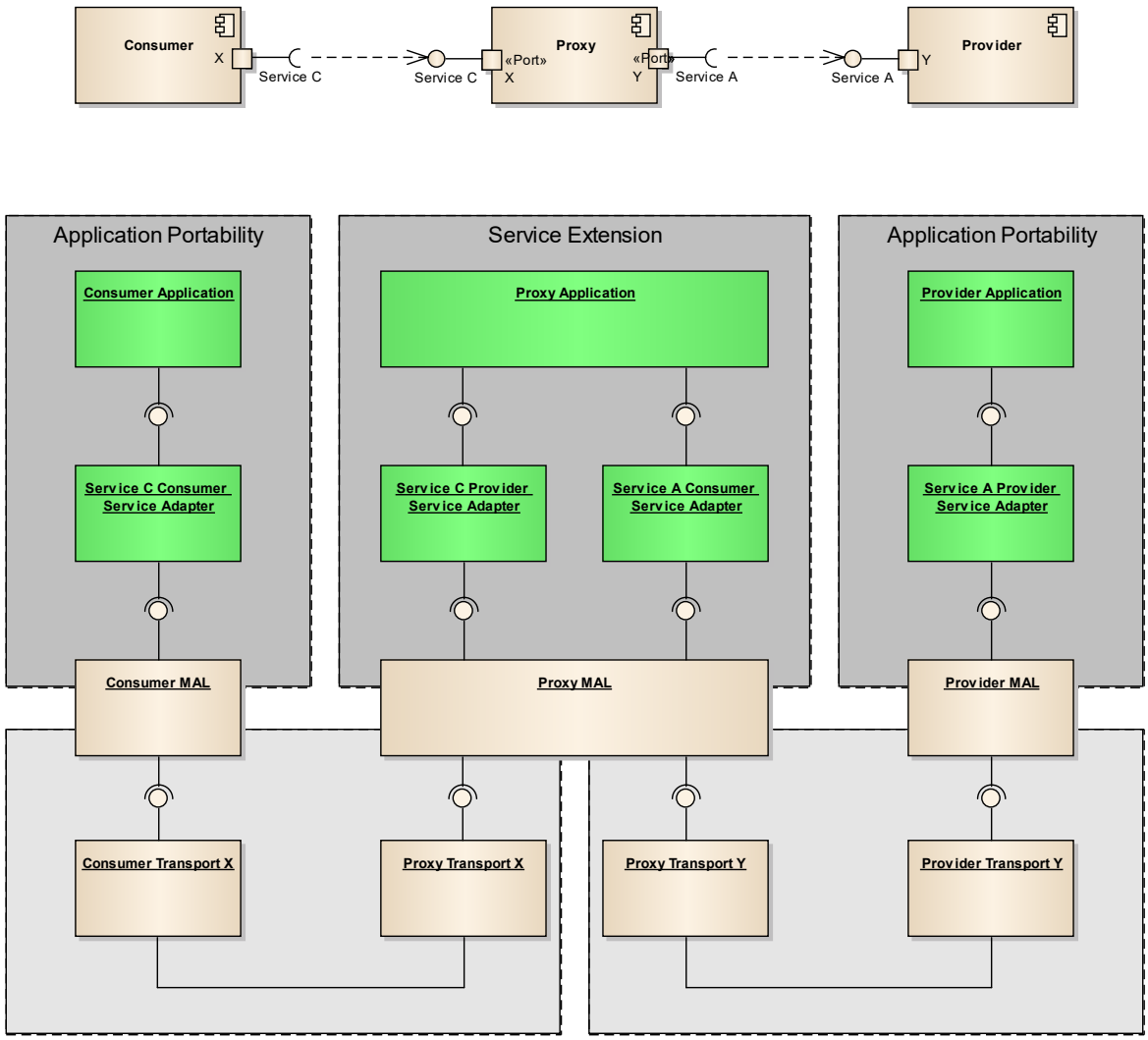


Figure 2-6: Service Extension Example

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 when 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 an MO service deployment and extends the concepts outlined in section 2.

3.2 SERVICE CONTEXT

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

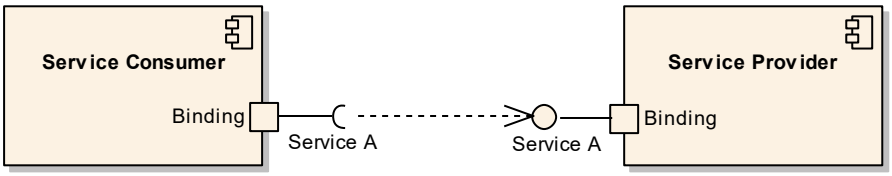


Figure 3-1: Basic MO Service Deployment

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 may be located wherever is appropriate for the deployment, be it on the ground inside a control system or at another ground facility (e.g. Earth-based, lunar or Martian facilities), 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.)

3.3 SERVICE DECOMPOSITION

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

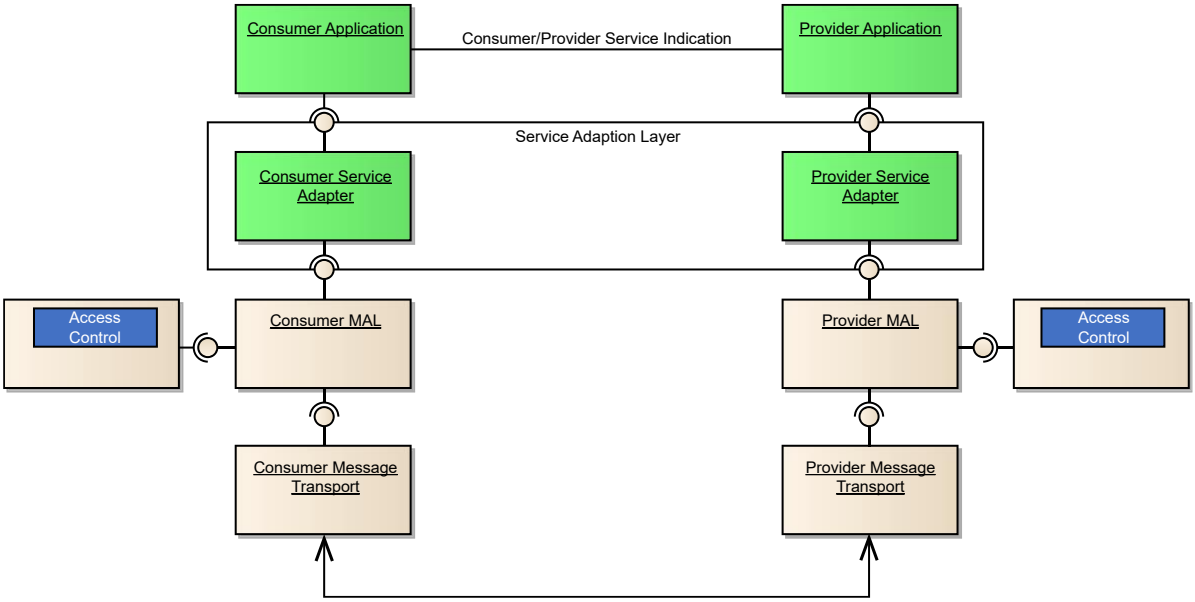


Figure 3-2: Layered MO Service Decomposition

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

b) **Consumer Service Adapter**. Responsible for converting from the messages provided by the MAL abstract interface below it to the service interface exposed to Consumer Application and vice versa.

c) **Consumer MAL**. Provides the standard messaging service that is used by the Service Adapter 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 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 and should be chosen according to [8] in order to determine access control requirements in accordance with mission risk/threat and in accordance with specific agency policies in force.

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. It handles data authentication and confidentiality on the transport level (see 3.6 for details). Combines both the relevant

messaging component (most probably a COTS) and an adaption from the MAL interface to that software.

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

g) **Provider Service Adapter**. Responsible for converting from the messages provided by the MAL abstract interface below it to the service interface exposed to a provider application and vice versa.

h) **Provider MAL**. Provides the standard messaging service that is used by the Service Adapter 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 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 and should be chosen according to [8] in order to determine access control requirements in accordance with mission risk/threat and in accordance with specific agency policies in force.

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. It handles data authentication and confidentiality on the transport level (see 3.6 for details). Combines both the relevant messaging component (most probably a COTS) and an adaption 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 defined as an abstract entity that is passed from one component to another component in the MO service model. In order for it to be communicated from one deployed component to another it must be brought to a concrete on-the-wire representation by means of a transport and encoding mapping.

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 Service Adaption Layer and the MAL API.

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 the on-the-wire representation 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 Layer 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 link and encoding technologies. 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 service entities is a deployment consideration.

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

NOTE – Redundancy may be achieved by several means depending on the deployment. A service consumer might be configured with a list (e.g. by a query to a Directory service or by provisions in an Interface Control Document) of redundant service provider components, all supporting the same service. The consumer then choses a provider according to deployment policy. Other deployments might achieve redundancy transparently to the consumer on transport level, e.g. by employing a load balancer to switch between service provider components. State synchronization between redundant service provider components needs to be considered during deployment planning.

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 component with a logical name (see 3.5.3).

3.5.1.5 Each MAL message contains header fields ('To' and 'From') that hold the logical names of a deployed service provider component and its service consumer component.

3.5.1.6 The definition of the logical names and their mapping to one or more transport addresses is a deployment consideration and outside the scope of this specification.

