The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

MISSION OPERATIONS COMMON OBJECT MODEL

RECOMMENDED STANDARD

CCSDS 521.1-B-1

BLUE BOOK

February 2014

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

MISSION OPERATIONS COMMON OBJECT MODEL

RECOMMENDED STANDARD

CCSDS 521.1-B-1

BLUE BOOK

February 2014

AUTHORITY

Issue:	Recommended Standard, Issue 1
Date:	February 2014
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 *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-3), 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 **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - -- The **standard** itself.
  - -- The anticipated date of initial operational capability.
  - -- The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than three years from its date of issuance, this **Recommended Standard** 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 Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CCSDS shall not be held responsible for identifying any or all such patent rights.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-3). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

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

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

Member Agencies

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

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- 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.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 521.1-B-1	Mission Operations Common Object Model, Recommended Standard, Issue 1	February 2014	Original issue

CONTENTS

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 GENERAL

This Recommended Standard defines the Mission Operations (MO) Common Object Model (COM) in conformance with the service framework specified in annex B of Mission Operations Services Concept (reference [C1]).

The MO COM is a generic service template that provides a Common Object Model to the Mission Operation services defined in reference [C1]. These Mission Operations services are defined in terms of the COM and the Message Abstraction Layer (MAL) (reference [2]).

1.2 PURPOSE AND SCOPE

This Recommended Standard defines, in an abstract manner, the COM in terms of:

a) the operations necessary to provide the service;

b) the parameter data associated with each operation;

c) the required behaviour of each operation;

d) the use of the model.

It does not specify:

a) individual implementations or products;

b) the implementation of entities or interfaces within real systems;

c) the methods or technologies required for communications.

1.3 DOCUMENT STRUCTURE

This Recommended Standard is organised as follows:

a) section 1 provides purpose and scope, and lists definitions, conventions, and references used throughout the Recommended Standard;

b) section 2 presents an overview of the concepts;

c) section 3 presents the COM specification;

d) section 4 is a formal specification of the COM data structures;

e) section 5 is a formal specification of the COM errors;

f) section 6 details the location of the formal service specification Extensible Markup Language (XML) schema.

1.4 DEFINITION OF TERMS

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

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

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

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

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

Consumed Service Interface: The API presented to the consumer component that maps from the Service operations to one or more service data units contained in MAL messages that are transported to the Provided Service Interface.

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

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

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

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

service data unit, SDU: A unit of data that is sent by a Service Interface, and is transmitted semantically unchanged, to a peer Service Interface.

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.

Object: A thing which is recognised as being capable of an independent existence and which can be uniquely identified. An object may be a physical object such as a spacecraft or a ground station, an event such as an eclipse, or a concept such as telemetry parameter. It forms the fundamental part of a service specification, e.g., a parameter definition, a parameter value at a given point in time, a command. There are no requirements on what an object may be except that it must be possible to uniquely identify an instance of it.

Event: A specific object representing 'something that happens in the system at a given point in time'.

Activity: Anything that has a measurable period of time (a command, a remote procedure, a schedule, etc.).

1.5 CONVENTIONS

1.5.1 NOMENCLATURE

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

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.

1.5.2 INFORMATIVE TEXT

In the normative sections of this document sections (3-6), informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

- Overview;
- Background;
- Rationale;
- Discussion.

1.5.3 DRAWING CONVENTIONS

In figures illustrating this document, UML modelling diagrams are used. (See reference [1] for further information regarding diagrams types and their meaning.)

1.6 REFERENCES

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

[1] *Mission Operations Reference Model*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 520.1-M-1. Washington, D.C.: CCSDS, July 2010.

[2] *Mission Operations Message Abstraction Layer*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 521.0-B-2. Washington, D.C.: CCSDS, March 2013.

NOTE – Informative references are listed in annex C.

2 OVERVIEW

2.1 GENERAL

This document contains the formal specification for the Common Object Model (COM). The COM provides a standard object model for MO Services to utilise. The following diagram presents the set of standards documentation in support of the Mission Operations Services Concept. The COM belongs to the Specifications documentation.

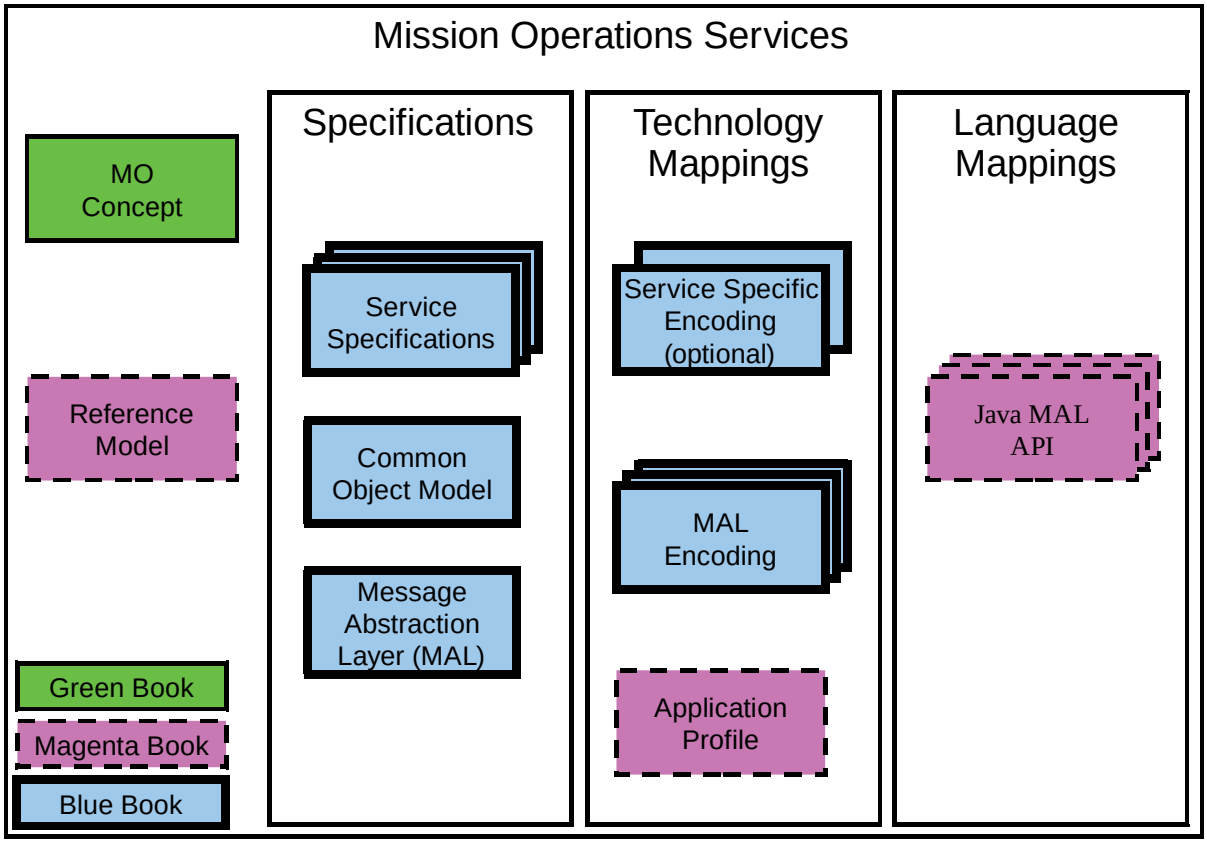

Figure 2-1: Mission Operations Services Concept Document Set

(For further information about the MO Concept, see reference [C1], and for the Reference Model, see reference [1].)

The COM Specification is split into two parts. The first specifies the common object model, and the second specifies the standard COM support services. The services and structures are defined in terms of the MAL so it is possible to deploy them over any supported protocol and message transport.

2.2 COMMON OBJECT MODEL

2.2.1 GENERAL

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

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

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

Each service that utilises the COM must define the object, or set of objects, that form the data model of the service.

2.2.2 COMMON MODEL OBJECT STRUCTURE

Whilst the COM does not limit what may be considered an object by a service specification, it does define how objects are referenced and how they can reference each other. Each object can link to up to two other objects as is shown in figure 2-2.

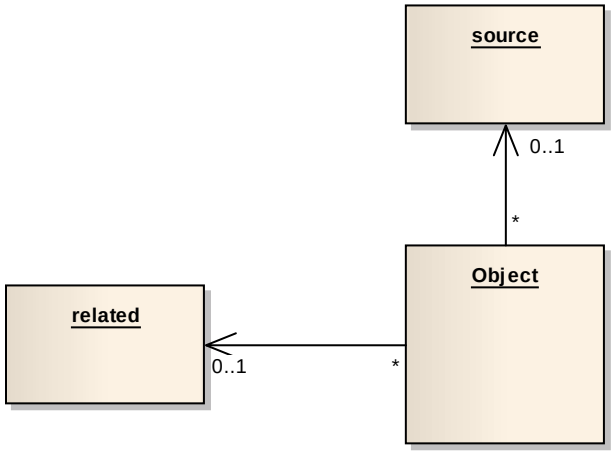

Figure 2-2: COM Structure

The two links have different roles, the related role is expected to be used to link to a related object (for example a parameter value object could link to a parameter definition object), and the source link would be used to link to an unrelated object (for example the operator who requested its value change). It is service specific how these links are to be used.

Each linked object can define its links (so a parameter definition object could define a related link to a parameter name object if this was required) and therefore provides the ability to have 'n' levels of hierarchy.

