The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

MISSION OPERATIONS— COMMON SERVICES

RECOMMENDED STANDARD

CCSDS 522.0-B-1

BLUE BOOK

May 2020

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

MISSION OPERATIONS— COMMON SERVICES

RECOMMENDED STANDARD

CCSDS 522.0-B-1

BLUE BOOK

May 2020

AUTHORITY

Issue:	Recommended Standard, Issue 1
Date:	May 2020
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-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the email address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA

Email: secretariat@mailman.ccsds.org

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 five 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 has processes for identifying patent issues and for securing from the patent holder agreement that all licensing policies are reasonable and non-discriminatory. However, CCSDS does not have a patent law staff, and 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-4). 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 sent to the CCSDS Secretariat at the email 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.
- China 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.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- 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.
- Hellenic Space Agency (HSA)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
- 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.
- Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- 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 522.0-B-1	Mission Operations—Common Services, Recommended Standard, Issue 1	May 2020	Original issue

CONTENTS

CONTENTS (continued)

Section Page

Figure

Table

1 INTRODUCTION

1.1 GENERAL

This Recommended Standard defines the Mission Operations (MO) Common Services in conformance with the service framework specified in subsection 4.6 of reference [E1], *Mission Operations Services Concept*.

The Common Services are a set of services that provide support facilities to the Mission Operation services.

These services are defined in terms of the Common Object Model (COM) (reference [3]) and the Message Abstraction Layer (MAL) (reference [2]).

1.2 PURPOSE AND SCOPE

This Recommended Standard defines, in an abstract manner, the Common services 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; and

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; or

c) the methods or technologies required for communications.

1.3 APPLICABILITY

This specification is applicable to any MO compliant system component that utilises the MO infrastructure for service discovery. Nominally, but not exclusively, this applies within ground control systems, from ground control systems to external entities and between external entities that wish to use the MO infrastructure for service discovery.

1.4 RATIONALE

The primary goal of CCSDS is to increase the level of interoperability among agencies. This Recommended Standard furthers that goal by providing a standard service specification for the discovery of services and also common infrastructure services within the context of MO deployments.

1.5 DOCUMENT STRUCTURE

This Recommended Standard is organised as follows:

a) section 1 provides purpose and scope, applicability, and rationale of this Recommended Standard and lists the definitions, conventions, and references used throughout the document;

b) section 2 presents an overview of the concepts;

c) section 3 presents the Common Services specification;

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

e) section 5 specifies the internet location of the formal service specification eXtensible Markup Language (XML).

1.6 DEFINITION OF TERMS

The following terms are defined in reference [1]; they are repeated here for convenience:

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.

domain: A logical namespace that the entities of a service belong to. It is used to scope the frame of reference of monitoring and control.

NOTE – 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.

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 components are made. Any given port on the hardware component may expose one or more service interfaces.

management service interface: A service interface that exposes management functions of a service capability contained in a component for use by service consumers.

network zone: A subdivision of a network, distinct from domain, indicating to a consumer how ‘local’ a service provider is.

provided service interface: A service interface that exposes the service capability 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.

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 capability set: A grouping of service operations. 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. The capability sets define a grouping of the service operations that remains sensible and coherent; they also provide a service provider with an ability to communicate to a consumer its capability.

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 directory: An entity that provides publish and lookup facilities to service providers and consumers.

NOTE – Strictly speaking, a directory is not required if a well-known service is to be used; however, in most circumstances, a directory provides required flexibility in the location of services. Service location can be statically configured, dynamically discovered through a service directory, or a combination of the two; this is an implementation choice. The service directory is itself, by definition, a service.

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.

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’.

service provider (provider): A component that offers a service to another by means of one of its provided service interfaces.

session: In the context of MO, a coherent data source; for example, the operational system in live real-time, or the operational system in replay, or a simulation of the operational system. Multiple sessions may exist in parallel.

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.

1.7 NOMENCLATURE

1.7.1 NORMATIVE TEXT

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.

1.7.2 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.

1.8 CONVENTIONS

1.8.1 FIGURES

Unified Modelling Language (UML) modelling diagrams are used in this document. Reference [2] provides further information regarding diagrams types and their meanings.

Each service that uses the MO Common Object Model provides a specification of the relationship between objects of the services. 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 shown in figure 1-1.

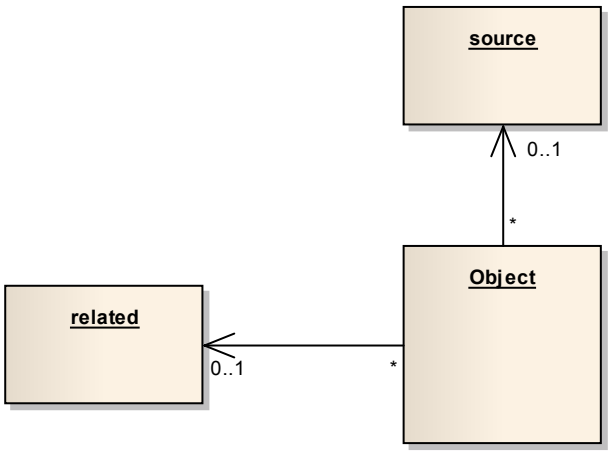

Figure 1-1: COM Structure

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 required) and therefore provides the ability to have 'n' levels of hierarchy.

1.8.2 TABLES

The table format information presented here is extracted from section 2 of reference [2]. It is repeated here to aid in understanding tables as they are presented in this document. A full description of the table formats presented in this document can be found in section 2 of reference [2] and in section 2 of reference [3].

Each interaction pattern definition contains a table that defines the template for operations that use that pattern.

Table 1-1: Example Operation Template

Operation Identifier	<<Operation name>>	
Interaction Pattern	<<Interaction pattern name>>	
Pattern Sequence	Message	Body Type
<<Message direction>>	<<Message name>>	<<Message type>>
…	…	…

The message direction denotes the direction of the message relative to the provider of the pattern and is either IN or OUT. All messages directed towards the provider are IN messages, and all messages directed away from the provider are OUT messages.

Blue cells (dark grey when printed on a monochrome printer) contain table headings, light grey cells contain fields that are fixed for a pattern, and white cells contain values that must be provided by the operation or structure.

Below each of these tables is a description of the operation messages and the fields in those messages. Requirements are provided to define the contents of those messages and fields and how they must be interpreted.

Certain fields may allow 'wildcard' values that match all possible values. For numeric fields, the wildcard value of zero ('0') is used, and for string fields, the string value of star ('*') is used.

1.8.3 ERRORS

Each operation may specify the errors it can return and may reference error codes from other specifications; for example:

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	Not Used

In the above example, the error code is defined in section 5 of reference .

1.9 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.

[3] *Mission Operations Common Object Model*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 521.1-B-1. Washington, D.C.: CCSDS, February 2014.

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

NOTE – Informative references are listed in annex E.

2 OVERVIEW

2.1 GENERAL

The Common Services provide infrastructure services to support the mission operations services. These Common services cover the following aspects of a system:

– Directory—Service publish and lookup;

– Login—Operator login;

– Configuration—Service configuration management.