NOTES

1 If information needs to be passed to the transport layer one may use the 'Supplements' fields of the MAL message header. The meaning assigned to these fields is outside the scope of this specification.

2 A real-world deployment of a service there may be many providers of that service, in many service provider components, and many service consumers. In a simple deployment, logical names of provider and consumer components and their mapping to a transport address are expected to be captured in an Interface Control Document. A more dynamic deployment may rely on a Directory service instead.

3.5.2 DOMAIN

3.5.2.1 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, rovers etc. In order to ensure that unique referencing of entities and data items is possible, the concept of a hierarchical namespace of entities and data items is required, called *domains*.

3.5.2.2 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.

3.5.2.3 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.

3.5.2.4 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, subsystem AOCS for a specific Agency/Mission/Craft becomes:

AgencyY.MissionA.SatB.AOCS

which means operational data cannot inadvertently be confused with data belonging to AgencyY.MissionX.SatY.

3.5.2.5 A domain is used as part of the object identity of MO Objects (see 3.8) and it may be used for subscription matching in instances of the PUBLISH-SUBSCRIBE pattern.

NOTE – Domains are not used for service addressing. A single service provider instance may handle data for multiple domains.

3.5.3 SERVICE ADDRESSING

3.5.3.1 Each service provider and consumer component is addressed by a logical name. This address allows a consumer/provider to locate and communicate with a provider/consumer. MO does not prescribe the format of the logical name.

3.5.3.2 A provider component may support several services as long as they can be differentiated by the information contained in the message header of all messages. The header fields that form a unique service instance identifier shall consist of 'Service Area', 'Service', 'Area version', either 'From' (for messages sent from a provider or broker) or 'To' (for messages sent to a provider or broker), and optionally an agreed set of named values of the 'Supplements' header field. The usage of any named value of the 'Supplements' header field for service instance identification shall be conveyed out-of-band.

3.5.3.3 In order to deliver messages to the correct endpoint logical names shall be mapped to transport-specific routing and addressing data. There shall exist at least one mapping for each transport by which a component is accessible. Each transport binding specification shall specify how components are addressed.

NOTES

1 These mappings may be provided out-of-band in an Interface Control Document for simple static deployments. Mappings may be queried from a Directory service for more dynamic and/or complex deployments.

2 For very simple deployments, where components are accessible from anywhere in the deployment by exactly one transport binding, parties may even agree on using the transport-specific routing and addressing data directly as the logical name, e.g. in form of a URI. However, this approach does not scale to scenarios that employ multiple transports or that need to rewrite transport addresses during routing.

3.6 SECURITY AND ACCESS CONTROL

3.6.1 OVERVIEW

To ensure that only authorised operational entities (consumer or provider) have access to service functions, it is critical that some form of authentication, both of consumer and provider, is an integral part of the concept. To avoid the need for an entity to support multiple authentication methods, it is highly desirable that all entities use the same mechanism and that authentication is required only once per entity 'login' even if multiple services are used. This does not preclude the ability of an implementation to challenge a security entity, however, the mechanism for this is outside the scope of this specification. CCSDS recommendations for authentication credentials are set forth in [7].

Where services are supported over open or public communications paths, a level of security is required to avoid unauthorised access or intrusion. Services, as well as the MAL, must be defined in such a way as to allow them to make use of secure communications channels. When planning the security concept, refer to [8] to ensure the security configuration is in accordance with the existing security policies of the agencies and is properly documented. Examples of possible deployment scenarios showing concrete MO stacks and security policies are presented in [B1].

Mission Operations Services are end-to-end application-level services based on messages which may be carried over different communications channels appropriate to the environment. As a consequence, depending on the security concern of the deployment, protection of messages at an application level can be provided by components of the MAL.

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 (see Section 1.6 for definitions):

- Authentication
- Data integrity
- Data origin authentication

– Authorisation

– Confidentiality

– Non-Repudiation

a) Authentication and authorisation are the main areas of concern for MO. Non-repudiation and integrity are supported by certain authentication solutions.

b) 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.

c) MO supports three modes of access control:

1) No authentication, no authorisation

No authentication or authorisation is applied on MO interactions. The system can only log operations performed but not by whom.

2) Authentication Only

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

3) Authentication and Authorisation

A closed system in which entities must authenticate. Authenticated entities have different levels of access. The system can restrict who performs what.

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

d) If required, data origin authentication and confidentiality can be supported at the MAL layer by means of the corresponding Access Control component and the MAL authenticationId header field.

e) If provider and consumer use the same encoding, this can be used to enable data authentication and confidentiality, depending on the specific binary representation.

NOTE – The MAL is Transport layer agnostic, as a consequence a MAL message could be conveyed by different Transport layers. In such a case, end-to-end data authentication and encryption of messages can only be provided by the MAL (see (g)). Moreover, the implementation depends on the deployment since, according also to the informational report (reference [B11]), security features can be implemented at multiple layers, depending on the mission. Security in each layer answers to different issues and mitigates different kinds of attacks.

f) If in the MO concept it is assumed that confidentiality is provided by the lower Transport layer, it shall be 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.

g) If confidentiality is required all the way to the service component, 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.

3.6.3 AUTHENTICATION AND AUTHORISATION

Authentication of consumers and providers is needed for authorisation purposes. Message data authentication following this authentication can be provided by the Transport Layer and/or the MAL.

Digital signatures can be used to 'sign' a specific message, and through some feature specific to the data 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. CCSDS recommendations for authentication credentials are described in [7].

As a consequence, if end-to-end data origin authentication is needed, two encoding stages shall occur: one in the transport layer and one in the MAL. The encoding in the MAL shall be common to every Access Control in the system, in order to compute the same signature of the message.

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 abstract 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 [4]) provides the mechanisms by which consumer and provider applications provide their security credentials to the system to obtain an authenticationId. This authenticationId is put in a dedicated field in the MAL message header; from this point the MAL performs authorisation using this authenticationId. It can also support data authentication and confidentiality functions, depending on the deployment. The authenticationId can contain data used for authorisation, data authentication or confidentiality of messages.

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/provider application to a receiving service provider/consumer application. The following sequence describes message sending from consumer to provider application but applies likewise for the reverse direction from provider to consumer application.

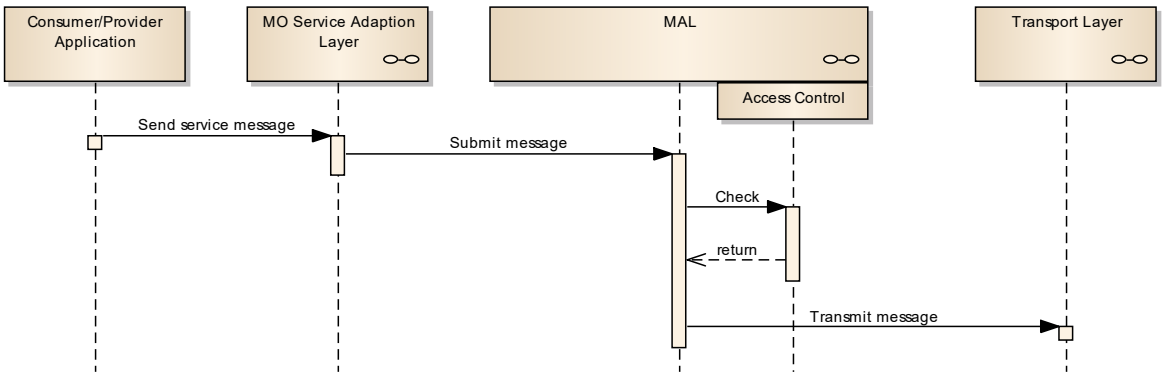


Figure 3-3: Authentication and Authorisation Sending Sequence

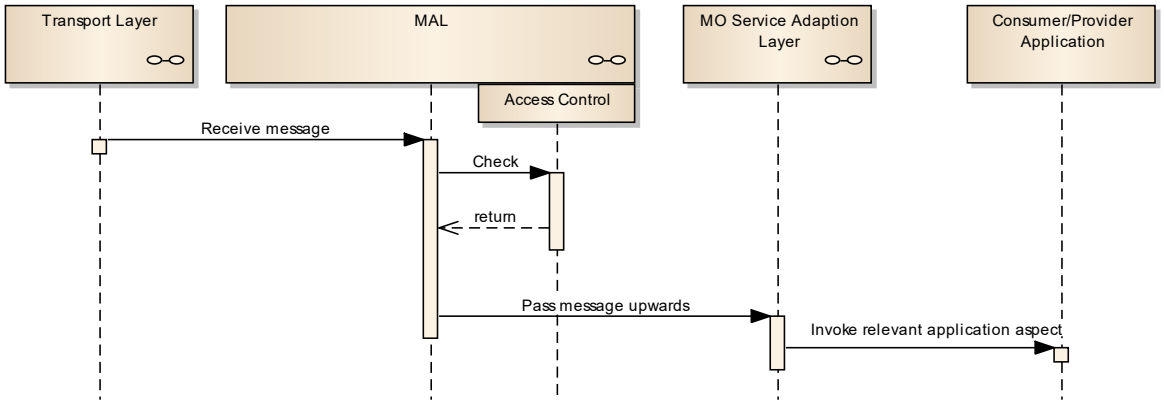


Figure 3-4: Authentication and Authorisation Reception Sequence

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 an authenticationId (the operation identifier is part of the standard MAL message header). The meaning of that authenticationId 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. Dependent on the deployment, it can also contain information for message data authentication or confidentiality purposes.

c) The MAL passes the message with the authenticationId to the consumer Access Control implementation. This performs any required authentication and authorisation checks at the sending side. The Access Control may also perform any processing or modification of the message depending on the deployment. It then passes this back to the MAL.

d) The MAL passes the message, containing the authenticationId, 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, potentially using the consumer access control implementation for digitally signing the encoded message (deployment specific), and then encrypts (if required) the message for transportation. Recommended algorithms for digital signatures and encryption are described in [6].

f) The receiving transport removes encryption relative to the Transport layer. It then performs data authentication of the message relative to the Transport layer 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 authenticationId 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. Depending on the deployment, the Access Control also performs data authentication and removes encryption.