2.2.3 COMMON MODEL OBJECT IDENTIFICATION

The identity of an object is composed of several parts, the domain of the object, the type of the object, and then the object instance identifier. Objects in one session are conceptually separate from objects in a different session (the rules of the MAL data model require this); the network zone is ignored and does not form part of object identification.

Each service specification is required to list the objects it defines, the structure used to represent the body of the object, and provide each object with a unique (to that service) service object number. The type of an object is the combination of the area number, service number, area version, and service object number. The combined parts are able to fit inside a MAL::Long (for implementations that prefer to index on a single numeric field rather than a structure).

The object instance identifier uniquely identifies an instance of an object for a given domain and object type. The object instance identifier is just a MAL::Long.

2.3 THE SUPPORT SERVICES

The COM specification also includes some support services that build upon the basic object model; these services are:

Event service: An event is a specific object representing 'something that happens in the system at a given point in time'. The event service provides a common mechanism for the distribution of events and also defines how a service that creates events should interact with the archive service.

Archive service: The archive service provides a generic means for persisting objects. It follows the basic Create/Retrieve/Update/Delete (CRUD) principles and therefore fits with most archiving systems. It provides a simple basic set of operations and a basic requirement on the information required to persist service objects in it.

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

The following subsections cover the COM services in more detail, the actual specification of each service is detailed in section 3.

2.4 EVENT SERVICE

An event is a specific object representing ‘something that happens in the system at a given point in time’. The event service defines a single publish/subscribe operation that supports the publishing of events and also the monitoring of events generated by other components.

An event, as it is a COM object, is identified by the normal object fields (domain, object type, and object instance identifier) with the addition of a string name. The name provides a more human friendly means to identify the event.

The body of an event object is assumed to be empty unless one is specified by the service that declares the event.

The COM object links of the event object are used as follows unless stated differently by the relevant service defining the event:

– the source links to the object that caused the event to be generated;

– the related link is service and event-type specific.

When a service implementation requires that the events it generates be persisted, upon event emission, the event should be stored in the COM archive. As an event is just a COM object, the normal archive mechanism (as defined in the archive service) is used to store the event.

2.5 ARCHIVE SERVICE

2.5.1 GENERAL

The archive service provides a basic archiving function for COM objects. It follows the Create Retrieve Update Delete (CRUD) principles and allows simple querying of the archive (more complex queries are supported but the specifications of these are outside this standard).

As changes are made during the lifetime of an object, this information is distributed to consumers using the service defined operations; as long as these updates follow the COM standard for object identification they can also be stored in a COM archive.

By storing these updates in an archive, any historical replay/retrieval functions can correctly reflect the history of the objects.

2.5.2 ARCHIVE OBJECT IDENTIFICATION

As every COM object is uniquely identified by the set (domain, area, service, area version, object type, object instance identifier) this provides the basic key for the storage of objects.

Additionally, each object may be tagged by a timestamp, the URI of the provider, a source object (the fully qualified identifier of the object which is at the origin of this particular object, if any) and a related object (the identifier of another object which gives additional information on this particular object, if any).

2.5.3 RETRIEVAL OPERATIONS

The retrieval operations provide a consumer with the ability to request stored information from the archive in bulk. There are three different retrieval scenarios that an MO consumer function may use for archive access:

Count A count of objects existing at a given point or range in time is extracted in a single transaction.

Retrieval A block of objects covering a period of time is extracted in a single transaction. If no objects exist for the time period no value is returned.

Monitor A Publish/Subscribe subscription of events generated by the archive to allow active monitoring of an archive for changes as they happen.

The count operation returns the count of objects that satisfy the request criteria and the bulk retrieval returns the COM objects that satisfy the selection criteria. The events generated by the archive allows a consumer to subscribe for changes to the archive.

2.5.4 STORE OPERATION

The store operation provides a consumer with the ability to populate an archive with information. It provides a standard mechanism for a COM archive to be populated in bulk. The following scenarios are foreseen:

Monitored A software component is monitoring information from one or more services and storing this information in a COM archive.

Bulk storage A software component is moving information from one archive, most likely a proprietary information store (possibly on-board), to a COM archive.

Consolidation A software component is consolidating information from one or more COM archives (possibly separated by session) into a single COM archive. This is a refinement of the bulk storage scenario where information is also being retrieved from the COM archives.

It is likely that in any deployed system-internal mechanisms would be available for storing information as it is created; however, the COM archive service store operation provides an MO compliant mechanism that allows an MO consumer to access the storage of the COM archive. It allows the creation of components that are agnostic to storage implementations.

2.6 ACTIVITY TRACKING SERVICE

2.6.1 GENERAL

The activity tracking service, or activity service for short, provides the ability to track the progress of activities; an activity is anything that has a measurable period of time (a command, a remote procedure, a schedule, etc.). The basic service provides the ability to track the progress of MAL operations, but it is expected to be used for other processes where appropriate. It defines an event pattern that supports the reporting of the progress of activities from the initial consumer request, tracking its progress across a transport link, to reception by the provider and execution in that provider.

The service uses the event service to report the progress of activities which supports the concept of external monitoring where one component is able to monitor the activities in the system without requiring knowledge of what components are active. This permits the implementation of a single component for the monitoring of activity in the system and also for the archiving of this activity.

It also supports monitoring of activities that are passed via a chain of components to a provider; these intermediate components are referred to as relays in this document. For example, to control a Rover the chain in figure 2-3 may be envisioned.

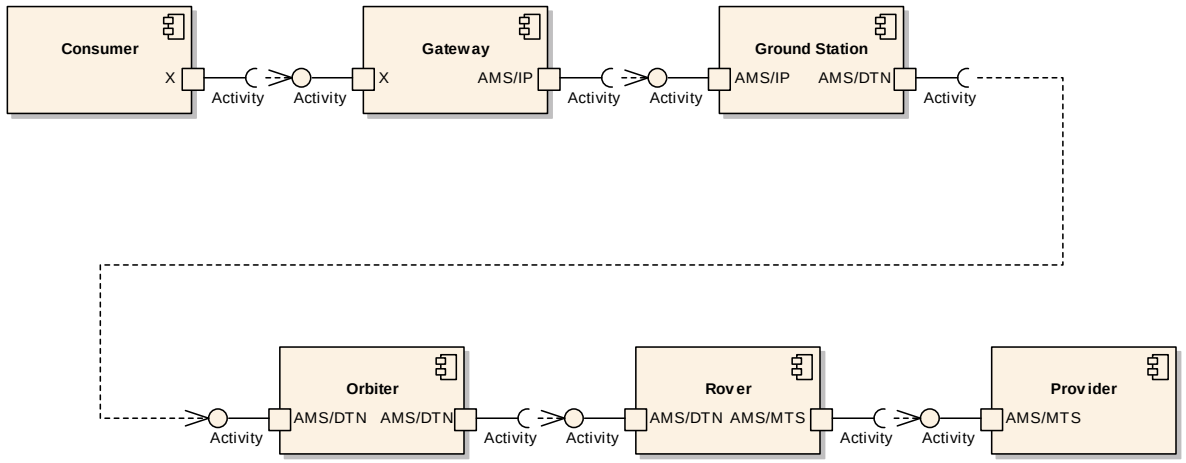

Figure 2-3: Activity Relay Example

Each component passes the operation message from the consumer to the provider with the activity service providing notification of the current location of the message in the chain,

where it is expected to move next, and when. This passing of messages from a consumer to a provider via relays is referred to as multi-hop in this document.

NOTE – It is a deployment decision whether a component generates the activity events. This allows for reporting to be configured depending on network topology or any other criteria deemed suitable.

2.6.2 ACTIVITY EVENTS

2.6.2.1 General

The activity service defines a standard set of events and uses the COM event service for the reporting of the progress of activities from the consumer to the provider and also for execution in the provider.

Four transport events are defined:

- Release is release from source consumer;
- Reception is reception by an intermediate relay;
- Forward is release from an intermediate relay;
- Acceptance is reception and acceptance by the destination provider;

where Reception and Forward events are only used in a multi-hop situation.

Each event has a fixed structure and provides positive as well as negative feedback on the transfer of the activity from consumer to provider (optionally via intermediate relays).

Once the activity has arrived successfully in the provider the execution progress of it is reported using an Execution event. The Execution event may be reported many times depending on the activity:

- Execution is used to report a stage in the execution of the activity.

In regular expression notation the event pattern is:

(Release (Reception Forward)* Acceptance (Execution)*)

The events are distributed using the COM event service, the events themselves link to the activity being monitored (the source of the event) using the object source link.

2.6.2.2 MAL Operations

A MAL operation is an example of an activity and therefore the activity service can be used to monitor MAL operations. The activity service defines a COM object, OperationActivity, which is used to hold the details of a MAL operation. When an interaction is started the

consumer publishes an OperationActivity which all activity reporting events for this interaction use as their source object.

The activity transport events report the progress of the operation from the consumer to the provider. The activity execution event is used to report the execution progress of the interaction in the provider. Table 2-1 defines the mapping from the MAL interaction pattern stages to the activity Execution event.

Table 2-1: MAL Interaction Activity Mapping

Activity Event	MAL Interaction Pattern				
	SEND	**SUBMIT**	**REQUEST**	**INVOKE**	**PROGRESS**
Release	Yes	Yes	Yes	Yes	Yes
Reception*	Yes	Yes	Yes	Yes	Yes
Forward*	Yes	Yes	Yes	Yes	Yes
Acceptance	Yes	Yes	Yes	Yes	Yes
Execution†		Yes for ACK stage	Yes for RESPONSE stage	Yes for ACK and RESPONSE stage	Yes for ACK, PROGRESS and RESPONSE stages
Stage count		1	1	2	Operation specific
* -- Reception and Forward events are only used in a multi-hop situation. † -- Execution events generated are dependent on the MAL Interaction Pattern.					