The Common services provide functions that exist in the vast majority of systems but have previously been proprietary.

The following diagram presents the set of standards documentation in support of the Mission Operations Services Concept. The Common services belong to the Specifications documentation.

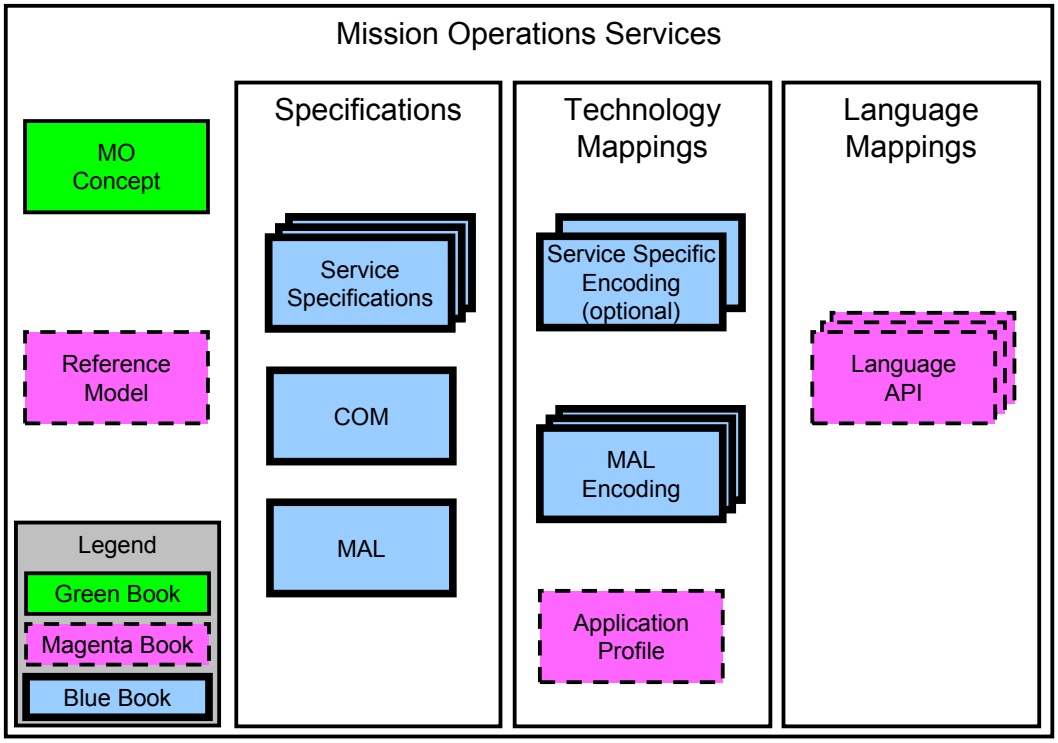

Figure 2-1: Mission Operations Services Concept Document Set

NOTE – References [E1] and [1] contain additional information about the MO Services Concept and the MO Reference Model, respectively.

The MO Common services build upon the layering concept outlined in reference [E1] and are defined in terms of the MO MAL outlined in reference [2], so it is possible to deploy

them over any supported protocol and message transport. The Common services are also defined in terms of the COM, as defined in reference [3], and build upon the COM event service and the COM archive.

NOTE – The services defined in this specification are not dependent on any of the other MO services, other than the MAL and the COM services, and require support from a compliant, deployed instance of the MO/MAL framework. An implementation is free to opt out of any service and also any capability set of these services; however, it may not make sense to support certain services without others.

2.2 DIRECTORY SERVICE

2.2.1 GENERAL

The directory service provides publish and lookup facilities to service providers and consumers. It allows providers to publish their location in the form of a URI (Universal Resource Indicator) so that consumers can locate it without having to know the location in advance. Strictly speaking, a directory is not required if a well-known service is to be used; however, in most circumstances, a directory provides required flexibility in the location of services.

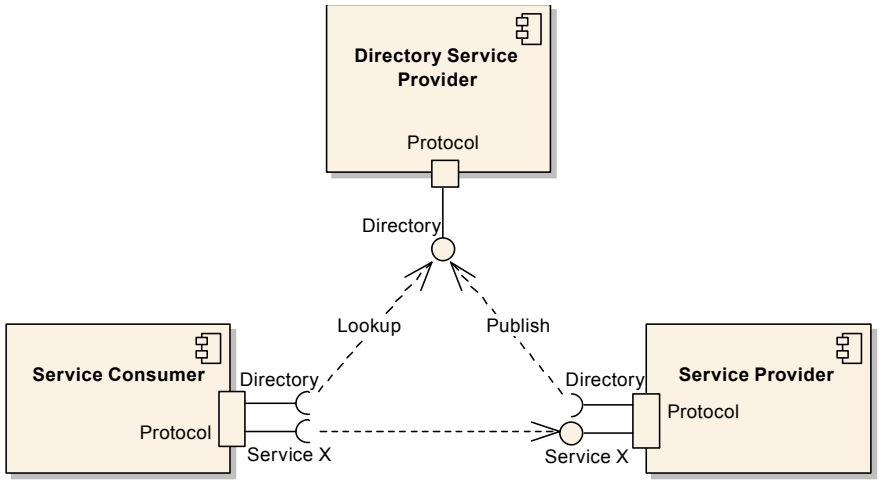

Figure 2-2: Directory Service Concept

A service provider publishes to the directory service the set of services, and the capabilities of those services, that it supports for a specific domain, network zone, and session.

Each entry for a service provider contains the associated service information, such as capability sets, the Quality of Service (QoS) levels that it can provide, and the URIs required to locate it. Capability sets provide a quick way for service providers to inform consumers of which optional capabilities of a service they support.

The addressing information associated with a service provider, the QoS and URI aspects, is held as a list, which allows a service provider to support more than one type of QoS and message transport technology in parallel.

The service directory does not contain any information about the relationships between services that are physically located in a single provider application, only that a provider supports several services. This is considered outside the scope of the Directory service.

The Directory service cannot be used by clients to locate the Directory service itself; therefore another means of determining its address is required, such as a well-known address or out-of-band agreement.

2.2.2 SERVICE PROVIDER PROPERTIES

Each entry in the Directory service for a particular service may also contain, over and above those listed in the previous section, a list of named properties that can be specified by the implementing service provider for passing relevant information to a service consumer. This list is open ended insomuch as any values can be added; however, table 2-1 lists the standard properties defined by the Directory service and its meanings. Other services may define other named properties required for that specific service.

Table 2-1: MO Service Provider Standard Properties

Property identifier	Permitted values	Default value	Description
TIMEREF	UTC TAI GPS	UTC	Defines the epoch of the time values obtained from this service provider. It is possible that an implementation-specific time reference is used; however, this is outside the scope of this specification.

2.3 LOGIN SERVICE

The Login service allows a service user to provide authentication information to the system. It takes the user's credentials and uses a deployment-specific mechanism to authenticate the user; the result of this is used by the MAL during access control.