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 authenticationId should be used. It is expected that an implementation of the MAL will be configured to accept this empty authenticationId and not perform any authorisation of it.

j) The MAL then passes the message to the MO Service Adaption Layer that invokes the relevant application aspect of provider application.

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 consumer/provider application to resupply its 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 consumer/provider

application. It is implementation specific how such an authentication challenge is performed.

b) Also, there may be a case for the consumer/provider application to either change its current role or to completely change the consumer/provider application 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 consumer/provider application supplies its security credentials and then the system challenges the existing consumer/provider application to supply its security 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 as shown in Figure 3-5. For example, a control centre can have fine-grained security/authentication inside on the LAN, with dedicated user roles and corresponding authorisations. However, a controlled spacecraft might only provide coarse authentication over the space link, i.e., it might only be possible to authorize whole control centres but not individual users. In this case, a control centre can use a Mission Control System (MCS) as a proxy (“Gateway” in the figure). Access to the MCS is covered by the control centre’s fine-grained security/authentication (checked by the “Gateway Access Control” in the figure). The MCS then bridges to the coarse security/authentication used by the spacecraft, e.g., only allowing properly authenticated and authorised consumer applications to invoke provider functions on the spacecraft on behalf of the control centre (“Authorisation Mapping” in the figure).

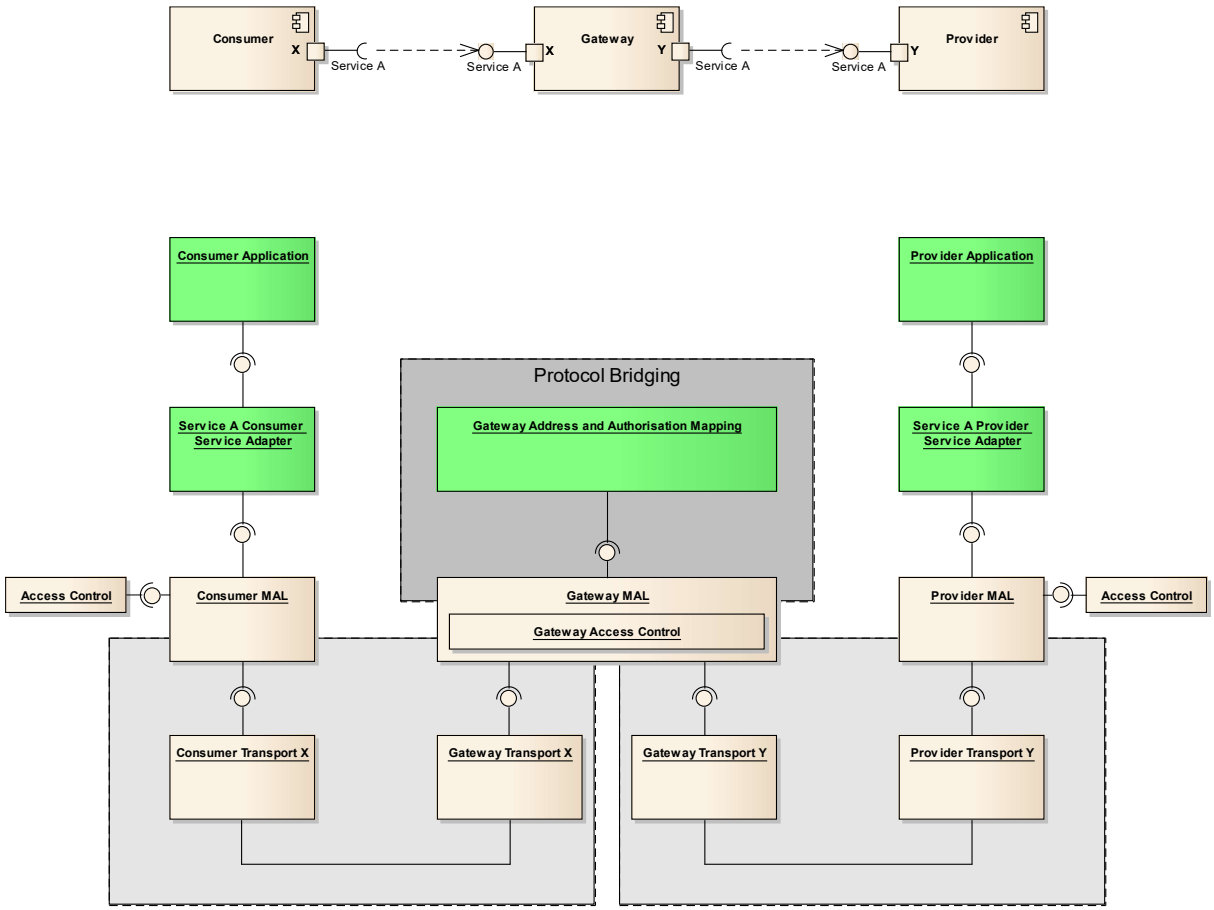


Figure 3-5: Security Bridging

3.6.6.2 By extending the capabilities of the bridge component above the MAL it is possible for a bridge component to filter the messages, according to an implementation-specific internal security model, between the two security areas. An example of a concrete deployment scenario leveraging security bridging is presented in [B1].

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, 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 and video, as well as 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, 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 (e.g., by using suitable error correction or reliable delivery protocols), 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.

Some messaging transports, MAL implementations, or application implementations might provide means to select QoS abilities. It is a deployment decision how QoS abilities are negotiated: In a simple deployment, QoS abilities may be agreed on in an Interface Control Document. In a more dynamic deployment, some transports may allow QoS negotiation on transport level. MAL and application level QoS negotiation may be achieved by a set of custom named values in the 'Supplements' header field of MAL messages and/or by custom named values in a Directory service. Usage of these named values shall be conveyed out-of-band, e.g., in an Interface Control Document.

NOTE – An example of transport level QoS negotiation is the ability of many transports to specify message delivery guarantees (e.g. at-most-once delivery or exactly-once delivery). An example of application level QoS negotiation is the introduction of a custom 'priority' value in the 'Supplements' MAL header field, which acts as a directive for applications to process messages with different priorities (documented in an Interface Control Document).

3.7.2 QOS FAILURE

It is possible that at some point, possibly because of network issues, constraints due to negotiated QoS abilities 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.

3.8 MO OBJECT MODEL

3.8.1 OVERVIEW

The MAL provides a standard data object model for MO Services to utilise. The MO Object model enables complex data modelling when defining services. When deployed, this allows services to compartmentalise data according to domains (see 3.5.2) and enable versioning of service objects. MO Objects have a unique identity and may be referenced from other MO Objects or data structures in service operations. Although an MO Object always has an associated structure definition, it is not required that this structure is ever exchanged on the wire in service operations.

In general, an implementation is not required to keep versions of MO Objects other than the latest one, except if explicitly required by a service specification or by a concrete deployment. A deployment may choose to provide additional services leveraging the MO Object concept, such as persisting and querying objects or subscribing to object change notifications. However, such services are out of scope of the MAL specification. Instead, all service specifications should directly define all functionality needed for working with their MO Objects. This limits inter-service dependencies and thus lowers deployment complexity.

3.8.2 OBJECT MODEL STRUCTURE

3.8.2.1 An MO Object is defined as a thing which is recognised as being capable of an independent existence and which can be uniquely identified, e.g. an object such as a spacecraft, an event such as an eclipse, or a concept such as a parameter definition. MAL does not limit what may be considered an object by a service specification.

3.8.2.2 MAL specifies how MO Objects can reference each other by providing an ObjectRef attribute type that may be used to link an MO Object to any number of other objects. Links are represented as fields of the MO Object's data structure.

NOTES

1 This allows MO service specifications to define hierarchical object structures.

2 A field may reference another MO Object either directly (if the field is of ObjectRef type) or indirectly (e.g. if the field is of a Composite type that itself contains an ObjectRef field or if the field is a list of ObjectRef items).

3.8.2.3 MO service specifications shall formally define which MO Objects may be referenced by a link. This is accomplished by denoting the data structure of the corresponding MO Object when providing the ObjectRef as follows:

ObjectRef<ExampleStructure>

NOTE – The object reference is shown as having a 'type' of ExampleStructure and is therefore a reference to an MO Object of ExampleStructure type.

3.8.2.4 In case the object reference type cannot be constrained at service specification time or shall be kept open deliberately, the representation shall use the MAL base type Element as follows:

ObjectRef<Element>

3.8.2.5 MO service specifications may provide further constraints on which MO Objects may be referenced by a link. These constraints are not modelled formally and shall be captured in the text of the service specification.