NOTE – The transport events will not be returned to the initiating application through the MAL interaction, because as far as the MAL is concerned they are different interactions.

If a negative event is being generated by one of the relays, then that relay is also required to fail the MAL interaction that the consumer initiated.

2.6.3 ACTIVITY CHAINING

There are situations where the execution of one activity causes the execution of other activities. For example an automated on-board procedure could trigger the execution of other activities on-board such as other automated procedures or operations. Another example is where one operation is used to load another operation for delayed execution (an on-board command schedule).

In all of these cases there needs to be a link between the activity reporting of one activity and the activity reporting of the other triggered activity. The source link of the COM object model provides this facility, where the second activity uses the COM source link to point to the first activity.

For example, if the execution of one MAL operation causes the execution of another MAL operation the set of activity objects, reporting events and links would be created as shown in figure 2-4.

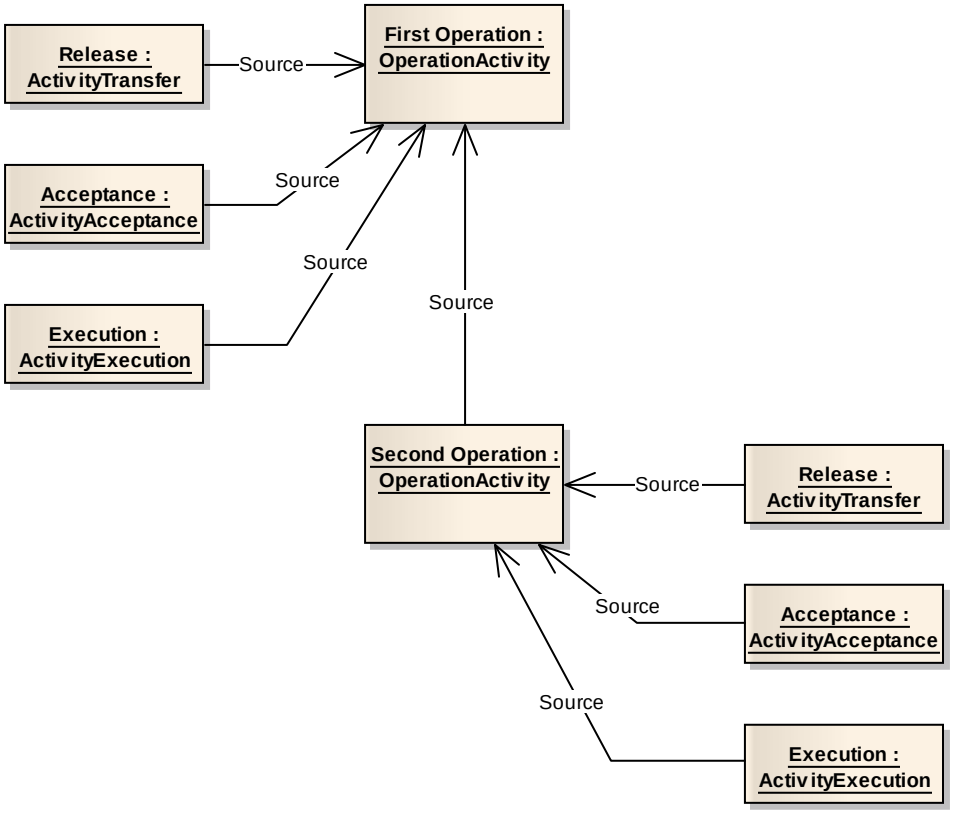

Figure 2-4: Activity Chaining Example

The sequence would be:

- First OperationActivity object is created to represent the operation and also an activity event of type RELEASE that points to the first OperationActivity using the source link.
- An activity event of type ACCEPTANCE is created upon reception and acceptance that points to the first OperationActivity using the source link.
- The execution of the operation causes an activity event of type EXECUTION to be created that points to the first OperationActivity using the source link.

– During the execution of the first operation a second operation is triggered. Second OperationActivity object is created to represent the operation and also an activity event of type RELEASE that points to the second OperationActivity using the source link.

– A second set of activity events are created during the execution of the second operation that point to the second OperationActivity using the source link.

2.6.4 ACTIVITY SERVICE PROXY

In deployments that do not implement the activity service either for the transfer of activities or even for the execution of activities where, for example, another protocol or execution platform is used, it is possible to define a service proxy that converts from the deployment protocol to the activity service as shown in figure 2-5.

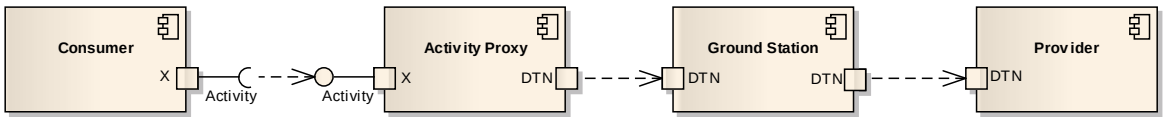

Figure 2-5: Activity Proxy Example

The role of the proxy is to convert from the deployed technology to the activity service messages and would be required to maintain the mapping of which MO activities are represented by the technology specific Protocol Data Units (PDUs) and convert between the reporting provided by the specific technology and that of the activity service.

For example, if Delay Tolerant Networking (DTN) is used to transfer an activity from the consumer to the provider, any DTN bundle reporting messages received would be converted into the equivalent MO activity service event. Any proxy implementation would have to have an awareness of which bundles contained which activities to be able to provide the activity reporting.

Another example of a relevant technology is the CCSDS Space Link Extension (SLE) which is often used between a mission control system (MCS) and the ground station. It is capable of providing reporting information for the transfer status of request from the MCS to the spacecraft. Once again the mapping would have to take into account the relationship between SLE PDUs (most likely CLTU in this example) and the MO activity transfer events.

NOTE – It is outside the scope of this specification to define the mapping from the activity events to a specific technology (such as DTN or SLE); this section only outlines the concept of a service proxy.

2.6.5 SINGLE HOP ACTIVITY EXAMPLE

The Consumer is calling an INVOKE operation on the Provider application. There is a third element (Monitor Application) that is, in this example, providing a system activity monitor role. The publish-subscribe broker is shown also to illustrate how the monitoring application does not need to be aware of which components are present in the system to receive the Activity events (see figure 2-6).

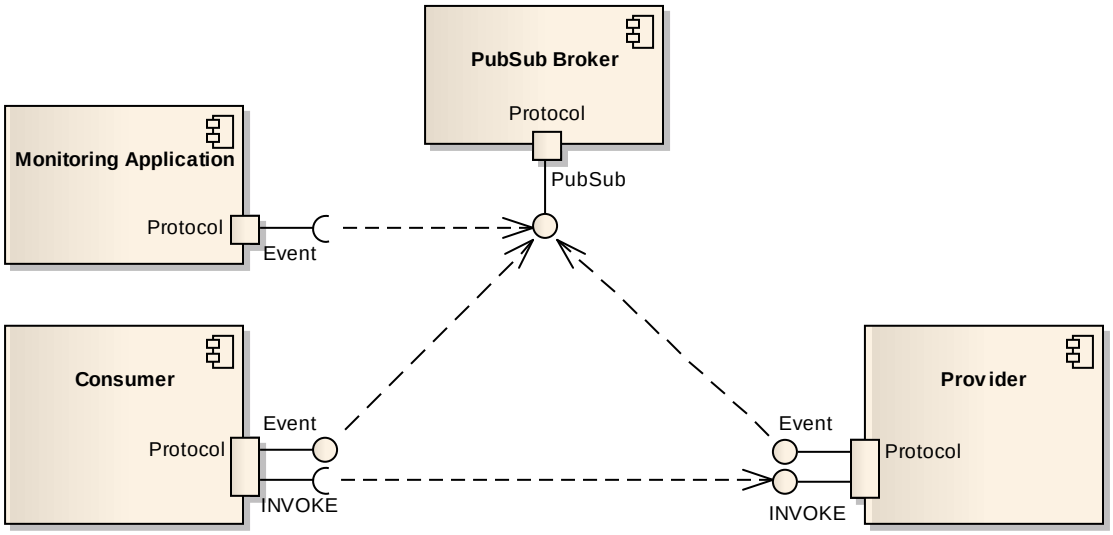

Figure 2-6: Activity Single-Hop Example

For the Activity service the sequence would be:

- Consumer sends an INVOKE message as normal, but also publishes an Activity event of type RELEASE. The Consumer should also archive an OperationActivity object that the activity event uses as a source link.
- Provider receives the INVOKE message as normal from the MAL and immediately publishes an Activity event of type ACCEPTANCE.
- Provider returns the INVOKE ACK message to the Consumer and also publishes an Activity event of type EXECUTION.
- Provider returns the INVOKE RESPONSE message to the Consumer and also publishes an Activity event of type EXECUTION.
- Consumer receives the INVOKE RESPONSE message.

The sequence diagram for this is shown in figure 2-7.