The Login service and the access control provided by the MAL are fully dependent on a deployment-specific security architecture (for example, the authentication protocol Kerberos). Both layers (Common and MAL) provide access to, and use of, this security service; they do not implement it themselves. (See reference [1] for more information regarding access control.)

2.4 CONFIGURATION SERVICE

The Configuration service provides the ability to provide access to the configuration of a service provider, list available configurations for activation, and finally request activation of those configurations in the service providers.

A configuration has two parts, an identifier and a definition. The identifier of a configuration allows it to be referred to in operations, for example, whereas the configuration definition holds the actual details of the configuration. For example, a configuration of a Parameter service might have the identifier 'Temperatures' and the definition a set of temperature parameter definitions.

Implementations of the Configuration service may support bespoke configuration-upload mechanisms (such as file upload). This is supported by the configuration specification by which consumers would be notified of a new configuration; however, the details of these upload methods are outside of the scope of the Recommended Standard.

The specific configuration of a service is not covered in this Recommended Standard; however, it is fully expected that a service, a parameter service, for example, might specify the configuration that it requires and use the Configuration service to hold that configuration.

It should be noted that the Configuration service uses the COM Event service to activate and store configurations. Therefore the event system must be used for the Configuration service to work.

Example use cases for the Configuration service are explored in annex C.

2.5 TYPES OF CONFIGURATION

The Configuration service defines three supported types of Configuration:

- Hard-coded configurations:
  - Hard-coded configurations have identification, but the definition of the configuration cannot be modified; that is, their definition is represented by code rather than a configuration file.
  - An example is a camera preset mode.
- COM-compliant configurations:
  - COM-compliant configurations use the COM object model for representing configuration information.
  - An example is the set of available parameters in a M&C Parameter service provider.

- Non-COM-compliant configurations:
  - Non-COM-compliant configurations do not use the COM object model for holding the configuration information and are therefore service, deployment, and implementation specific.
  - An example is a non-MO application configuration file.

It also defines two levels of Configuration, as a Provider may support multiple different services:

- Service Configuration: the configuration of a single service in a provider, including all the attribute values of its definition objects.
- Provider Configuration: the set of 'Service Configurations' of a single provider, including zero to many Service Configurations.

Finally, it defines three supported ways of representing a Configuration:

- no representation (i.e., hard-coded configurations such as camera preset modes);
- COM-standardised representation (only for COM-compliant configuration types);
- implementation-specific representation.

For COM-compliant configurations, the Configuration service objects will only hold references to the actual service-specific COM objects. There will be no duplication. Therefore storing a new configuration (assuming that the service in question has created new objects) would only store links/references to those new objects in the COM archive.

It should be noted that there is no direct relationship between the contents of the Directory service and the Configuration service. For example, there can be a different set of providers in an instance of the Directory service from that managed by an instance of the Configuration service. Basically, the Directory service is used to hold the address and the supported operations of providers, whereas the Configuration service is used to hold the configuration of providers; however, there is no requirement for these two sets of providers to be the same (although that may make sense in some deployments).

The MO-supported approach is to use the Directory service to create and manage the set of providers, with the provider objects being shared between the Directory service and the Configuration service. When the set of providers used by the Configuration service is separate from those of a Directory service, the list of possible providers must be managed using implementation-specific mechanisms and is considered out of scope for this specification.

2.6 COM-STANDARDISED REPRESENTATION

The COM-standardised representation defines a simple mapping from the COM Objects used by the Configuration service to XML. It uses the standard MO XML mapping defined in reference [4].

The use of the standard representation is optional; however, defining and using the standard representation allows the movement and sharing of service configuration to be a simple matter of importing the standard XML.

The use of the COM-standardised representation is only supported by COM-based services.

3 SPECIFICATION: COMMON

3.1 OVERVIEW

This section details the Common Services; the structures used by the services and operations are detailed in section 4. The services and structures are defined in terms of the MO MAL defined in reference [2], so it is possible to deploy them over any supported protocol and message transport.

The services defined here are also specified in terms of the COM, which is defined in reference [3].

To aid comprehension, several tables are included for each service and operation definition. The table formats are the same as those used for the specification of the MO COM Services in reference [3]. The formats are fully described in reference [2] in section 2 and in reference [3] section 2, for COM usage.

All service specifications in this document are part of the Common Area.

3.2 SERVICE: DIRECTORY

3.2.1 OVERVIEW

The Directory service allows service providers to publish information about which services they provide and allows consumers to discover service provider address and capability information.

Service provider information is made available using the Directory service publishProvider operation and removed using the withdrawProvider operation.

The lookupProvider operation provides the ability for consumers to query the directory service based on a filter such as required service capability.

Finally, a provider can supply its set of service specification XML files when publishing its capabilities. This allows consumers that are able to process MO service XML files to obtain the files for further processing. It is expected that this would be used either to ensure that no modifications have been made by a provider to a standard service or, in the case of a consumer that is able to dynamically interact with new service specifications, to obtain the service specification to allow interaction with that new service.

Table 3-1: Directory Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
Common	Directory	3	1	1
Interaction Pattern	Operation Identifier	Operation Number	Support in Replay	Capability Set
REQUEST	lookupProvider	1	Yes	1
REQUEST	publishProvider	2	No	2
SUBMIT	withdrawProvider	3	No	
REQUEST	getServiceXML	4	No	3

3.2.2 HIGH-LEVEL REQUIREMENTS

The directory service shall provide the capability to

a) request service provider details based on a filter;

b) publish new service provider details;

c) withdraw service provider details; and

d) request any supplied service XML for a service provider.

3.2.3 FUNCTIONAL REQUIREMENTS

3.2.3.1 A service provider shall use the publishProvider operation to make available the set of services and the capabilities of those services that it offers.

3.2.3.2 The publish operation shall take a PublishDetails structure, which defines the capabilities and addressing information of that service provider.

3.2.3.3 It shall be possible for a provider to offer multiple service addresses and bindings to allow the provider to support multiple message transports and encodings.

3.2.3.4 Each service address entry shall include the service interface binding signature description, interface address, MAL protocol mapping, message encoding, the supported QoS levels, and associated QoS properties available for that connection.

3.2.3.5 A service consumer shall indicate its required QoS level and priority by using these values in the lookupProvider operation.

3.2.3.6 For services that have operations based on the publish and subscribe interaction pattern, a separate URI may be required for the broker component. To support this, the

directory service shall hold two URIs for each service, the URI of the primary interface and also a secondary one that, if present, is used for publish- and subscribe-based operations.

3.2.3.7 If a broker component is shared between more than one provider, known as a shared broker, the broker shall be published as a directory service entry without a primary interface URI.

3.2.3.8 Providers that wish to use the shared broker shall reference it by using its object instance identifier in the 'brokerProviderObjInstId' field in their directory service entry.

3.2.4 COM USAGE

3.2.4.1 A ServiceProvider object represents a service provider. The COM object body shall hold the service provider ID taken from the providerId field of the PublishDetails composite.