3.8.2.6 Each service shall specify its usage of the MO Object model as detailed in the following:

a) Each service specification shall list the MO Objects it defines, the structure used to represent each object and whether each object can be versioned.

b) Each service specification shall specify if the service explicitly requires object versioning, i.e. if the service has requirements on operating on MO Object versions other than the latest one.

c) Each service specification shall specify the lifecycle of its MO Objects, i.e. when and how MO Objects or new versions of existing MO Objects are created or deleted.

d) Each service specification shall specify how the 'Key' identifier as part of the MO Object Identity is populated.

NOTE – Some services may require the service provider to auto-generate a 'Key' identifier with some mechanism to relay the 'Key' to a consumer, other services may require a consumer to supply the 'Key' for new MO Objects.

e) Each service specification shall specify the fields used to link an MO Object to other MO Objects.

f) Each service specification should specify all operations required for working with its MO Objects without reliance on additional services for MO Object management.

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 an 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.

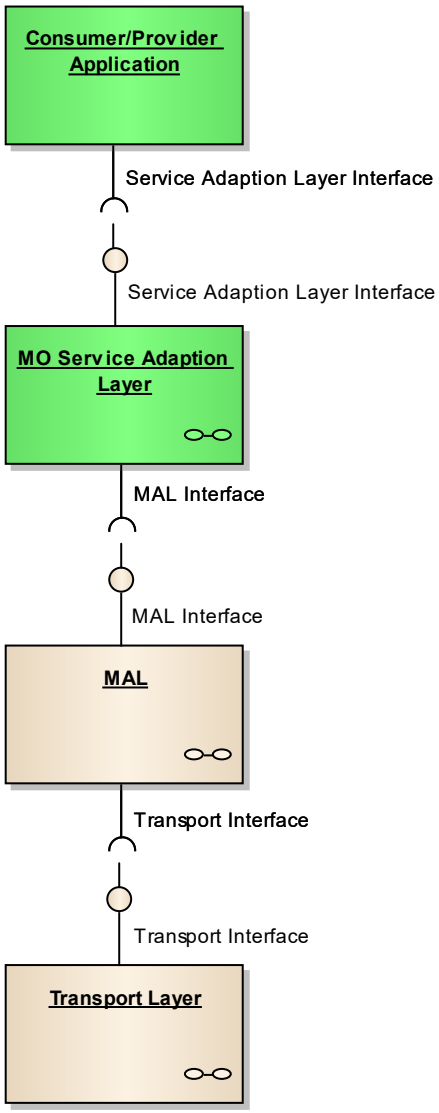


Figure 4-1: Top-Level MO Service Architecture

Each object represents a layer in the MO Service Stack. Each layer provides an abstract 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.

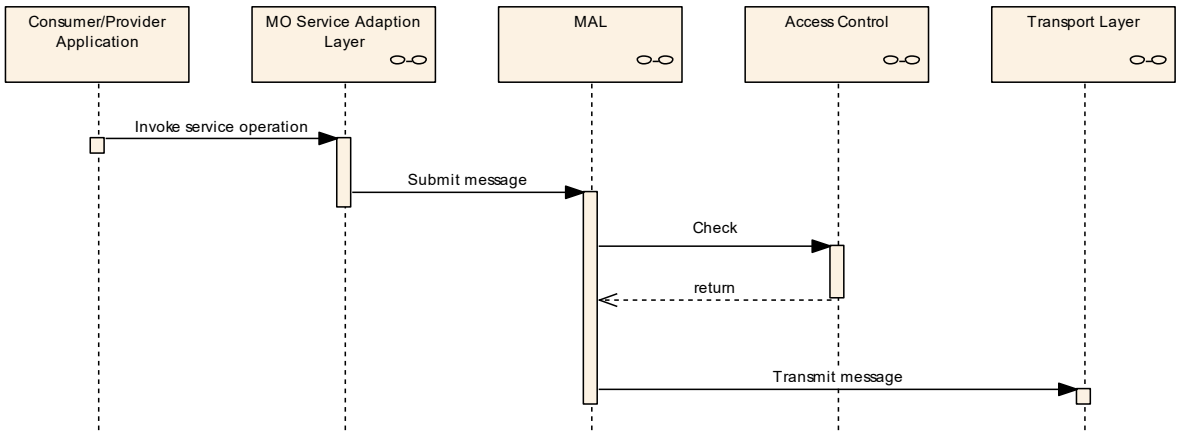


Figure 4-2: High-Level Sending Sequence

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.

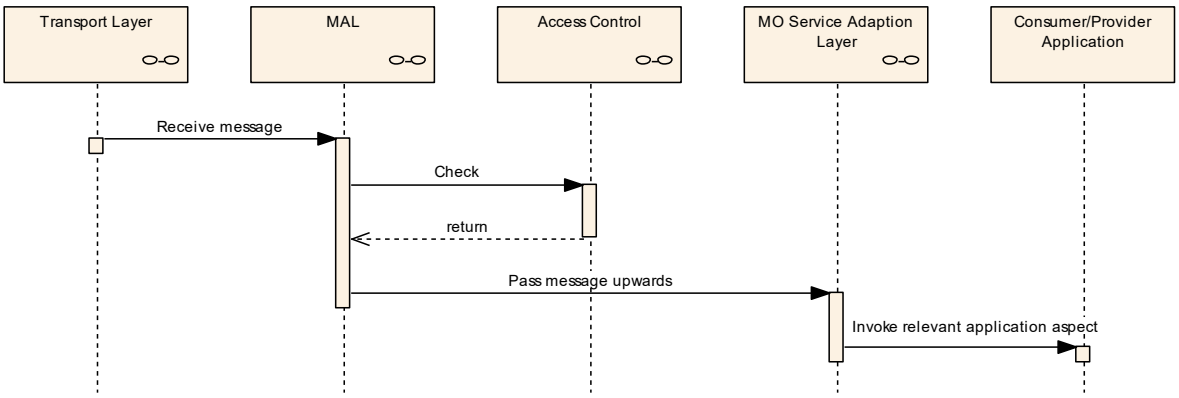


Figure 4-3: High-Level Reception Sequence

a) The Transport Layer receives a message from another Transport Layer in a Sender side 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 Adaption Layer is tasked with converting from the service-specific operation-based interface to the message interface of the MAL. Logically it contains both the Consumer

Service Adapter used by a consumer application and the Provider Service Adapter used by the provider application:

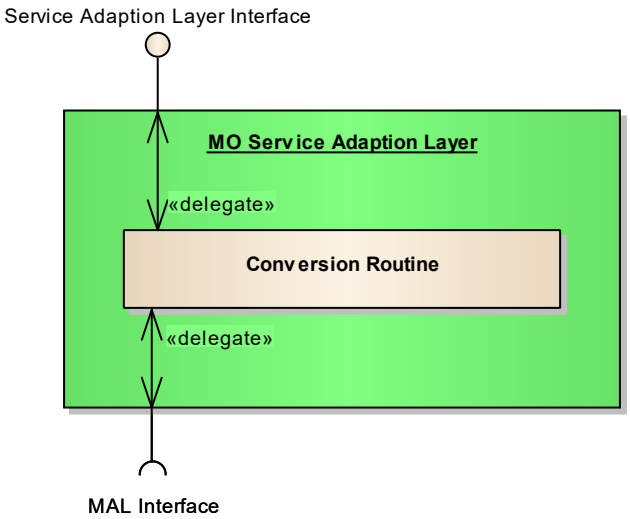


Figure 4-4: MO Service Adaption Layer Architecture

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 language-specific interface of the MAL to send messages and receive messages.

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.

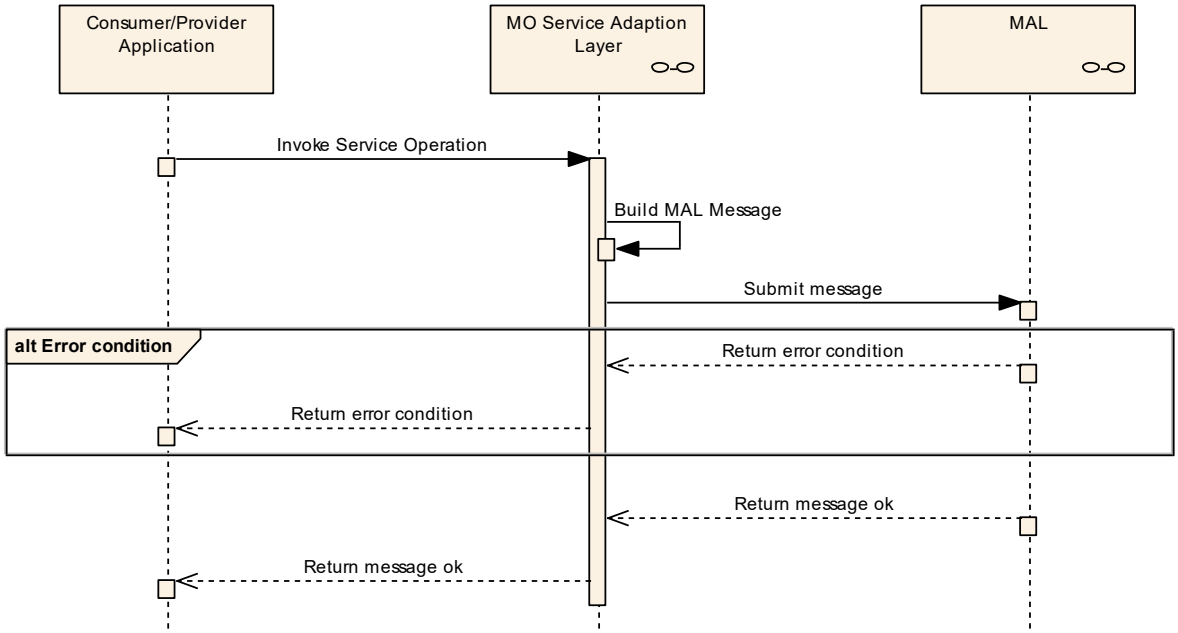


Figure 4-5: MO Service Adaption Layer Sending Sequence

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 in which 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.