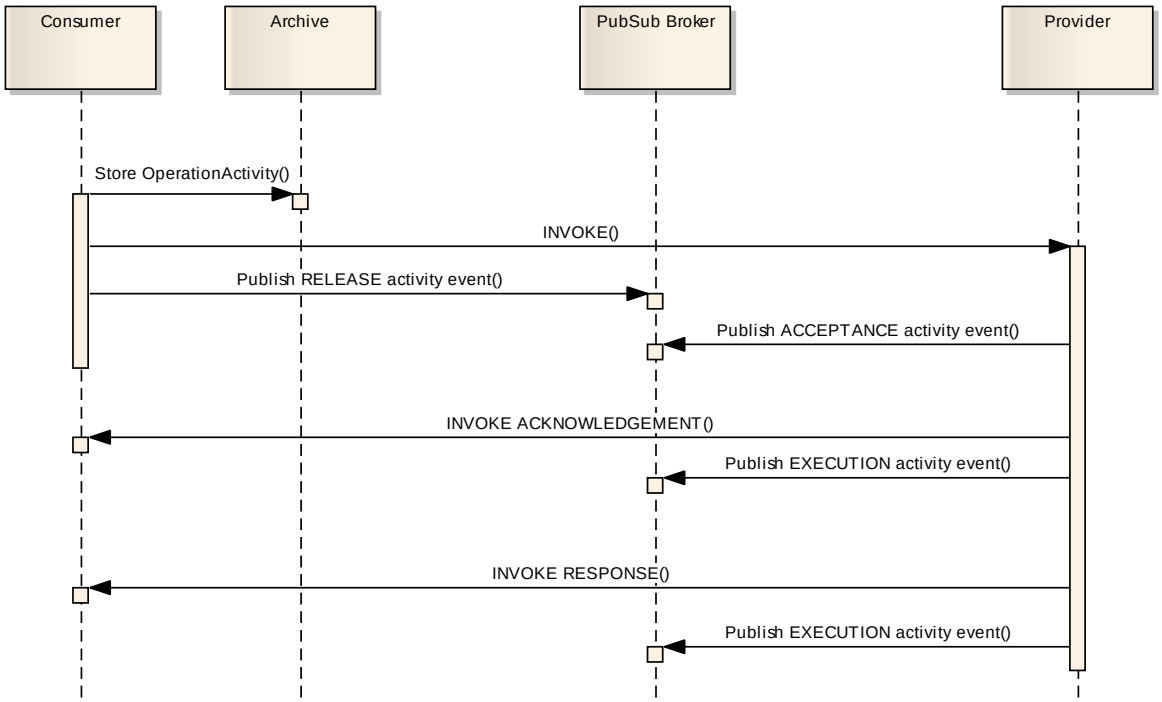

Figure 2-7: Activity Single-Hop Sequence

NOTE – The activity events are published to the event PubSub channel as are any errors. The Monitor application receives all of the events through a subscription to the channel. A standard COM archive could be populated by the Monitor application using this mechanism.

2.6.6 MULTI-HOP ACTIVITY EXAMPLE

The Consumer is calling an INVOKE operation on the Provider application; there is a third element (Monitor Application) that is, in this example, providing a system activity monitor role. The publish-subscribe broker is shown also to illustrate how the monitoring application does not need to be aware of which components are present in the system to receive the Activity events.

In this example the Consumer and Provider are separated by two other relays, a Gateway and Ground Station (GS) relay, which are used to bridge between the three separate networks present (see figure 2-8).

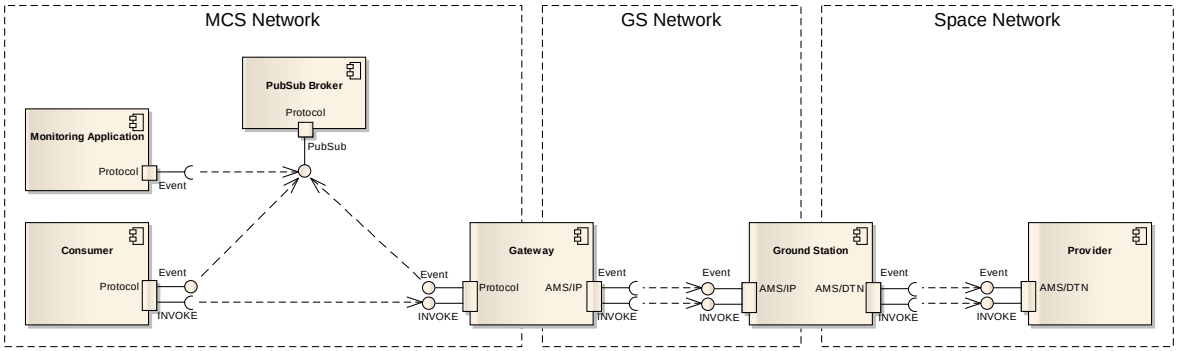

Figure 2-8: Activity Multi-Hop Example

In the MCS network there is a true publish-subscribe broker for the distribution of Pub-Sub events; in the GS and Space Networks, however, the relevant preceding component has subscribed directly with the next component to receive updates. So, for example, the Gateway knows which Ground Station it is using and therefore connects directly to it for the Activity events.

For the Activity service the sequence would be:

– Consumer sends an INVOKE message as normal, but also publishes an Activity event of type RELEASE. The Consumer should also archive an OperationActivity object that the activity event uses as a source link.

– Gateway receives the INVOKE message and publishes an Activity event of type RECEPTION on the MCS network.

– Gateway transmits INVOKE message to GS and publishes an Activity event of type FORWARD on the MCS network.

– GS receives the INVOKE message and publishes an Activity event of type RECEPTION on the GS network.

  • Gateway receives the event on the GS network and PUBLISHES it on the MCS network.

– GS transmits INVOKE message to Provider and publishes an Activity event of type FORWARD on the GS network.

  • Gateway receives the event on the GS network and PUBLISHES it on the MCS network.

– Provider receives the INVOKE message as normal and publishes an Activity event of type ACCEPTANCE on the Space network.

  • GS receives the event on the Space network and PUBLISHES it on the GS network.

    - Gateway receives the event on the GS network and PUBLISHES it on the MCS network.

– Provider returns the INVOKE ACK message to the GS and also publishes an Activity event of type EXECUTION on the Space network.

    - GS receives the event on the Space network and PUBLISHES it on the GS network.

    - Gateway receives the event on the GS network and PUBLISHES it on the MCS network.

– GS receives the INVOKE_ACK message and transmits it to the Gateway.

– Gateway receives the INVOKE_ACK message and transmits it to the Consumer.

– Provider returns the INVOKE RESPONSE message to the GS and also publishes an Activity event of type EXECUTION on the Space network.

    - GS receives the event on the Space network and PUBLISHES it on the GS network.

    - Gateway receives the event on the GS network and PUBLISHES it on the MCS network.

– GS receives the INVOKE_RESPONSE message and transmits it to the Gateway.

– Gateway receives the INVOKE_RESPONSE message and transmits it to the Consumer.

– Consumer receives the INVOKE_RESPONSE message.

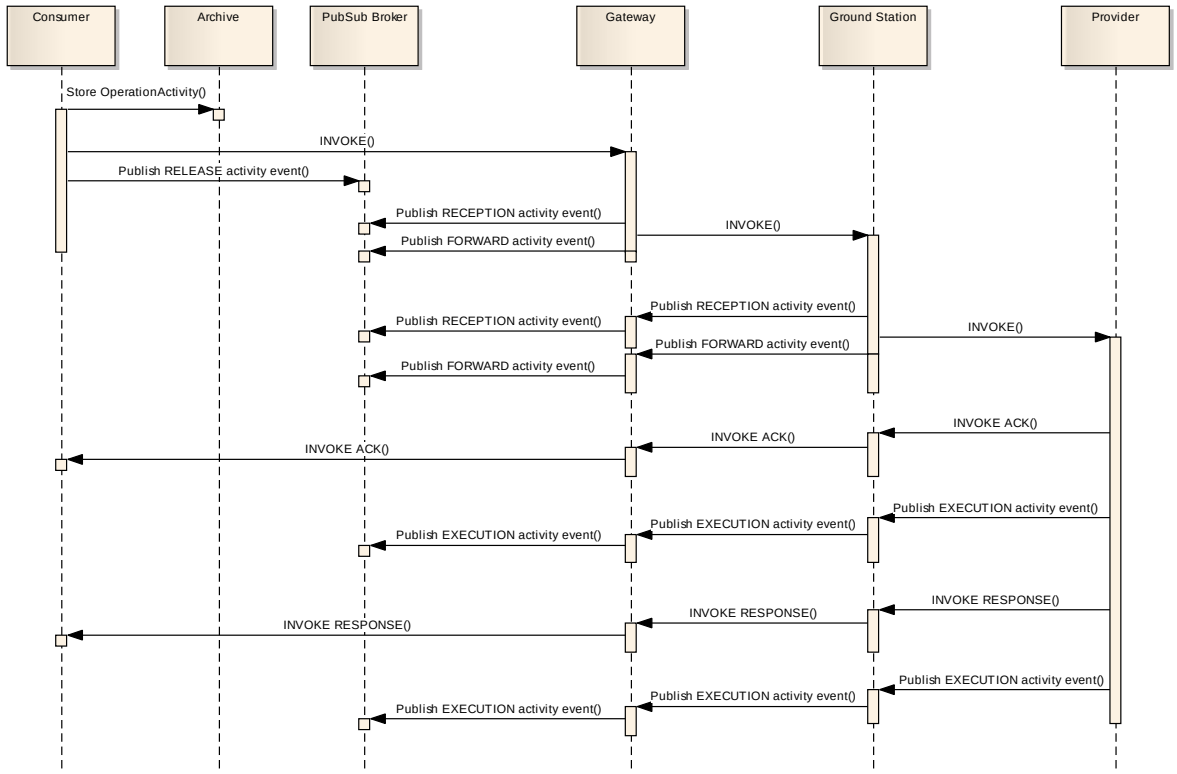

Figure 2-9: Activity Multi-Hop Sequence

NOTE – The activity events in the applications are published to the event PubSub channel as are any errors. The Monitor application receives all of the events through a subscription to the channel on the MCS network. A standard COM archive could be populated by the Monitor application using this mechanism.