3.2.4.2 The ServiceProvider COM object related link shall not be used and therefore shall be set to NULL.

3.2.4.3 The ServiceProvider COM object source link shall not be used and therefore shall be set to NULL.

3.2.4.4 A ProviderCapabilities object represents the supported capabilities of a service provider. The COM object body shall hold the provider details of the service provider.

3.2.4.5 The ProviderCapabilities COM object related link shall indicate which ServiceProvider COM object it links to.

3.2.4.6 The ProviderCapabilities COM object source link shall not be used and therefore shall be set to NULL.

3.2.4.7 ServiceProvider and ProviderCapabilities objects shall be created by the publishService operation.

3.2.4.8 The object instance identifier for the ServiceProvider and ProviderCapabilities objects shall be populated by the provider of the directory service.

3.2.4.9 For the AddressDetails of a particular ProviderCapabilities object, if the brokerProviderObjInstId field contains a value and the brokerURI field is NULL, then the brokerURI value of the referenced provider shall be used.

Table 3-2: Directory Service Object Types

Object Name	Object Number	Object Body Type	Related points to	Source points to
ServiceProvider	1	MAL::Identifier	Set to NULL	Set to NULL
ProviderCapabilities	2	ProviderDetails	1	Set to NULL

3.2.5 COM OBJECT RELATIONSHIPS

Figure 3-1 shows the COM object relationships for this service:

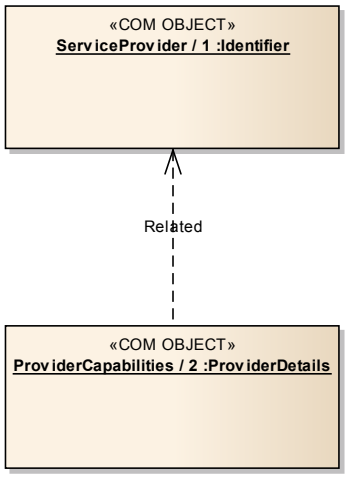

Figure 3-1: Directory Service COM Object Relationships

3.2.6 COM ARCHIVE SERVICE USAGE

All directory objects must be stored in the COM archive by the provider of the Directory service if the implementation of the Directory service uses a COM archive.

3.2.7 OPERATION: lookupProvider

3.2.7.1 Overview

The lookup operation allows a service consumer to query the directory service to return a list of service providers that match the requested criteria. If no match is found, then an empty list is returned.

NOTE – The various filters that may be specified as part of this operation are combined using AND logic.

Operation Identifier	lookupProvider	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	filter : (ServiceFilter)
OUT	RESPONSE	matchingProviders : (List<ProviderSummary>)

3.2.7.2 Structures

3.2.7.2.1 The filter field shall define the lookup query and be used to match details previously published using the publishProvider operation.

NOTE – The specifics of the ServiceFilter fields are defined in the following requirements.

3.2.7.2.2 If the serviceProviderId field is NULL, then all service provider identifiers shall be matched.

3.2.7.2.3 If the final identifier of the domain field of the filter is the wildcard '*', then all subdomains shall be searched for matches. (See reference [2], subsection 3.5.6.5g)).

3.2.7.2.4 If the wildcard is used in any other part of the domain other than the final one, then an INVALID error shall be returned.

3.2.7.2.5 If the domain field is NULL, then all domains shall be matched.

3.2.7.2.6 If the network field is NULL, then all networks shall be matched.

3.2.7.2.7 If the sessionType field is NULL, then all session types shall be matched.

3.2.7.2.8 If the sessionName field is NULL, then all session names shall be matched.

3.2.7.2.9 The serviceKey field shall be used to match against ServiceKey fields held in the PublishDetails used to publish a specific provider.

3.2.7.2.10 If the serviceKey field is NULL, then all areas, services, and versions shall be matched.

3.2.7.2.11 If the area field is the wildcard '0', then all areas names shall be matched.

3.2.7.2.12 If the service field is the wildcard '0', then all services shall be matched.

3.2.7.2.13 If the version field is the wildcard '0', then all area versions shall be matched.

3.2.7.2.14 If the requiredCapabilitySets field is NULL or an empty list, then all service capability sets shall be matched.

3.2.7.2.15 The operation shall return a list of service providers that match the filter.

3.2.7.2.16 If no service providers match the supplied filter, then an empty list shall be returned.

3.2.7.3 Errors

The operation may return the following error:

ERROR: INVALID

Invalid domain filter value.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

3.2.8 OPERATION: publishProvider

3.2.8.1 Overview

The publishProvider operation adds a new entry or updates an existing entry in the list of service providers held in the directory service.

Operation Identifier	publishProvider	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	newProviderDetails : (PublishDetails)
OUT	RESPONSE	providerObjId : (MAL::Long) capabilitiesObjId : (MAL::Long)

3.2.8.2 Structures

3.2.8.2.1 The newProviderDetails field shall hold the provider details of the service to be added or updated in the directory service.

3.2.8.2.2 If any of the fields of the newProviderDetails domain or network fields are either empty or contain the wildcard '*', an INVALID error shall be returned.

3.2.8.2.3 If the providerId field of the PublishDetails structure is empty or contains the wildcard '*', an INVALID error shall be returned.

3.2.8.2.4 For each contained ServiceKey structure, if the area/service/version fields contain '0', then an INVALID error shall be returned.

3.2.8.2.5 For each contained supportedCapabilitySets list, if the list is empty or contains '0', then an INVALID error shall be returned.

3.2.8.2.6 If the supportedLevels list is empty or the priorityLevels field is '0' for each contained AddressDetails structure found either within the ProviderDetails or the inner ServiceCapability structures, then an INVALID error shall be returned.

3.2.8.2.7 If an error is being returned, then no changes shall be made.

3.2.8.2.8 If the providerId field of the PublishDetails structure matches an existing ServiceProvider COM object, the operation shall update the existing details of that provider.

3.2.8.2.9 If the providerId field of the PublishDetails structure does not match an existing ServiceProvider COM object, then the operation shall create a new ServiceProvider COM object to represent the new service provider.

3.2.8.2.10 A new ProviderCapabilities COM object shall be created to store the capabilities of the provider.

3.2.8.2.11 The created objects should be stored in the COM archive by the directory service provider.

3.2.8.2.12 The operation shall return the COM object instance identifiers of the ServiceProvider and ProviderCapabilities COM objects representing the provider.

3.2.8.3 Errors

The operation may return the following error:

ERROR: INVALID

Submitted values are invalid.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

3.2.9 OPERATION: withdrawProvider

3.2.9.1 Overview

The withdrawProvider operation removes an existing entry from the list of service providers held in the directory service. If no match is found for the withdraw request, then nothing is changed.

Operation Identifier	withdrawProvider	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Signature
IN	SUBMIT	providerObjId : (MAL::Long)

3.2.9.2 Structures

3.2.9.2.1 The providerObjId field shall hold the object instance identifier for the ServiceProvider COM object to remove from the directory service.