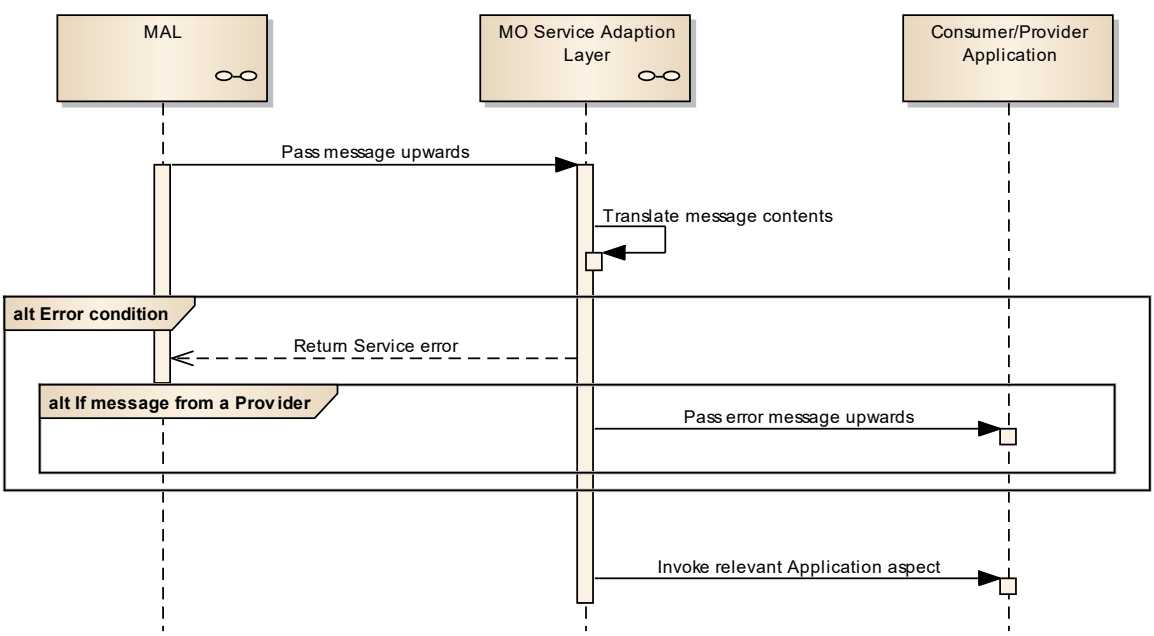


Figure 4-6: MO Service Adaption Layer Reception Sequence

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

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

c) If there is an error during the translation, 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).

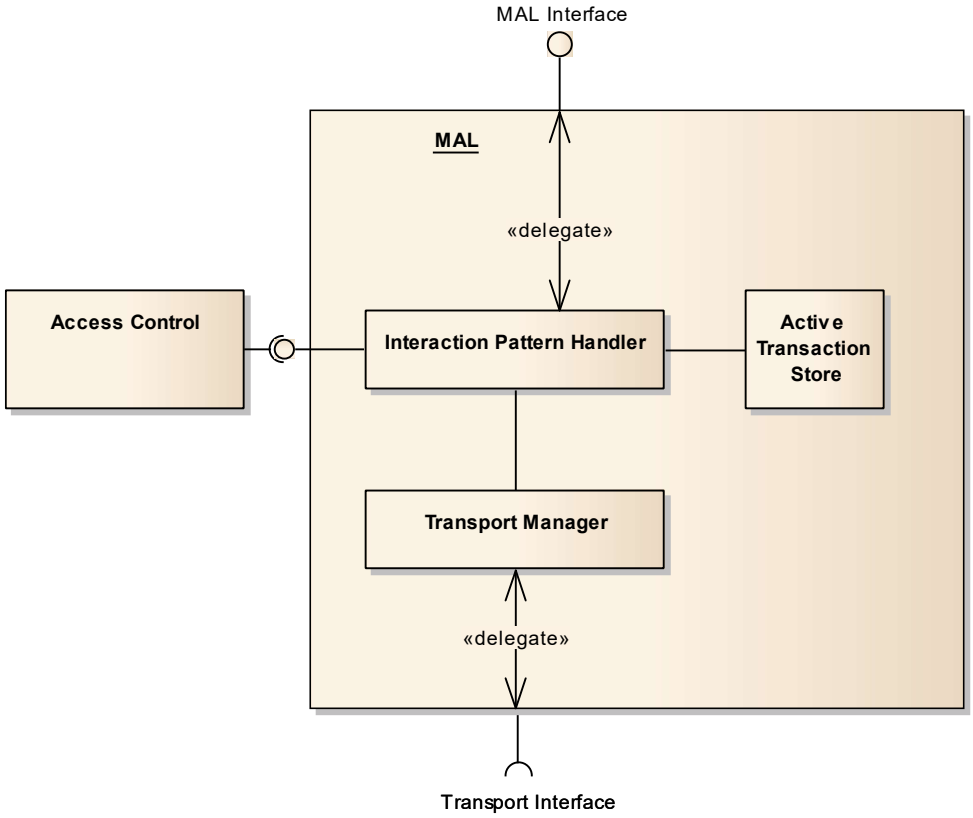
f) If there was an error, the sequence ends at this point.

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

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 layer and the physical-transport-based layer.



87

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 required that these must be implemented as separate software modules, but that their behaviour is implemented as designed.

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 Transaction Store is used by the MAL for recording which transactions 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.

d) The Access Control uses a standard abstract interface defined by the MAL and provides access control for entities such as users to services. It can reject or modify messages that pass through the MAL. The Access Control, depending on the deployment, can also provide functions of message data authentication and confidentiality. The actual control policy in place is deployment specific and should be in accordance with CCSDS recommendations [8] and existing policies of agencies. An example is presented in [B1].

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.

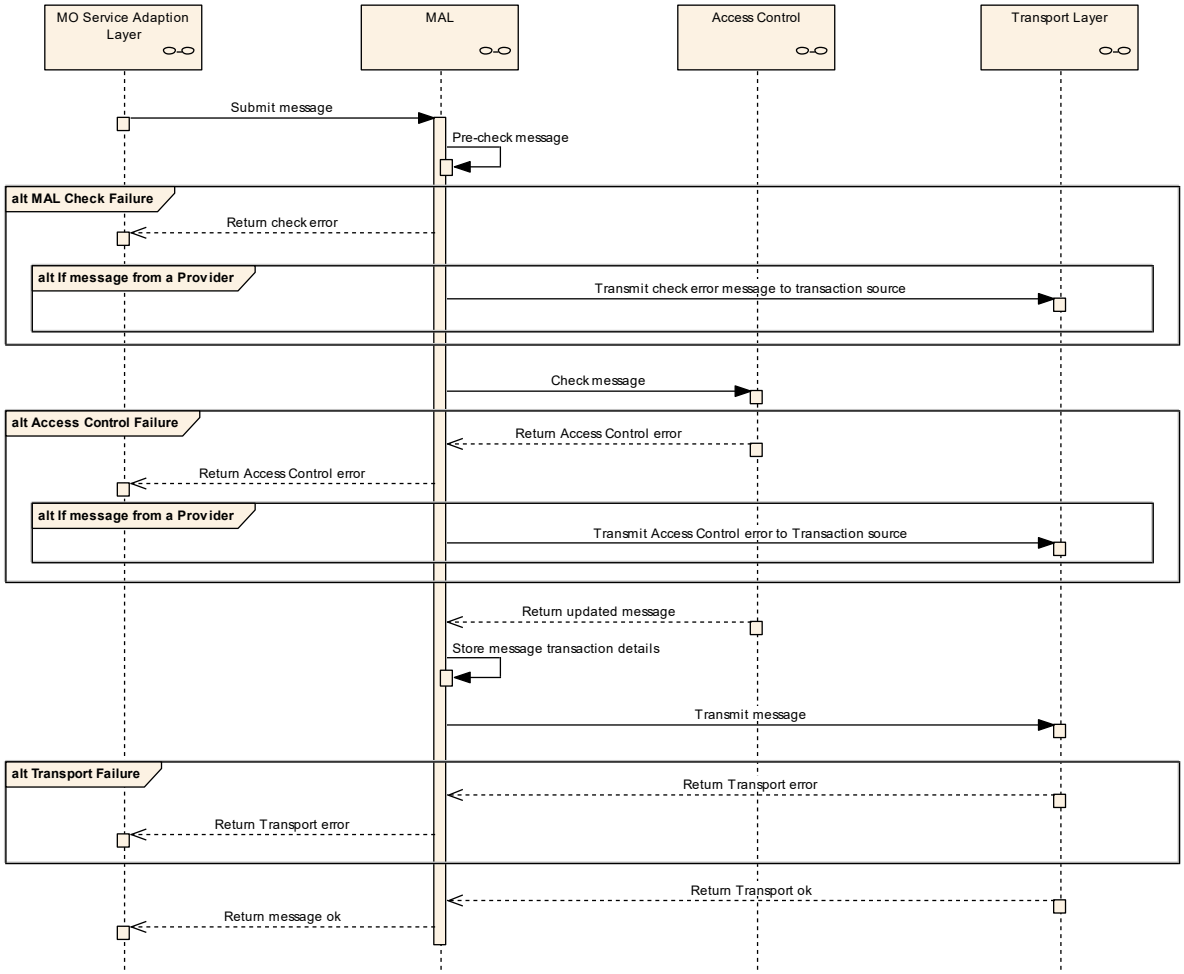


Figure 4-8: MAL Sending Sequence

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, the sequence ends at this point.

e) The MAL then submits the message to the Access Control part using its abstract interface. The processing and modification 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, the sequence ends at this point.

i) If the Access Control part accepts the message, then it returns an updated 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 and the final message for that interaction pattern, then the transaction details are removed from the store.