2.7 COM SERVICE SPECIFICATIONS

To aid comprehension, several tables are included for the service and each operation definition. The formats are described in sections 2, 4 and 5 of reference [2].

The text below provides an overview of the tables:

- Service overview: A service comprises a set of operations. The tables in the following subsections specify the operations in terms of the Interaction Patterns.
- COM usage: Where a service complies with the COM it must define the COM objects it uses to represent its data and how they should be used. This section lists the objects defined by the service and also lists what other COM objects they may use the related link for.

- COM Event Service usage: If the service uses the COM event service, it must list the events it is going to generate here. If it defines any new events it should also list them in this section. Each event uses a special table to define the various parts of the event.
- COM Archive Service usage: This section details how an implementation of the service should use the COM Archive service. It details, in text, how the objects of the service should be persisted in the archive and also which other parts of an archive should be expected to be present for an implementation of the service to function correctly.
- COM Activity service usage: This section details how an implementation of the service should use the COM Activity service. It details, in text, what is meant by activity for the service being defined, how the objects of the service relate to the events of the activity service, and how the events of the Activity service should be used to provide activity monitoring of the objects of the service.
- Structures: The specification of the service will also detail the structures passed as the message bodies and message returns. If these structures are MAL types they will have the prefix 'MAL::' and be specified in reference [2]; otherwise they are specified in section 4 of this document.
- Errors: The specification of the service will also detail the errors that can be raised over and above the standard set of communications errors defined in the MAL.

3 SPECIFICATION: COM

3.1 GENERAL

This section details the Common Object Model area; the structures used by the service are detailed in section 4. The area and structures are defined in terms of the MO Message Abstraction Layer (MAL), so it is possible to deploy them over any supported protocol and message transport.

3.2 REQUIREMENTS

3.2.1 The instance identifier of an object must not use the value of '0' for actual object instances as this is the wildcard value.

3.2.2 The related field of an object must not use the value of '0' as this is the wildcard value.

3.2.3 The instance identifier of the source field of an object must not use the value of '0' as this is the wildcard value.

3.2.4 If the MAL data type specification is being used the body of an object shall be a concrete type.

3.2.5 If the MAL data type specification is being used the body of an object shall not be an abstract Attribute.

3.2.6 If the MAL data type specification is being used the body of an object shall not be a list.

3.3 SERVICE: EVENT

3.3.1 GENERAL

An event is a specific object representing 'something that happens in the system at a given point in time'. The event service provides a generic mechanism for the distribution of events.

An event is a special type of COM object and is identified by the normal object fields (domain, object type, and object instance identifier).

It may also contain:

- a link to a source object (that caused the event to be generated);
- a link to a service specific related object;
- a body that is specific to the event type and whose data structure is defined by the service that declares the event.

The event service only defines a single operation that supports the publishing of events and subscription of events generated by other components.

Events are published with a MAL update type of DELETION. Events are something that happen at a given point in time; therefore they in themselves have no lifetime. They are created and deleted at the instant in time. For efficiency purposes only the deletion update is used.

Table 3-1: Event Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
COM	Event	2	1	1
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in Replay**	**Capability Set**
PUBLISH-SUBSCRIBE	monitorEvent	1	Yes	1

3.3.2 COM USAGE

3.3.2.1 The event service may be used for the publishing and monitoring of COM events.

3.3.2.2 The identification of the event shall follow the COM object rules.

3.3.2.3 The COM object source link shall point to the object that caused the event to be generated unless stated otherwise by the defining service specification.

3.3.2.4 The COM object related link shall be service and event type specific.

3.3.2.5 When an event is published using the PUBLISH-SUBSCRIBE monitorEvent operation defined in the event service it shall be published with a MAL update type of DELETED.

3.3.2.6 When an event is published, the COM object instance identifier shall be populated by the publisher.

3.3.2.7 The object instance identifier should be unique for a domain and object type.

3.3.2.8 An event may have an associated body that is published with it.

3.3.2.9 The type and contents of the body of an event shall be specified in the defining service specification but must be a concrete type or an Attribute.

3.3.3 COM ARCHIVE SERVICE USAGE

3.3.3.1 Each type of event is a separate COM object type and therefore may be stored in a COM archive.

3.3.3.2 When a service implementation requires that the events it generates be persisted, upon event emission, the event shall be stored as normal.

3.3.3.3 The related link is service specific and shall always be null if not explicitly used by a service.

3.3.3.4 The source link shall point to the source object of the event.

3.3.4 OPERATION: MONITOREVENT

3.3.4.1 Overview

The monitorEvent operation allows a consumer to subscribe for events.

Operation Identifier	monitorEvent	
Interaction Pattern	PUBLISH-SUBSCRIBE	
Pattern Sequence	Message	Body Type
OUT	PUBLISH/NOTIFY	ObjectDetails MAL::Element

3.3.4.2 Structures

3.3.4.2.1 The MAL::EntityKey.firstSubKey shall contain the event object number as a base 10 string. For example, for an object number of ‘14’ the key value would be ‘14’ with no padding.

3.3.4.2.2 The MAL::EntityKey.secondSubKey shall contain the area, service, and version ObjectType fields as a MAL::Long populated as (in hex) 0xAAAASSSSVVXXXXXX where AAAA is the area (16 bits), SSSS is the service (16 bits), VV is the version (8 bits), and XXXXXX is unused and set to zero (24 bits). For example, for an area of ‘1’, and a service of ‘2’, and a version of ‘3’ the field would contain (in hex) 0x0001000203000000.

3.3.4.2.3 The MAL::EntityKey.thirdSubKey shall contain the event object instance identifier.

3.3.4.2.4 The MAL::EntityKey.fourthSubKey shall contain the area, service, version and number fields of the event source ObjectType using same methodology as given for the second sub key but replacing the XXXXXX part with the number field.

3.3.4.2.5 The related and source links of the event shall populate the ObjectDetails part of the publish/notify message.

3.3.4.2.6 The body of the event shall populate the final part of the publish/notify message.

3.3.4.2.7 The timestamp of the Event shall be taken from the publish message.

3.3.4.2.8 An event in one domain may be generated by something in another domain; therefore the domain of the event shall not be required to be the same domain as the source of the event.

3.3.4.3 Errors

The operation does not return any errors.

3.4 SERVICE: ARCHIVE

3.4.1 OVERVIEW

The Archive service provides a basic interface to a standard archiving function. It follows the basic CRUD principles and allows simple querying of the archive. It provides operations to add new objects to an archive, delete objects from an archive, update existing objects in an archive, and also query the content of the archive.

The query operation provides a basic querying ability, allowing a consumer to filter on fields from the object headers (such as domain, etc.) and also filter on the body of the object if it uses the MAL data type specification.

The query operation is extensible but the extensions would be outside this standard.

Finally, a consumer of the archive can monitor it for changes by subscribing for archive events from the event service. Any change to the archive is published using the event service if it is supported by an implementation.

Table 3-2: Archive Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
COM	Archive	2	2	1
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in Replay**	**Capability Set**
INVOKE	retrieve	1	Yes	1
PROGRESS	query	2	Yes	
INVOKE	count	3	Yes	
REQUEST	store	4	No	2
SUBMIT	update	5	No	3
REQUEST	delete	6	No	4

3.4.2 COM EVENT SERVICE USAGE

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

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

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

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

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

Table 3-3: Archive Service Events

Event Name	Object Number	Object Body Type	Related points to	Source points to
ObjectStored	1	Not used	Not specified	The Object source is the newly stored object.
ObjectUpdated	2	Not used	Not specified	The Object source is the updated object.
ObjectDeleted	3	Not used	Not specified	The Object source is the deleted object.

3.4.3 OPERATION: RETRIEVE

3.4.3.1 Overview

The retrieve operation retrieves a set of objects identified by their object instance identifier.

Operation Identifier	retrieve	
Interaction Pattern	INVOKE	
Pattern Sequence	Message	Body Type
IN	INVOKE	ObjectType List<MAL::Identifier> List<MAL::Long>
OUT	ACK	
OUT	RESPONSE	List<ArchiveDetails> List<MAL::Element>

3.4.3.2 Structures

3.4.3.2.1 The first part of the request shall contain the type of the object required.

3.4.3.2.2 If any of the fields of the object type contains the wildcard value of '0' then an INVALID error shall be returned.

3.4.3.2.3 The second part of the request shall contain the domain to match.

3.4.3.2.4 If the domain contains the wildcard value of '*' then an INVALID error shall be returned.

3.4.3.2.5 The third part of the request shall contain the list of object instance identifiers to match.

3.4.3.2.6 If the object instance identifier list contains the wildcard value '0' then all object instances shall be matched.

3.4.3.2.7 If any explicitly requested object cannot be matched then an UNKNOWN error shall be returned.

3.4.3.2.8 The response shall contain the set of matched objects.

3.4.3.2.9 The first returned list shall contain the matched object instance identifiers and object details of the matched objects.

3.4.3.2.10 The second returned list shall contain the object bodies ordered identically to the first list unless no body for the object is declared in the service specification, in which case a NULL replaces the complete list.