3.2.9.2.2 If the supplied identifier is ‘0’, an INVALID error shall be returned.

3.2.9.2.3 If the supplied identifier does not match an existing ServiceProvider COM object then an UNKNOWN error shall be returned.

3.2.9.2.4 If an error is being returned, then no changes shall be made.

3.2.9.2.5 The matched provider shall be removed from the directory service.

3.2.9.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN

Provider to withdraw was not found.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID

Submitted values are invalid.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

3.2.10 OPERATION: getServiceXML

3.2.10.1 Overview

The getServiceXML operation returns the list of XML files that were submitted by the service provider via the publishProvider operation.

If no files were supplied, then this operation returns an empty list.

Operation Identifier	getServiceXML	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	providerObjId : (MAL::Long)
OUT	RESPONSE	xmlFiles : (List<MAL::File>)

3.2.10.2 Structures

3.2.10.2.1 The providerObjId field shall hold the COM object instance identifier for the ServiceProvider for which to obtain the service XML.

3.2.10.2.2 If the supplied instance identifier is ‘0’, an INVALID error shall be returned.

3.2.10.2.3 If the supplied identifier does not match an existing ServiceProvider COM object, then an UNKNOWN error shall be returned.

3.2.10.2.4 The list of XML files supplied during the publishProvider operation for the matched provider shall be returned.

3.2.10.2.5 If no XML files were supplied by the provider, then an empty list shall be returned.

3.2.10.3 Errors

The operation may return one of the following errors:

a) ERROR: INVALID

Submitted values are invalid.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

b) ERROR: UNKNOWN

Provider was not found.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

3.3 SERVICE: LOGIN

3.3.1 OVERVIEW

The Login service defines the primary mechanism for the submission of authentication credentials to a deployment specific security system. It supports operations to allow a user to login, logout, report available roles, and also hand over the login to another user.

The service is closely tied to the Access Control aspect of the MAL where the returned authentication identifiers are used in the MAL message header to authenticate and authorise messages via Access Control.

The login service supports the concept of roles, through which users may log in with a specific role; the meaning of each role is mission-specific. However, it is expected that a specific role allocates the user privileges to invoke operations on mission operation services.

The use of login roles is optional, but if they are used, then the role details are held in the COM archive, and the COM archive operations are used to manage the role definitions. The responsibility for maintenance of the login roles is outside the scope of this specification as it is a deployment issue to define the possible roles and associate users to those roles.

The form in which the password is sent to the Login service provider must be agreed upon beforehand and is dependent on the security system deployed. For example, most security implementations do not recommend the use of plain text passwords but rather some encrypted version of the password. For this reason, the contents of the messages sent between the login service consumer and provider during authentication handshaking are implementation-specific.

The authentication and authorisation concept of the MO services is covered in subsection 3.6 of the Reference Model (reference [2]).

The Reference Model (reference [2]) also provides sequence diagrams for security and login in subsections 5.2 and 5.3.

Table 3-3: Login Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
Common	Login	3	2	1
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in Replay**	**Capability Set**
REQUEST	login	1	No	1
SUBMIT	logout	2	No	
REQUEST	listRoles	3	No	2
REQUEST	handover	4	No	3

3.3.2 HIGH LEVEL REQUIREMENTS

The login service shall provide the capability to

a) log in;

b) log out;

c) report the list of possible roles for a login profile; and

d) hand over an existing login from one user to another.

3.3.3 COM USAGE

3.3.3.1 Instances of a user login shall be represented as LoginInstance COM objects.

3.3.3.2 Instances of a user role shall be represented as LoginRole COM objects.

3.3.3.3 LoginInstance objects shall be created by the login and handover operations.

3.3.3.4 The object instance identifier for a LoginInstance object shall be populated by the provider of the Login service.

3.3.3.5 The LoginInstance object shall use the related link to indicate which LoginRole (if any) the login uses.

3.3.3.6 If a LoginInstance object was created during a login operation, then the source link shall be set to NULL.

3.3.3.7 If a LoginInstance object was created during a handover operation it shall use the source link to indicate which LoginInstance the login was handed over from.

Table 3-4: Login Service Object Types

Object Name	Object Number	Object Body Type	Related points to	Source points to
LoginRole	1	MAL::Identifier	Set to NULL	Set to NULL
LoginInstance	2	Profile	1	2

3.3.4 COM EVENT SERVICE USAGE

3.3.4.1.1 When a user logs in, a LoginEvent COM event shall be generated.

3.3.4.1.2 When a user logs out, a LogoutEvent COM event shall be generated.

3.3.4.1.3 When the handover operation is successful, a LogoutEvent COM event shall be generated for the previous login, and a LoginEvent COM event shall be generated for the new login.

3.3.4.1.4 The LoginEvent event shall use the related link to indicate which LoginInstance object is being logged in.

3.3.4.1.5 The LogoutEvent event shall use the related link to indicate which LoginInstance object is being logged out.

3.3.4.1.6 The LoginEvent event shall not use the COM object source link and shall set it to NULL.

3.3.4.1.7 The LogoutEvent event shall use the source link to indicate which LoginEvent event it is logging out.

3.3.4.1.8 The events shall be published using the COM event service.

Table 3-5: Login Service Events

Event Name	Object Number	Object Body Type	Related points to	Source points to
LoginEvent	3	No body	2	Set to NULL
LogoutEvent	4	No body	2	3

3.3.5 COM OBJECT RELATIONSHIPS

Figure 3-2 shows the COM object and event relationships for the Login service.

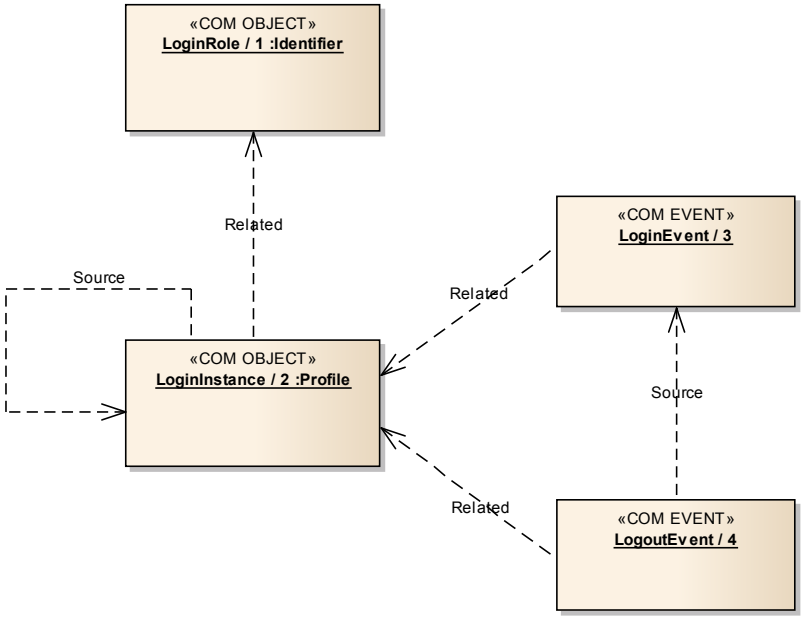

Figure 3-2: Login Service COM Object and Event Relationships

3.3.6 COM ARCHIVE SERVICE USAGE

3.3.6.1 LoginInstance objects should be stored in the COM archive.

3.3.6.2 When a login service event is published, the event object should be stored in the COM archive by the Login service provider.