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.

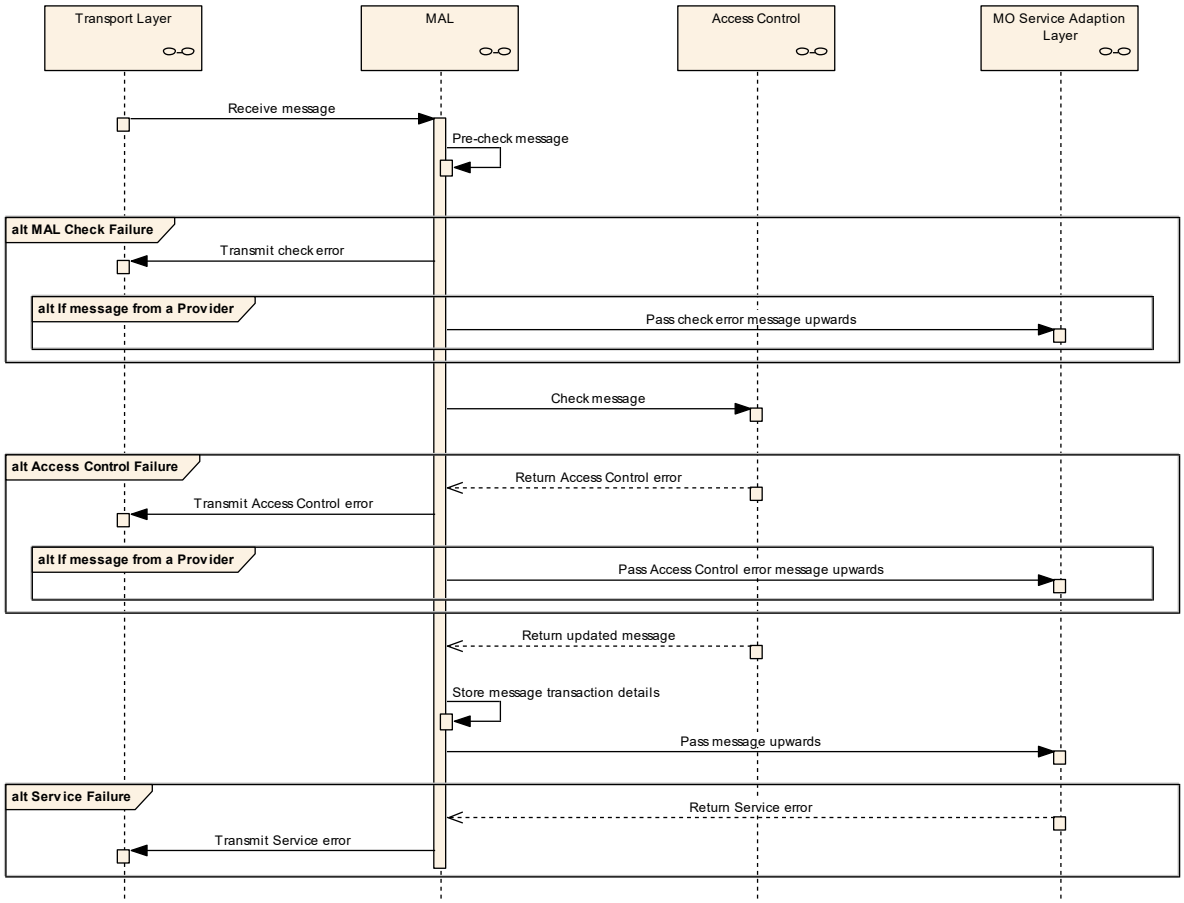


Figure 4-9: MAL Reception Sequence

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 processing and modification 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 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 and the final message for that interaction pattern, then the transaction details are removed from the store.

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 of 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 part is conceptually part of the MAL and is used by the MAL to provide data 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 [3]). Examples of deployment scenarios with concrete security protocols are shown in [B1].

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.

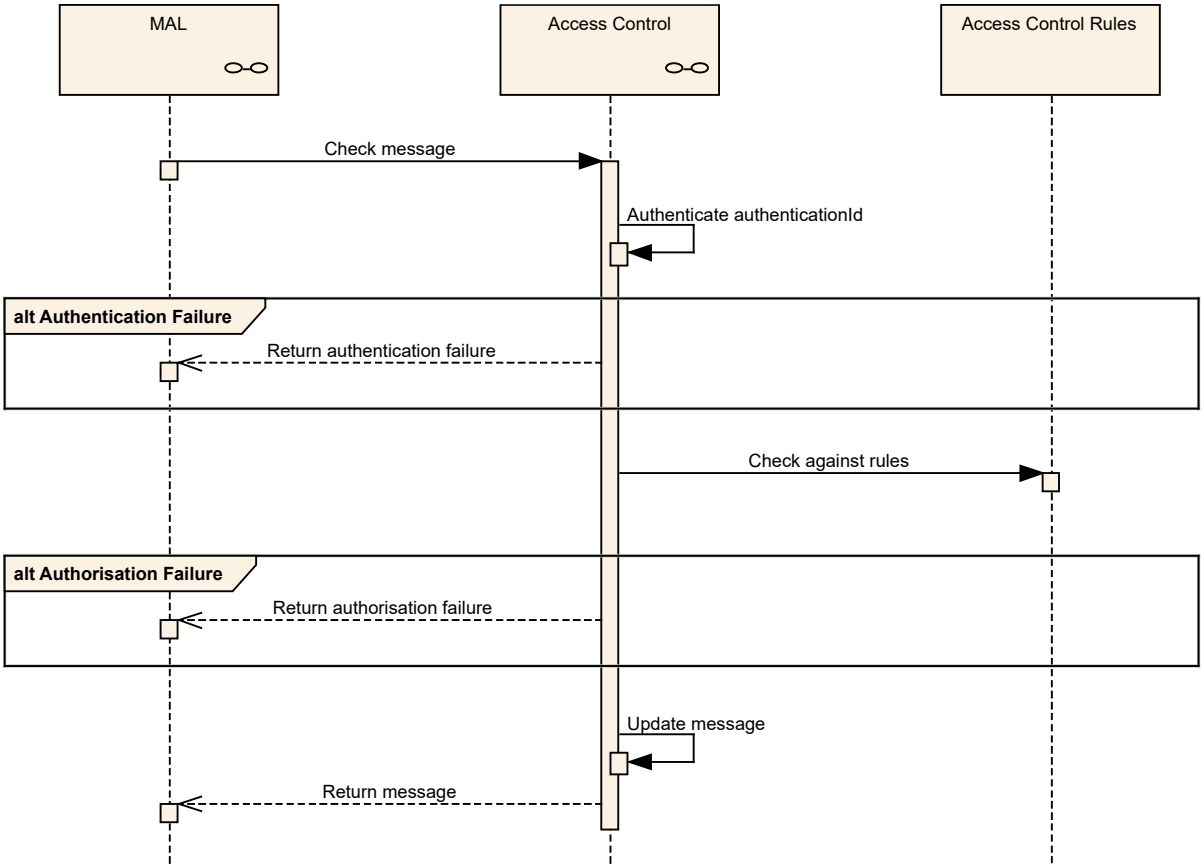


Figure 4-10: Access Control Processing Sequence

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 authenticationId of the message. Depending on the deployment, it can also perform functions of message data authentication and confidentiality. It can apply processing and modification of the message, potentially using the authenticationId.

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 and return the updated message to the MAL. It can apply processing and modification of the message, potentially using the authenticationId.