3.4.3.2.11 There shall be an entry in each returned list for each matched object.

3.4.3.2.12 When no objects have been matched only a response with NULL for each part of the response shall be returned.

3.4.3.2.13 The ordering of the returned objects is not specified and implementation specific.

3.4.3.2.14 If ordering of the returned objects is required then the query operation should be used instead.

3.4.3.3 Errors

The operation may return the following errors:

a) ERROR: UNKNOWN

    1) One or more of the requested objects specified in the operation do not exist and therefore cannot be found.

2) The indexes of the error values shall be contained in the extra information field.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	List<MAL::UInteger>

b) ERROR: INVALID—The request contains a wildcard value in either the object type field or the domain.

Error	Error #	ExtraInfo Type
INVALID	70000	Not Used

3.4.4 OPERATION: QUERY

3.4.4.1 Overview

The query operation retrieves a set of object instance identifiers, and optionally the object bodies, from a list of supplied queries. The PROGRESS interaction pattern is used as the returned set of data may be quite large and this allows it to be split over several MAL messages.

Operation Identifier	query	
Interaction Pattern	PROGRESS	
Pattern Sequence	Message	Body Type
IN	PROGRESS	MAL::Boolean ObjectType List<ArchiveQuery> List<QueryFilter>
OUT	ACK	
OUT	UPDATE	ObjectType List<MAL::Identifier> List<ArchiveDetails> List<MAL::Element>
OUT	RESPONSE	ObjectType List<MAL::Identifier> List<ArchiveDetails> List<MAL::Element>

3.4.4.2 Structures

3.4.4.2.1 The first part of the request shall contain a Boolean that if set to TRUE requests that the body of the objects is returned otherwise only the ObjectType and ArchiveDetails shall be returned and the returned list of the bodies of the objects shall be replaced with a NULL.

3.4.4.2.2 The second part of the request shall contain the type of the object required.

3.4.4.2.3 Each part of the object type may contain the wildcard value of '0'.

3.4.4.2.4 The third and fourth parts of the request shall contain the queries to evaluate.

3.4.4.2.5 A single query shall be formed by the combination of an ArchiveQuery from the first list and a QueryFilter from the second list.

3.4.4.2.6 The two lists shall be ordered identically so that the query and the filter parts can be matched together.

3.4.4.2.7 If a query does not contain a QueryFilter part then that entry in the QueryFilter list shall be replaced with a NULL value.

3.4.4.2.8 If the request does not contain any QueryFilters then the complete list may be replaced with a NULL.

3.4.4.2.9 The size of the two lists must be the same unless the complete second list is replaced with a NULL otherwise an INVALID error shall be raised.

3.4.4.2.10 For each query, the ArchiveQuery and the QueryFilter shall contain the COM object fields to filter on.

3.4.4.2.11 The ArchiveQuery may contain the wildcard value of NULL on each of the fields.

3.4.4.2.12 If an ArchiveQuery contains an end time but no start time then it shall match the single object that has a timestamp closest to, but not greater than, the end time field.

3.4.4.2.13 The end time field may specify a time in the future.

3.4.4.2.14 If the sortFieldName of the ArchiveQuery does not reference a defined field then an INVALID error shall be returned.

3.4.4.2.15 Each query shall be evaluated separately from each other, the filter of one query will not affect the filter of another. This forms a logical OR operation.

3.4.4.2.16 If the QueryFilter contains an error then an INVALID error shall be returned. The definition of erroneous values are filter specific and defined in the relevant filter structure specification.

3.4.4.2.17 The updates and the responses shall contain the set of matched objects.

3.4.4.2.18 If a wildcard was used in the ObjectType part of the request then the updates and response shall contain the ObjectType of each matched object.

3.4.4.2.19 If there was not any wildcards in the ObjectType part of the request the ObjectType in the updates and response shall be replaced by a NULL.

3.4.4.2.20 The first returned list shall contain the domain of the objects being returned.

3.4.4.2.21 If multiple ObjectTypes or domains have been matched then multiple Update message may be returned.

3.4.4.2.22 There shall be an entry in the second and third lists for each matched object.

3.4.4.2.23 The second returned list shall contain the archive details stored for the matched objects.

3.4.4.2.24 If the initial Boolean of the request was True the third returned list shall contain the bodies of the objects.

3.4.4.2.25 If the initial Boolean of the request was NULL or False the third returned list shall be replaced by a NULL.

3.4.4.2.26 The returned lists shall be sorted based on the sorting options specified in ArchiveQuery.

3.4.4.2.27 Each domain/object type pair shall be sorted separately from other domain/object type pairs; there is no requirement for sorting to be applied across domain/object type pairs.

3.4.4.2.28 When the field being sorted on contains a NULL value, or does not exist in the matched object (because of a containing composite being NULL), these entries shall be added to the end of the returned list in the order that they are matched.

3.4.4.2.29 When no objects have been matched only a response with NULL for each part of the response shall be returned.

3.4.4.3 Errors

The operation may return the following error:

ERROR: INVALID

a) One or more of the query filters supplied contains an invalid value.

b) The extra information field contains the indexes of the erroneous values from the originating list supplied.

Error	Error #	ExtraInfo Type
INVALID	70000	List<MAL::UInteger>

3.4.5 OPERATION: COUNT

3.4.5.1 Overview

The count operation counts the set of objects based on a supplied query.

Operation Identifier	count	
Interaction Pattern	INVOKE	
Pattern Sequence	Message	Body Type
IN	INVOKE	ObjectType List<ArchiveQuery> List<QueryFilter>
OUT	ACK	
OUT	RESPONSE	List<MAL::Long>

3.4.5.2 Structures

3.4.5.2.1 The ObjectType, ArchiveQuery, and QueryFilter parts of the request shall be populated exactly the same as for the query operation.

3.4.5.2.2 The response shall contain the count of matched objects.

3.4.5.2.3 There shall be an entry in each returned list for each entry in the request list.

3.4.5.2.4 The returned lists shall be ordered the same as the request query lists so that the response can be matched to the corresponding request.

3.4.5.3 Errors

The operation may return the following error:

ERROR: INVALID

a) One or more of the query filters supplied contains an invalid value.

b) The extra information field contains the indexes of the erroneous values from the originating list supplied.

Error	Error #	ExtraInfo Type
INVALID	70000	List<MAL::UInteger>

3.4.6 OPERATION: STORE

3.4.6.1 Overview

The store operation stores new objects in the archive and causes an ObjectStored event to be published by the archive.

When new objects are being stored in an archive by a service provider the archive service provider is capable of allocating an unused object instance identifier for the objects being stored. The returned object instance identifier should be used by the service provider for identifying the object instances to its consumer to ensure that only a single object instance identifier is used for each object instance.

Operation Identifier	store	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	MAL::Boolean ObjectType List<MAL::Identifier> List<ArchiveDetails> List<MAL::Element>
OUT	RESPONSE	List<MAL::Long>

3.4.6.2 Structures

3.4.6.2.1 The first part of the request indicates whether the operation should return the object instance identifiers used; if TRUE it shall return them; otherwise it shall return NULL.

3.4.6.2.2 The second part of the request shall contain the type of object being stored.

3.4.6.2.3 The third part of the request shall contain the domain of the objects being stored.

3.4.6.2.4 The fourth part of the request shall contain the list of archive details to use, one for each object being stored.

3.4.6.2.5 If the object instance identifier supplied in the archive details is set to 0 then the store operation shall allocate a new and unused object instance identifier.

3.4.6.2.6 If the object instance identifier supplied in the archive details is not set to 0 and is currently used in the archive then a DUPLICATE error is returned and no objects from the request shall be stored.

3.4.6.2.7 The fifth part of the request shall contain the list of objects to store.

3.4.6.2.8 The fourth and fifth list must be the same size as there is only entry in each for each object to be stored. If they differ in size and INVALID error is returned with the extra error information integer giving the index of the list entry without a matching entry in the other list.

3.4.6.2.9 An INVALID error shall be returned if a wildcard value of '0' is used in the object type.

3.4.6.2.10 An INVALID error shall be returned if a wildcard value of '*' is used in the domain identifier list.

3.4.6.2.11 An INVALID error shall be returned if the values of '0', '*', or NULL are used in the network, timestamp, or provider fields of the archive details except for the object instance identifier.

3.4.6.2.12 The type of the body of the object should be checked against the declared type in the relevant service specification, if different an INVALID error is raised.

3.4.6.2.13 If any error is returned then the store operation shall be rolled back and nothing is stored as a result of the operation.

3.4.6.2.14 The response shall contain the set of new object instance identifiers if the request supplied an initial TRUE Boolean value; otherwise it shall return NULL.

3.4.6.2.15 The returned list shall be ordered identically to the submitted list so that the returned object instance identifiers can be mapped to the correct objects.

3.4.6.3 Errors

The operation may return the following errors:

a) ERROR: INVALID

    1) One or more of the objects being stored contains an invalid value.

2) The extra information field contains the indexes of the erroneous values from the originating list supplied.

Error	Error #	ExtraInfo Type
INVALID	70000	List<MAL::UInteger>

b) ERROR: DUPLICATE

1) One or more of the objects being stored has supplied an object instance identifier that is already in use in the archive.

2) The extra information field contains the indexes of the erroneous values from the originating request list.

Error	Error #	ExtraInfo Type
DUPLICATE	70001	List<MAL::UInteger>

3.4.7 OPERATION: UPDATE

3.4.7.1 Overview

The update operation updates an object (or set of objects) and causes an ObjectUpdated event to be published by the archive.

Operation Identifier	update	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	ObjectType List<MAL::Identifier> List<ArchiveDetails> List<MAL::Element>

3.4.7.2 Structures

3.4.7.2.1 The first part of the request shall contain the type of object being updated.

3.4.7.2.2 The second part of the request shall contain the domain of the objects being updated.

3.4.7.2.3 The third part of the request shall contain the list of ArchiveDetails.

3.4.7.2.4 The object instance identifier contained in the ArchiveDetails, combined with the object type and domain from the request, shall be used to match objects.