3.3.6.3 LoginRole objects should be stored in the COM archive.

3.3.6.4 The COM archive service should be used for maintaining the LoginRole objects.

3.3.7 OPERATION: login

3.3.7.1 Overview

The login operation allows a user to log in to the system. A user can log in more than once by using a different role; however, a specific deployment may place limits on the number of users that may use a specific role, and in that case, it will fail the login operation with the TOO_MANY error.

Operation Identifier	login	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	userDetails : (Profile) password : (MAL::String)
OUT	RESPONSE	authId : (MAL::Blob) objInstId : (MAL::Long)

3.3.7.2 Structures

3.3.7.2.1 The authenticationId field of the REQUEST message must be NULL otherwise an INVALID error shall be returned.

3.3.7.2.2 The authenticationId field shall be checked before applying all other tests here.

3.3.7.2.3 The userDetails field shall contain the details of the new user and role combination.

3.3.7.2.4 If the username field of the supplied Profile structure is either the wildcard '*' or empty, an INVALID error shall be returned.

3.3.7.2.5 If roles are required by the system and the role field of the supplied Profile structure is NULL, then an INVALID error shall be returned.

3.3.7.2.6 If roles are not used by the system, the role field of the supplied Profile structure shall be ignored and may be set to NULL.

3.3.7.2.7 An UNKNOWN error shall be returned if the username, password, and role combination are not correct for the system, that is, unknown user/role or incorrect password.

3.3.7.2.8 A DUPLICATE error shall be returned if the username and role combination is currently in use.

3.3.7.2.9 A TOO_MANY error shall be returned if the username or role is already used and exceeds (deployment dependent) the maximum number of concurrent logins/roles.

3.3.7.2.10 If the login is successful, the provider shall create a new LoginInstance COM object and store it in the COM archive if the implementation of the Login service uses a COM archive.

3.3.7.2.11 The related link of the new LoginInstance COM object shall be set to the requested LoginRole COM object.

3.3.7.2.12 A LoginEvent COM event shall be generated at this point.

3.3.7.2.13 The returned authId field shall be used as the authenticationId field in future MAL messages by the consumer MAL for authentication. The token is specific to the user and role in use.

3.3.7.2.14 The returned objInstId field shall contain the LoginInstance COM object instance identifier that was created by the login operation.

3.3.7.3 Errors

The operation may return one of the following errors:

a) ERROR: DUPLICATE

Username/role combination currently in use.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	Not Used

b) ERROR: INVALID

Submitted profile contains invalid values. No further information is provided as it may compromise security.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

c) ERROR: TOO_MANY

Role concurrent session limit count exceeded.

Error	Error #	ExtraInfo Type
TOO_MANY	Defined in MAL	Not Used

d) ERROR: UNKNOWN

Unknown username/role/password combination.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

3.3.8 OPERATION: logout

3.3.8.1 Overview

The logout operation allows a user to log out from the system. No information is passed in the message as the MAL authentication Id is enough to identify the login.

Operation Identifier	logout	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Signature
IN	SUBMIT	Empty

3.3.8.2 Structures

3.3.8.2.1 Upon reception of the message, the operation shall remove the matched user from the set of logged-in users in the login service provider.

3.3.8.2.2 A logged-in user shall be matched using the supplied authenticationId.

3.3.8.2.3 A LogoutEvent COM event shall be generated at this point.

3.3.8.2.4 No errors shall be returned by this operation.

3.3.8.3 Errors

The operation does not return any errors.

3.3.9 OPERATION: listRoles

3.3.9.1 Overview

The listRoles operation returns the list of available roles for a specific user. This operation is expected to be called before a user logs in so that the software can provide a list of possible roles.

It should be noted that this operation requires both a username and password field before returning any information. This is to ensure that it does not provide a security attack vector by allowing the discovery of valid usernames without first knowing the correct password.

Operation Identifier	listRoles	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	username : (MAL::Identifier) password : (MAL::String)
OUT	RESPONSE	permittedRoles : (List<MAL::Long>)

3.3.9.2 Structures

3.3.9.2.1 The username field shall hold the details of the user.

3.3.9.2.2 If the username field is either the wildcard '*', NULL, or empty, an INVALID error shall be returned.

3.3.9.2.3 An UNKNOWN error shall be returned if the username and password combination are not correct for the system, that is, unknown user or incorrect password.

3.3.9.2.4 The operation shall return a list of LoginRole object instance identifiers that are permitted for the user or NULL if roles are not used by the system.

3.3.9.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN

Unknown username/password combination.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID

Submitted profile contains invalid values.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

3.3.10 OPERATION: handover

3.3.10.1 Overview

The handover operation allows an existing login to be transferred to a new user. Two cases are expected here: the first is when the operation is used to change the user's current role, and the second is when an operation's context is handed over to another user.

Operation Identifier	handover	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	newUserDetails : (Profile) newUserPassword : (MAL::String)
OUT	RESPONSE	newAuthId : (MAL::Blob) newLoginInstId : (MAL::Long)

3.3.10.2 Structures

3.3.10.2.1 The newUserDetails field shall contain the details of the new user and role combination.

3.3.10.2.2 If the username field of the supplied Profile structure is either NULL, the wildcard '*', or empty, an INVALID error shall be returned.

3.3.10.2.3 If roles are required by the system and the role field of the supplied Profile structure is NULL, then an INVALID error shall be returned.

3.3.10.2.4 The role field of the supplied Profile structure may be NULL if roles are not used by the system.

3.3.10.2.5 An UNKNOWN error shall be returned if the username, password, and role combination are not correct for the system, that is, unknown user/role or incorrect password.

3.3.10.2.6 A DUPLICATE error shall be returned if the username and role combination is currently in use.

3.3.10.2.7 A TOO_MANY error shall be returned if the username or role are already used and exceed the permitted maximum usage value (deployment dependent).

3.3.10.2.8 The DUPLICATE and TOO_MANY checks shall take into account the fact that the current operator/role combination will be logged out after the handover operation completes.

3.3.10.2.9 If the handover is successful, the provider shall create a new LoginInstance COM object and store it in the COM archive.

3.3.10.2.10 The related link of the new LoginInstance COM object shall be set to the requested LoginRole COM object.

3.3.10.2.11 If an error is raised, then the handover operation shall fail and the original login will remain active.

3.3.10.2.12 The source link of the new LoginInstance COM object shall be set to the LoginInstance COM object that represents the previous login.

3.3.10.2.13 If the handover operation is successful, a LogoutEvent COM event shall be generated for the previous login and a LoginEvent COM event shall be generated for the new login.