4.5 TRANSPORT LAYER ARCHITECTURE

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

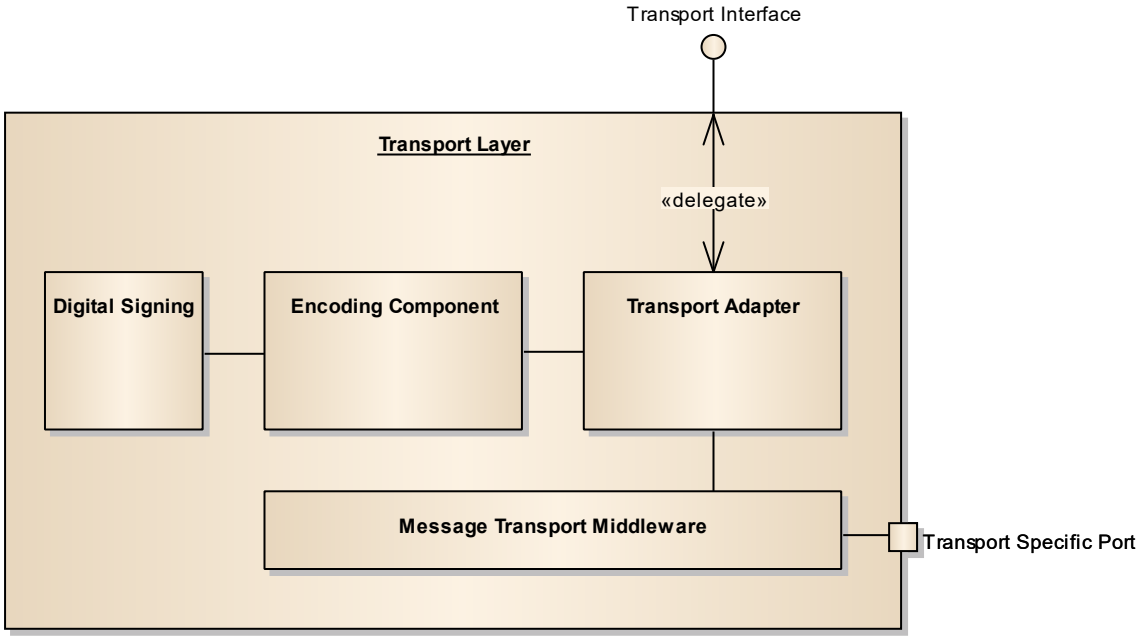


Figure 4-11: Transport Layer Architecture

There are several parts to the layer that are shown in the above figure. It is not required that these must be implemented as separate software modules, but that their behaviour is implemented as designed.

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.

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

c) 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.

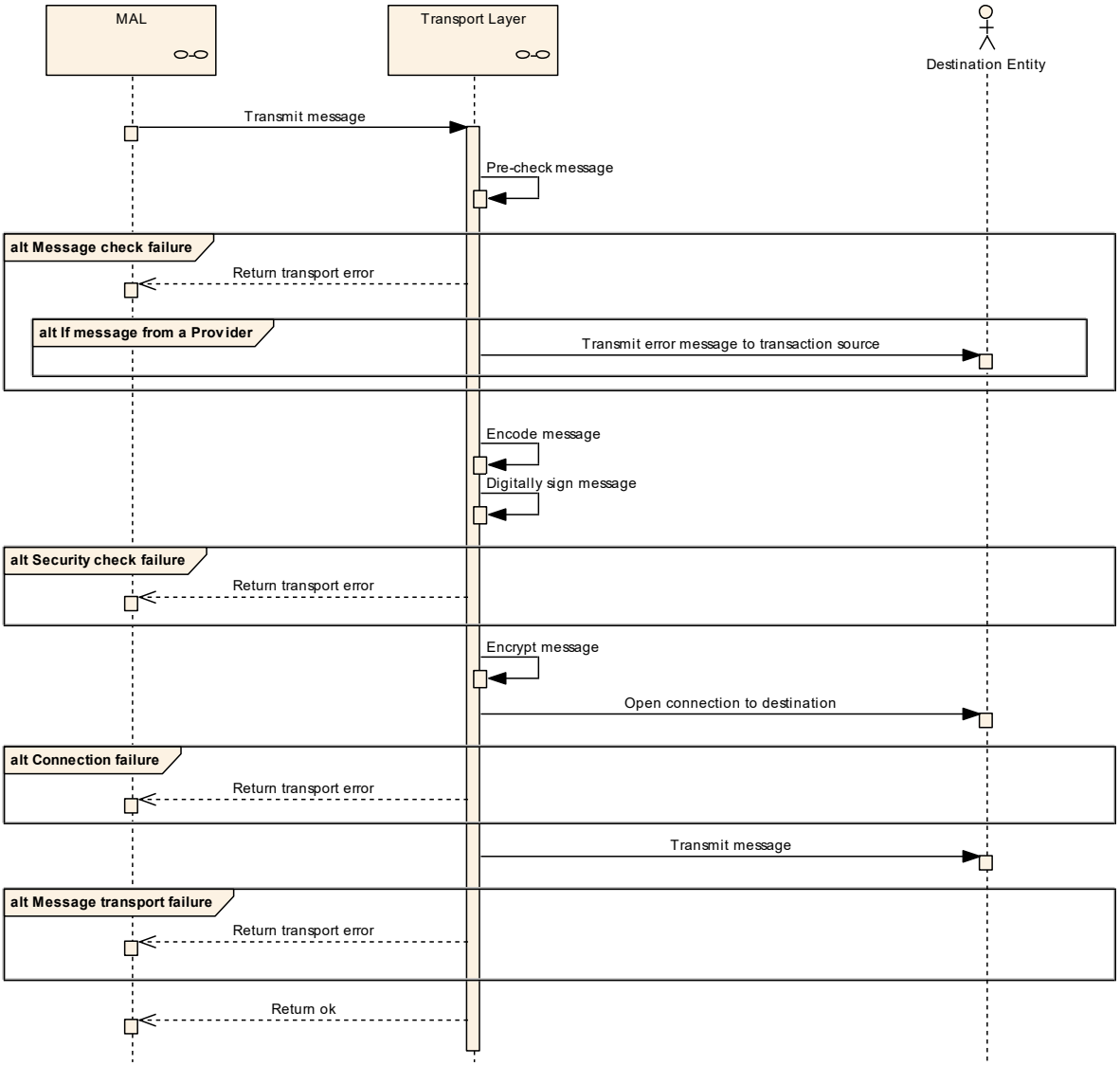


Figure 4-12: Transport Layer Sending Sequence

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 encrypts the encoded message, if required, opens a connection, and transmits the (optionally encrypted) 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.

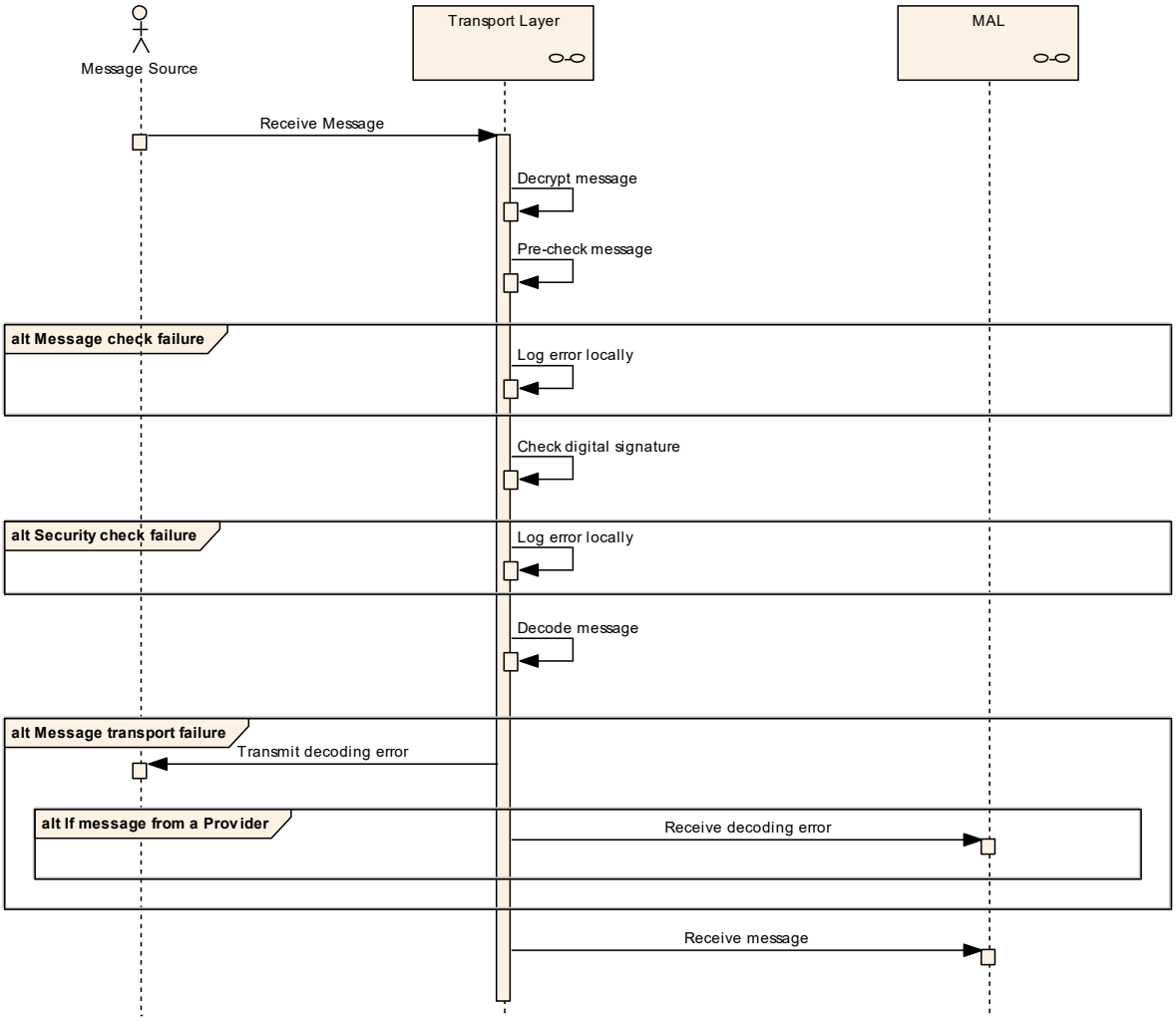


Figure 4-13: Transport Layer Reception Sequence

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, if required, 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.