3.4.7.2.5 If requested object cannot be matched then an UNKNOWN error shall be returned and nothing will be updated.

3.4.7.2.6 The remainder of the ArchiveDetails shall be used to update the matched objects.

3.4.7.2.7 The fourth part of the request shall contain the list of objects to replace the matched objects with.

3.4.7.2.8 No wildcard values shall be accepted in the object type, the domain, and the object instance identifier; an INVALID error is returned in this case and no objects are updated.

3.4.7.3 Errors

The operation may return the following errors:

a) ERROR: UNKNOWN

1) One or more of the requested objects specified in the operation do not exist and therefore cannot be found.

2) The indexes of the error values shall be contained in the extra information field.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	List<MAL::UInteger>

b) ERROR: INVALID

1) One or more of the objects being updated contains a wildcard value in the object identifier fields.

2) The extra information field contains the indexes of the erroneous values from the originating list supplied.

Error	Error #	ExtraInfo Type
INVALID	70000	List<MAL::UInteger>

3.4.8 OPERATION: DELETE

3.4.8.1 Overview

The delete operation deletes an object (or set of objects) and causes an ObjectDeleted event to be published by the archive.

Operation Identifier	delete	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	ObjectType List<MAL::Identifier> List<MAL::Long>
OUT	RESPONSE	List<MAL::Long>

3.4.8.2 Structures

3.4.8.2.1 The first part of the request shall contain the type of object to match and is not permitted to contain the wildcard value.

3.4.8.2.2 The second part of the request shall contain the domain of the objects to match and is not permitted to contain the wildcard value.

3.4.8.2.3 If either the first or second part contain a wildcard value then an INVALID error shall be returned and no object deleted.

3.4.8.2.4 The third part of the request shall contain the list of object instance identifiers to match.

3.4.8.2.5 If the object instance identifier list contains the wildcard value ‘0’ then all object instances shall be matched.

3.4.8.2.6 If any explicitly requested object cannot be matched then an UNKNOWN error shall be returned and nothing will be deleted.

3.4.8.2.7 The matched objects shall be deleted from the archive.

3.4.8.2.8 The response shall contain the set of object instance identifiers of the deleted objects.

3.4.8.3 Errors

The operation may return the following errors:

a) ERROR: UNKNOWN

1) One or more of the requested objects specified in the operation do not exist and therefore cannot be found.

2) The indexes of the error values shall be contained in the extra information field.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	List<MAL::UInteger>

b) ERROR: INVALID—The supplied object type or domain contains a wildcard value.

Error	Error #	ExtraInfo Type
INVALID	70000	Not Used

3.5 SERVICE: ACTIVITYTRACKING

3.5.1 GENERAL

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

Table 3-4: ActivityTracking Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
COM	ActivityTracking	2	3	1
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in Replay**	**Capability Set**

3.5.2 COM USAGE

3.5.2.1 The activity tracking service may be used for the monitoring of MAL operations.

3.5.2.2 A MAL implementation must populate the transaction identifier for SEND interactions if the activity tracking service is to be used.

3.5.2.3 If a MAL operation is using the activity tracking service, when the interaction is started, the consumer shall populate and archive an OperationActivity object.

3.5.2.4 The OperationActivity object shall use the transaction identifier of the MAL operation for its object instance identifier.

3.5.2.5 The OperationActivity object shall use the domain of the MAL operation for its domain.

3.5.2.6 All Activity events for the interaction shall link to the OperationActivity object using their source link where the object instance identifier to use is the MAL interaction transaction identifier.

3.5.2.7 If a relay node is unable to relay the operation then that relay shall fail the MAL interaction that the consumer initiated as well as generate the correct activity event.

3.5.2.8 The executionStage field of the Execution event structure shall hold the current execution stage of the activity.

3.5.2.9 For SUBMIT, INVOKE, and PROGRESS interactions, the ACKNOWLEDGE stage shall have an execution stage value of '1'.

3.5.2.10 For REQUEST interactions the RESPONSE stage shall have an execution stage value of '1'.

3.5.2.11 For PROGRESS interactions the execution stage value for the PROGRESS updates shall start at '2' and increate for each update sent.

3.5.2.12 For PROGRESS interactions the execution stage value for the RESPONSE shall be the total number of PROGRESS updates plus '2'.

3.5.2.13 The stageCount field of the Execution event structure shall be populated from the stage count row of table 2-1.

Table 3-5: ActivityTracking Service Object Types

Object Name	Object Number	Object Body Type	Related points to
OperationActivity	6	OperationActivity	Not used

3.5.3 COM EVENT SERVICE USAGE

3.5.3.1 The activity tracking service shall use the event service for the monitoring of activities being transferred from a consumer to a provider and also for execution in the provider.

3.5.3.2 A Release event shall be generated when an activity is released from a consumer.

3.5.3.3 A Reception event shall be generated when an activity is received by an intermediate relay.

3.5.3.4 A Forward event shall be generated when an activity is released from an intermediate relay.

3.5.3.5 An Acceptance event shall be generated when an activity is received by the destination provider.

3.5.3.6 Reception and Forward events shall only be used in a multi-hop activity transfer.

3.5.3.7 For Reception and Forward events, the source URI in the UpdateHeader of the published event shall contain the URI of the relay. It is protocol specific how this value is derived.

3.5.3.8 Execution progress of an activity shall be reported using Execution events.

3.5.3.9 The Execution event may be reported many times as it is used to report each stage in the execution of the activity.

Table 3-6: ActivityTracking Service Events

Event Name	Object Number	Object Body Type	Related points to	Source points to
Release	1	ActivityTransfer	Not specified	The activity being monitored is the source of the event.
Reception	2	ActivityTransfer	Not specified	The activity being monitored is the source of the event.
Forward	3	ActivityTransfer	Not specified	The activity being monitored is the source of the event.
Acceptance	4	ActivityAcceptance	Not specified	The activity being monitored is the source of the event.
Execution	5	ActivityExecution	Not specified	The activity being monitored is the source of the event.

3.5.4 COM ARCHIVE SERVICE USAGE

3.5.4.1 The events generated as part of the activity monitoring pattern may be persisted in a COM archive.

3.5.4.2 If the activity events are being persisted then the objects that represent the activities (such as an OperationActivity object) should also be persisted.

4 DATA TYPES

4.1 AREA DATA TYPES: COM

4.1.1 COMPOSITE: OBJECTTYPE

The ObjectType structure uniquely identifies the type of an object. It is the combination of the area number, service number, area version, and service object type number. The combined parts are able to fit inside a MAL::Long (for implementations that prefer to index on a single numeric field rather than a structure).

Name	ObjectType		
Extends	MAL::Composite		
Short Form Part	1		
Field	Type	Nullable	Comment
area	MAL::UShort	No	Area Number where the object type is defined. Must not be ‘0’ for values as this is the wildcard.
service	MAL::UShort	No	Service Number of the service where the object type is defined. Must not be ‘0’ for values as this is the wildcard.
version	MAL::UOctet	No	Area Version of the service where the object type is defined. Must not be ‘0’ for values as this is the wildcard.
number	MAL::UShort	No	The service specific object number. Must not be ‘0’ for values as this is the wildcard.

4.1.2 COMPOSITE: OBJECTKEY

The ObjectKey structure combines a domain and an object instance identifier such that it identifies the instance of an object for a specific domain.

Name	ObjectKey		
Extends	MAL::Composite		
Short Form Part	2		
Field	Type	Nullable	Comment
domain	List<MAL::Identifier>	No	The domain of the object instance.
instId	MAL::Long	No	The unique identifier of the object instance. Must not be ‘0’ for values as this is the wildcard.

4.1.3 COMPOSITE: OBJECTID

The ObjectId structure combines an object type and an object key such that it identifies the instance and type of an object for a specific domain.

Name	ObjectId		
Extends	MAL::Composite		
Short Form Part	3		
Field	Type	Nullable	Comment
type	ObjectType	No	The fully qualified unique identifier of the type.
key	ObjectKey	No	The combination of the object domain and object instance identifier.

4.1.4 COMPOSITE: OBJECTDETAILS

The ObjectDetails type is used to hold the extra information associated with an object instance, namely the related and source links.

Name	ObjectDetails		
Extends	MAL::Composite		
Short Form Part	4		
Field	Type	Nullable	Comment
related	MAL::Long	Yes	Contains the object instance identifier of a related object (e.g., the ActionDefinition that an Action uses). This is service specific. The ObjectType of the related object is specified in the service specification. The related object must exist in the same domain as this object.
source	ObjectId	Yes	An object which is at the origin of the object creation (e.g., the procedure from which an action was triggered).

4.1.5 COMPOSITE: INSTANCEBOOLEANPAIR

InstanceBooleanPair is a simple pair of an object instance identifier and a Boolean value.

Name	InstanceBooleanPair		
Extends	MAL::Composite		
Short Form Part	5		
Field	Type	Nullable	Comment
id	MAL::Long	No	The object instance identifier.
value	MAL::Boolean	No	An associated Boolean value.

4.2 SERVICE DATA TYPES: ARCHIVE

4.2.1 ENUMERATION: EXPRESSIONOPERATOR

The ExpressionOperator enumeration holds a set of possible expression operators.