3.3.10.2.14 The returned newAuthId field shall be used as the authenticationId field in future MAL messages by the consumer MAL for authentication. The token is specific to the new user and role in use.

3.3.10.2.15 The returned newLoginInstId field shall contain the new LoginInstance COM object instance identifier that was created by the operation.

3.3.10.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN

Unknown username/role/password combination.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID

Submitted profile contains invalid values.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

c) ERROR: TOO_MANY

Role concurrent session limit count exceeded.

Error	Error #	ExtraInfo Type
TOO_MANY	Defined in MAL	Not Used

d) ERROR: DUPLICATE

Username/role combination currently in use.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	Not Used

3.4 SERVICE: CONFIGURATION

3.4.1 OVERVIEW

The Configuration service allows a service consumer to activate predefined configurations of a service provider.

The service uses a COM object to represent the different configurations allowed by a service provider; a service consumer selects one configuration to activate using the Configuration service activate operation.

The contents of a configuration for a service provider is deployment specific; however, the management of these configurations, and the selection of a configuration for current use, is the purpose of this service.

Implementations of this service may also use bespoke methods for configuration representation (such as hard-coded configuration), which is outside the scope of this specification; however, the status and management of these configurations can still be managed with this service.

NOTE – This service uses the COM Event service to distribute changes in state. Therefore participating service providers should ensure they are all using the same shared COM Event service if they wish to receive Configuration service events.

Table 3-6: Configuration Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
Common	Configuration	3	5	1
Interaction Pattern	Operation Identifier	Operation Number	Support in Replay	Capability Set
INVOKE	activate	1	No	1
REQUEST	list	2	No	2
REQUEST	getCurrent	3	No	
REQUEST	exportXML	4	No	3
SUBMIT	add	5	No	4
SUBMIT	remove	6	No	
INVOKE	storeCurrent	7	No	5
REQUEST	importXML	8	No	6

3.4.2 HIGH LEVEL REQUIREMENTS

The Configuration service shall provide the capability to

a) activate a configuration;

b) list available configurations;

c) manage the available configurations;

d) get and store the current configuration; and

e) import/export configurations from/to XML.

3.4.3 COM USAGE

3.4.3.1 A ProviderConfiguration COM object represents a single set of configuration details that may be used by multiple service providers. The object body shall hold an identifier to identify the provider configuration.

3.4.3.2 The ProviderConfiguration COM object related link shall link to the ConfigurationObjects COM object that contains the objects that form the provider configuration.

3.4.3.3 If the ProviderConfiguration represents a hard-coded configuration for a provider, then the related link shall be set to NULL.

3.4.3.4 The ProviderConfiguration COM object source link should be the object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.

3.4.3.5 A ServiceConfiguration COM object represents a single configuration of a service in a provider; multiple ProviderConfiguration COM objects may reference the same service configuration via its linked ConfigurationObjects COM object. The object body shall hold the configuration name and the service key that identifies the service it is a configuration of.

3.4.3.6 The ServiceConfiguration COM object related link shall link to the ConfigurationObjects COM object that contains the objects that form the service configuration.

3.4.3.7 If the ServiceConfiguration represents a hard-coded configuration for a single service, then the related link shall be set to NULL.

3.4.3.8 The ServiceConfiguration COM object source link should be the object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.

3.4.3.9 A ConfigurationObjects COM object represents a set of links to a set of COM objects that form an actual configuration. The object body shall hold a configuration object details structure that links to the objects that form the configuration.

3.4.3.10 The ConfigurationObjects COM object related link shall be set to NULL.

3.4.3.11 The ConfigurationObjects COM object source link should be the object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.

3.4.3.12 A ConfigurationFile COM object provides a mechanism for holding files that contain non-COM configuration information so that it can still be managed and referenced by the Configuration service. The object body shall hold the configuration file.

3.4.3.13 The ConfigurationFile COM object related link shall be set to NULL.

3.4.3.14 The ConfigurationFile COM object source link should be the object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.

3.4.3.15 A non-COM configuration shall be represented by linking from a ConfigurationObjects COM object to one or more ConfigurationFile COM objects. The ConfigurationFile COM objects shall be referenced by a ConfigurationObjectSet in the ConfigurationObjects body.

3.4.3.16 A ProviderConfigurationLink COM object represents the use of a provider configuration by a specific service provider. The object body is not used and shall be NULL.

3.4.3.17 The ProviderConfigurationLink COM object related link shall link to the ServiceProvider COM object that represents the service provider in the Directory Service.

3.4.3.18 The ProviderConfigurationLink COM object source link shall link to the ProviderConfiguration COM object in use by the provider.

Table 3-7: Configuration Service Object Types

Object Name	Object Number	Object Body Type	Related points to	Source points to
ServiceConfiguration	1	ServiceConfigurationIdentifier	3	The object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.
ProviderConfiguration	2	MAL::Identifier	3	The object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.
ConfigurationObjects	3	ConfigurationObjectDetails	Set to NULL	The object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.
ConfigurationFile	4	MAL::File	Set to NULL	The object that caused it to be created, most likely a COM OperationActivity object or an operator login in the case of off-line editors being used.
ProviderConfigurationLink	5	No body	Directory::1	2

3.4.4 COM EVENT SERVICE USAGE

3.4.4.1 A ConfigurationSwitch COM event shall represent a request for change in active configuration.

3.4.4.2 The ConfigurationSwitch COM event body shall hold the configuration to be activated.

3.4.4.3 The ConfigurationSwitch COM event related link shall point to the COM ServiceProvider object that is required to attempt the configuration switch.

3.4.4.4 The ConfigurationSwitch COM event source link shall indicate the activate COM OperationActivity object that caused it to be created.

3.4.4.5 A ConfigurationSwitched COM event shall be raised when a configuration is made active.

3.4.4.6 The ConfigurationSwitched COM event body shall hold the result of the activation attempt.

3.4.4.7 If the activation was successful, then the body of the ConfigurationSwitched event shall be set to TRUE, otherwise FALSE for failure.

3.4.4.8 The ConfigurationSwitched COM event related link shall point to the ConfigurationSwitch event that triggered the switch attempt.

3.4.4.9 The ConfigurationSwitched COM event source link shall point to the previous active configuration or be NULL if no configuration was previously active.

3.4.4.10 A ConfigurationStore COM event shall represent a request to store the current configuration.

3.4.4.11 The ConfigurationStore COM event body may contain the configuration name and ServiceKey of the service configuration to store in a ServiceConfigurationIdentifier composite; it shall be NULL if the provider configuration is to be stored.

3.4.4.12 The ConfigurationStore COM event related link shall link to the ServiceProvider object that represents the provider that must store its configuration.

3.4.4.13 The ConfigurationStore COM event source link shall indicate the storeCurrent COM OperationActivity object that caused it to be created.

3.4.4.14 A ConfigurationStored COM event shall be raised when a configuration has been stored.

3.4.4.15 The ConfigurationStored COM event body shall hold the ObjectId of the new Configuration object if the store was successful, otherwise NULL for failure.