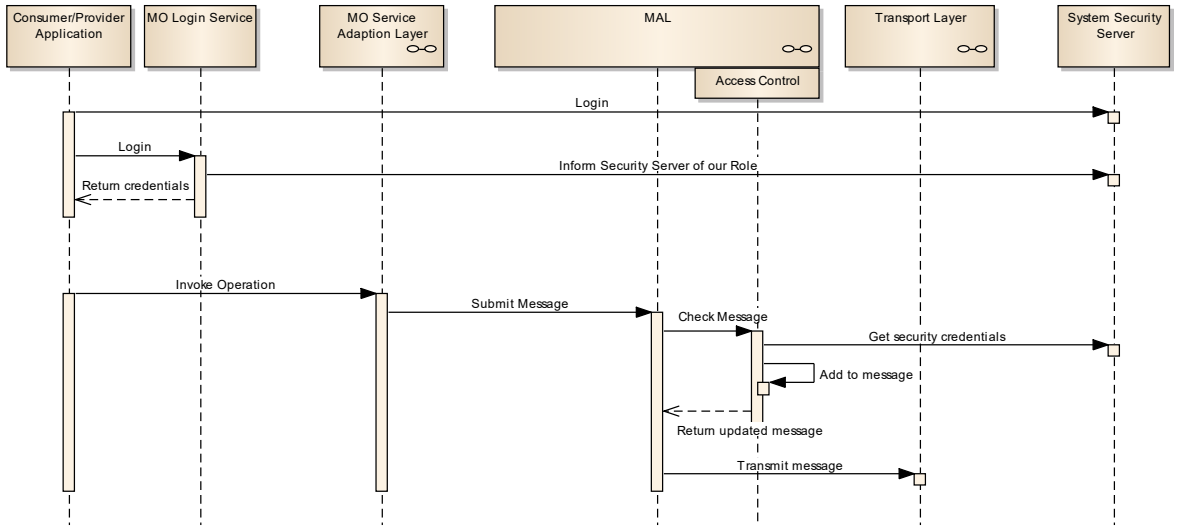


Figure 5-1: Login Sequence

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

a) The consumer/provider application logs into the operating system as normal using the relevant technology and account credentials.

b) The consumer/provider application then uses a deployment specific MO Login service implementation to inform the system of its role.

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

d) The MO Login service returns the authenticationId 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 authenticationId and whether it is required that these then be used in other messages passed by a consumer or provider to its

MAL implementation. For example, it is possible that a specific implementation is able to determine the correct authenticationId from the system without any further information.

f) The consumer/provider application then uses a language-specific API on the MO Service Adaption Layer to invoke an 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 authenticationId, 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. More details regarding a possible implementation of a security challenge are presented in [B1].

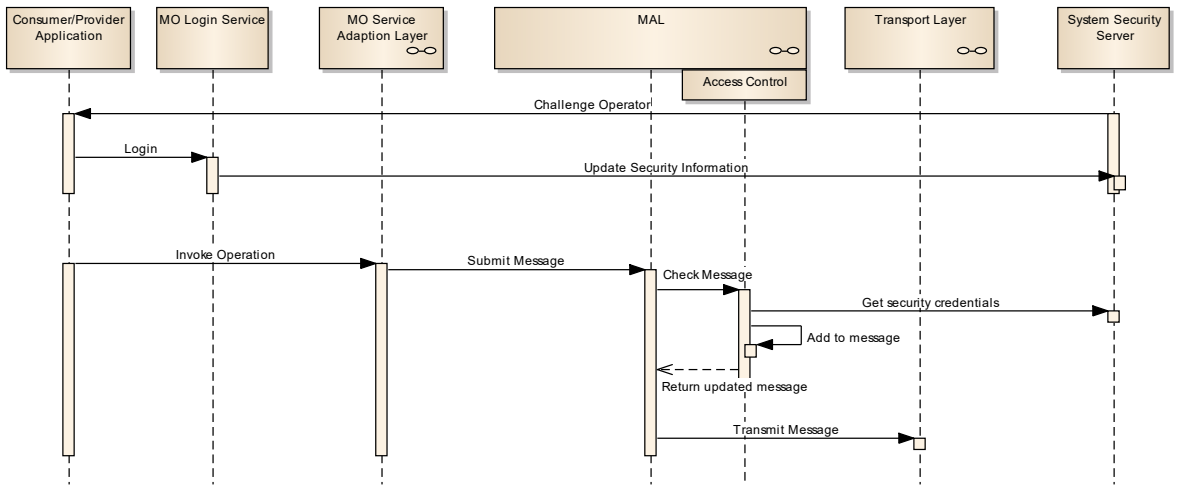


Figure 5-2: Security Challenge Sequence

The basic outline of this sequence is that the consumer/provider application is challenged to revalidate its security credentials by the security system deployed, leading to an updated authenticationId. This updated authenticationId is then used by the MO service system in future messages:

a) The consumer/provider application is challenged by the security system to re-enter its security credentials. This is a deployment specific operation.

b) In this example the MO Login service is used to re-enter the consumer's/provider's security credentials and derive a new authenticationId.

c) In the message exchange sequence, in which the Access Control implementation updates the messages, the new authenticationId is used from this point in time onwards.

d) It is an implementation and deployment detail how the new authenticationId is 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.

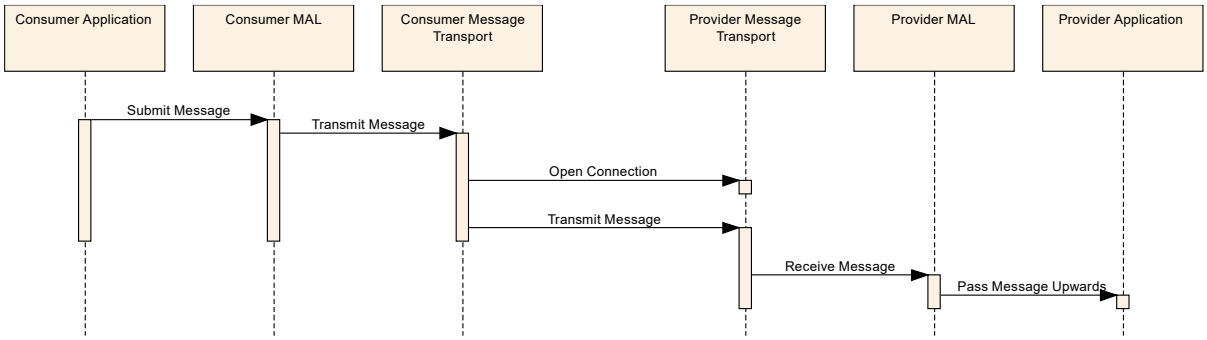


Figure 5-3: Initial Communications Sequence

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)

API	Application Programming Interface
COTS	Commercial off-the-shelf
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LAN	Local Area Network
MAL	Message Abstraction Layer
MCS	Mission Control System
MO	Mission Operations
PDU	Protocol Data Unit
QoS	Quality of Service
RPC	Remote Procedure Call
SDU	Service Data Unit
SOIS	Spacecraft Onboard Interface Services
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URI	Universal Resource Identifier
XML	Extensible Markup Language
ZMTP	ZeroMQ Message Transport Protocol

ANNEX B

INFORMATIVE REFERENCES

(INFORMATIVE)

[B1] *Mission Operations Services Concept*. Report Concerning Space Data System Standards, CCSDS 520.0-G-4. Green Book. Issue 4. Washington, D.C.: CCSDS, [forthcoming].

[B2] *Mission Operations Monitor & Control Services*. Recommendation for Space Data System Standards, CCSDS 522.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, October 2017.

[B3] *Mission Operations – MAL Space Packet Transport Binding and Binary Encoding*. Recommendation for Space Data System Standards, CCSDS 524.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, August 2015.

[B4] *Mission Operations – Message Abstraction Layer Binding to TCP/IP Transport and Split Binary Encoding*. Recommendation for Space Data System Standards, CCSDS 524.2-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 2017.

[B5] *Mission Operations – Message Abstraction Layer Binding to HTTP Transport and XML Encoding*. Recommendation for Space Data System Standards, CCSDS 524.3-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, June 2018.

[B6] *Mission Operations – Message Abstraction Layer Binding to ZMTP Transport*. Recommendation for Space Data System Standards, CCSDS 524.4-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, October 2019.

[B7] *Orbit Data Messages*. Recommendation for Space Data System Standards, CCSDS 502.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 2009.

[B8] *Tracking Data Message*. Recommendation for Space Data System Standards, CCSDS 503.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 2007.

[B9] *Attitude Data Messages*. Recommendation for Space Data System Standards, CCSDS 504.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 2008.

[B10] *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.

[B11] *The Application of Security to CCSDS Protocols*. Report Concerning Space Data System Standards, CCSDS 350.0-G-3. Green Book. Issue 3. Washington, D.C.: CCSDS, March 2019.

NOTE – Normative references are listed in 1.8.