Name	ExpressionOperator	
Short Form Part	5	
Enumeration Value	Numerical Value	Comment
EQUAL	1	Checks for equality.
DIFFER	2	Checks for difference (not equal).
GREATER	3	Checks for greater than.
GREATER_OR_EQUAL	4	Checks for greater than or equal to.
LESS	5	Checks for less than.
LESS_OR_EQUAL	6	Checks for less than or equal to.
CONTAINS	7	Case sensitive containment test (String types only)
ICONTAINS	8	Case insensitive containment test (String types only).

4.2.2 COMPOSITE: QUERYFILTER

QueryFilter is the base structure for archive filters.

Name	QueryFilter
Extends	MAL::Composite
Abstract	

4.2.3 COMPOSITE: ARCHIVEDETAILS

The ArchiveDetails structure is used to hold information about a single entry in an Archive.

Name	ArchiveDetails		
Extends	MAL::Composite		
Short Form Part	1		
Field	**Type**	**Nullable**	**Comment**
instId	MAL::Long	No	The object instance identifier of the archived object.
details	ObjectDetails	No	The details of the Object.
network	MAL::Identifier	Yes	The network zone of the object.
timestamp	MAL::FineTime	Yes	The time at the object creation.
provider	MAL::URI	Yes	The component that created the object (a component may be anything from an onboard equipment to a software process on the ground).

4.2.4 COMPOSITE: ARCHIVEQUERY

The ArchiveQuery structure is used to specify filters on the common parts of an object in an archive.

Name	ArchiveQuery		
Extends	MAL::Composite		
Short Form Part	2		
Field	**Type**	**Nullable**	**Comment**
domain	List<MAL::Identifier>	Yes	Only the objects whose domain matches the provided domain will be returned. The domain field supports the wildcard value of '*' only in the last part of the domain. If NULL then all domains shall be matched.
network	MAL::Identifier	Yes	Optional network zone. Only the objects whose network zone matches the provided value will be returned. If NULL then all values will be matched.
provider	MAL::URI	Yes	Optional provider. Only the objects whose provider URI matches the provided provider URI will be returned. If NULL then all values will be matched.
related	MAL::Long	No	Object instance identifier of the related Object. Only the objects whose related Object matches the provided related Object will be returned unless the '0' wildcard value is supplied.
source	ObjectId	Yes	Optional source. Only the objects whose source matches the provided source will be returned. The fields of the ObjectId may contain the wildcard values ('0' for numeric fields; '*' for Identifier fields). If NULL then all values will be matched.

startTime	MAL::FineTime	Yes	Optional start time. Only the objects whose timestamp is equal or greater than the provided start time will be returned. If NULL then no start time shall be applied.
endTime	MAL::FineTime	Yes	Optional end time. Only the objects whose timestamp is equal or less than the provided end time will be returned. If NULL then no end time will be applied.
sortOrder	MAL::Boolean	Yes	If set to TRUE then returned values shall be sorted in ascending order; if FALSE then in descending order. If NULL then no sorting shall be applied.
sortFieldName	MAL::String	Yes	If the returned values are to be sorted because the sortOrder field is not NULL then this field may contain the name of the field in the object body (MAL::Composite) to sort against. If the object body is not a composite but an Attribute or Enumeration then to sort on the values an empty string of ‘’ should be used. Enumerations are sorted on their ordinal. If this field is NULL then the objects shall be sorted on the COM object timestamp. The field follows the naming convention of CompositeFilter::fieldName. If the field points to a composite, list, abstract type (including Attribute), or Blob, then no sorting shall be applied.

4.2.5 COMPOSITE: COMPOSITEFILTER

The CompositeFilter allows an archive query to specify a filter based on the content of the body of an object if that body is specified using the MAL data type specification.

Name	CompositeFilter		
Extends	Composite		
Short Form Part	3		
Field	**Type**	**Nullable**	**Comment**
fieldName	MAL::String	No	The name of the field in the object body (MAL::Composite) to match against. It is specified by referring to the name of the field. If the field does not exist in the Composite then the filter shall evaluate to false. If the field is nested inside another Composite, it can be referenced by separating the field names by a '.' character; for example the field 'instId' or the source of an ArchiveDetails composite would be referenced using 'details.source.key.instId'. If the body is not a Composite but an Attribute or Enumeration then it can be referred to by using a blank ('') string value here. Accessing values from lists is not supported by this method.
type	ExpressionOperator	No	The type of the filter to apply.

fieldValue	MAL::Attribute	Yes	The value to compare with. The type of the supplied value and the matched field must be the same. Must not contain NULL for expression operators CONTAINS, ICONTAINS, GREATER, GREATER_OR_EQUAL, LESS, or LESS_OR_EQUAL; otherwise an INVALID error should be returned. Must contain a UInteger ordinal value if the field being matched is an Enumeration. Blob fields can only be used with EQUAL/DIFFER. Must contain a String value if operator is CONTAINS or ICONTAINS otherwise an INVALID error should be returned.

4.2.6 COMPOSITE: COMPOSITEFILTERSET

The CompositeFilterSet contains a list of CompositeFilters that are ANDed together to form a more complex filter.

Name	CompositeFilterSet		
Extends	QueryFilter		
Short Form Part	4		
Field	**Type**	**Nullable**	**Comment**
filters	List<Composite Filter>	No	The list of filters to apply.

4.3 SERVICE DATA TYPES: ACTIVITYTRACKING

4.3.1 COMPOSITE: ACTIVITYTRANSFER

The ActivityTransfer structure holds details for a Release, Reception, or Forward event of an activity.

Name	ActivityTransfer		
Extends	Composite		
Short Form Part	1		
Field	**Type**	**Nullable**	**Comment**
success	MAL::Boolean	No	The success result of this stage: TRUE if successful, FALSE otherwise. If this is TRUE then the next two fields should be populated.
estimateDuration	MAL::Duration	Yes	The estimated amount of time it will take to be received by the next destination if Release or Forward. Contains the estimated amount of time it will take to be forwarded by this relay if Reception event. May be NULL if unknown or cannot be calculated.
nextDestination	MAL::URI	Yes	This contains the URI of the next destination, either another relay or the provider. It is protocol specific how this value is derived.

4.3.2 COMPOSITE: ACTIVITYACCEPTANCE

The ActivityAcceptance structure is used to hold details of an Acceptance event.

Name	ActivityAcceptance		
Extends	Composite		
Short Form Part	2		
Field	**Type**	**Nullable**	**Comment**
success	MAL::Boolean	No	The success result of this stage: TRUE if successful, FALSE otherwise.

4.3.3 COMPOSITE: ACTIVITYEXECUTION

The ActivityExecution structure is used to report the execution status of an activity in the final destination.

Name	ActivityExecution		
Extends	Composite		
Short Form Part	3		
Field	Type	Nullable	Comment
success	MAL::Boolean	No	The success result of this stage: TRUE if successful, FALSE otherwise.
executionStage	MAL::UInteger	No	The execution stage of the operation.
stageCount	MAL::UInteger	No	The total number of execution stages that will be reported.

4.3.4 COMPOSITE: OPERATIONACTIVITY

The OperationActivity structure contains the details of a MAL operation activity.

Name	OperationActivity		
Extends	Composite		
Short Form Part	4		
Field	Type	Nullable	Comment
interactionType	MAL::InteractionType	No	The interaction type of the original operation message header.

5 ERROR CODES

Table 5-1 lists the errors defined in this specification.

Table 5-1: COM Error Codes

Error	Error #	Comment
INVALID	70000	Operation specific
DUPLICATE	70001	Operation specific

6 SERVICE SPECIFICATION XML

This specification defines a normative XML schema for validating MO COM service specifications and is an extension of the MAL XML schema (see reference [2]). The use of XML for service specification provides a machine readable format rather than the text based document format. The published specifications and XML schemas are held in an online SANA registry, located:

http://sanaregistry.org/r/moschemas/

The XML Schema that is used to validate the actual XML COM service specifications is located:

http://sanaregistry.org/r/moschemas/COMSchema-v.v.xsd

The normative XML for this specification, validated against the XML schemas, is located:

http://sanaregistry.org/r/moschemas/ServiceDefCOM-v.v.xml

Where the ‘v.v’ part is replaced with the issue number of the corresponding document.

The latest version of any specification shall always be directly available by removing the ‘-v.v’ part from the address, for example:

http://sanaregistry.org/r/moschemas/ServiceDefCOM.xml

ANNEX A

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

A1 SECURITY CONSIDERATIONS

The security considerations of this specification are the same as those of reference [2].

A2 SANA CONSIDERATIONS

The recommendations of this document request SANA populate the registry specified in reference [2] with the schema and XML detailed in section 6 of this document.

As stated in reference [2], the registration rule for change to this registry requires an engineering review by a designated expert. The expert shall be assigned by the WG Chair, or in absence, Area Director.

A3 PATENT CONSIDERATIONS

The recommendations of this document have no patent issues.

ANNEX B

DEFINITION OF ACRONYMS

(INFORMATIVE)

API	Application Programming Interface
CLTU	Command Link Transmission Unit
DTN	Delay/Disruption Tolerant Networking
GS	Ground Station
IP	Interaction Pattern
MAL	Message Abstract Layer
MCS	Mission Control System
MO	Mission Operations
PDU	Protocol Data Unit
SANA	Space Assigned Numbers Authority
SLE	CCSDS Space Link Extension
XML	eXtensible Markup Language
URI	Universal Resource Identifier

ANNEX C

INFORMATIVE REFERENCES

(INFORMATIVE)

[C1] *Mission Operations Services Concept*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 520.0-G-3. Washington, D.C.: CCSDS, December 2010.

NOTE – Normative references are listed in 1.6.