3.4.4.16 The ConfigurationStored COM event related link shall point to the ConfigurationStore event that triggered the store attempt.

3.4.4.17 The ConfigurationStored COM event source link shall be NULL.

Table 3-8: Configuration Service Events

Event Name	Object Number	Object Body Type	Related points to	Source points to
ConfigurationSwitch	6	COM::ObjectId	Directory::1	The source link of the event is the object that triggered the activation, most likely the activate operation.
ConfigurationSwitched	7	MAL::Boolean	6	The previous active configuration or NULL if no configuration was previously active.
ConfigurationStore	8	ServiceConfigurationIdentifier	Directory::1	The source link of the event is the object that triggered the activation, most likely the storeCurrent operation.
ConfigurationStored	9	COM::ObjectId	8	Set to NULL.

3.4.5 COM OBJECT RELATIONSHIPS

Figure 3-3 shows the COM object and event relationships for this service:

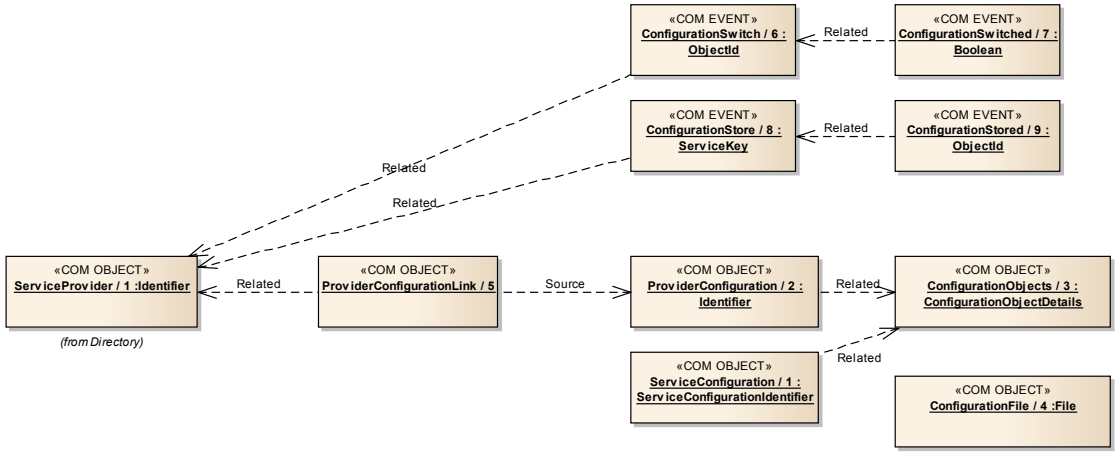

Figure 3-3: Configuration Service COM Object and Event Relationships

3.4.6 COM ARCHIVE SERVICE USAGE

3.4.6.1 Configuration definition objects (ServiceConfiguration, ProviderConfiguration, ProviderConfigurationLink, ConfigurationObjects, ConfigurationFile) must be stored in the COM archive by the provider of the Configuration service if the implementation of the Configuration service uses a COM archive.

3.4.6.2 The COM objects that hold the actual configuration definition details must be stored in the COM archive by the provider of the relevant service if the implementation of the Configuration service uses a COM archive.

3.4.6.3 When a configuration event (ConfigurationSwitch, ConfigurationSwitched, ConfigurationStore, ConfigurationStored) is published, the event objects should be stored in the COM archive.

3.4.7 OPERATION: activate

3.4.7.1 Overview

The activate operation instructs a service provider to make a specific configuration active. The operation returns once the new configuration is active or when the activation attempt has failed for some reason.

The requested configuration must either have already been added to the Configuration service provider using the add operation or alternatively be known to the Configuration service by another implementation-specific mechanism.

NOTE – The service is reconfigured for all service consumers, not just the calling consumer.

Operation Identifier	activate	
Interaction Pattern	INVOKE	
Pattern Sequence	Message	Body Signature
IN	INVOKE	serviceProvider : (COM::ObjectKey) configObjId : (COM::ObjectId)
OUT	ACK	Empty
OUT	RESPONSE	activationResult : (MAL::Boolean) previousConfig : (List<COM::ObjectId>)

3.4.7.2 Structures

3.4.7.2.1 The serviceProvider field shall contain the domain and object instance identifier of the ServiceProvider COM object being (re)configured.

3.4.7.2.2 If the service provider referenced by the serviceProvider field is not known, an UNKNOWN error shall be returned.

3.4.7.2.3 The configObjId field shall hold the COM object identifier that identifies the configuration to activate.

3.4.7.2.4 The configObjId field shall reference either a ProviderConfiguration or ServiceConfiguration object.

3.4.7.2.5 An UNKNOWN error shall be returned if the object instance identifier held in the configObjId field does not match an existing configuration.

3.4.7.2.6 An INVALID error shall be returned if the object instance identifier held in the configObjId field does not reference either a ProviderConfiguration or ServiceConfiguration object.

3.4.7.2.7 If the object instance identifier held in the configObjId field does not reference a valid configuration for the service provider, an INVALID error shall be returned. A valid configuration is one that is returned from the list operation for the matched service provider.

3.4.7.2.8 A ConfigurationSwitch COM Event shall be published if no errors are being returned containing the ObjectId of the service or provider configuration to use, that is, the contents of the configObjId field of this operation.

3.4.7.2.9 The acknowledgement message shall be sent at this point.

3.4.7.2.10 The service provider that implements the selected service shall, after the reception of the ConfigurationSwitch event, reconfigure itself and publish a ConfigurationSwitched COM event.

3.4.7.2.11 If the operation fails, the previous configuration shall remain active.

3.4.7.2.12 In the case of a provider configuration, in which multiple service configurations are being switched, the provider must switch all configurations successfully or roll back to the previous configuration. No partial reconfiguration is supported.

3.4.7.2.13 If a provider level configuration is successful, and a COM archive is being used, then the service provider that implements the selected service shall store in the COM archive a new ProviderConfigurationLink COM object that links its ServiceProvider object to the new activated ProviderConfiguration COM object.

3.4.7.2.14 The response message shall be sent when the configuration is either made active or fails.

3.4.7.2.15 If the activation was successful, then the activationResult field shall be set to TRUE; otherwise it shall be set to FALSE for failure.

3.4.7.2.16 The previousConfig field shall point to the previously active configuration or NULL if no configuration was previously active or the activationResult was FALSE for failure.

3.4.7.2.17 If a service configuration was requested, in which the configObjId field referenced a ServiceConfiguration, the response shall contain a list of a single item of the previous ServiceConfiguration COM object identifier.

3.4.7.2.18 If a provider configuration was requested, in which the configObjId field referenced a ProviderConfiguration, the response shall contain a list of object identifiers composed of the previous ProviderConfiguration COM object identifier followed by the list of ServiceConfiguration object identifiers that were active for that provider.

3.4.7.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN

Requested configuration or service provider is unknown.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID

Requested configuration is invalid.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

3.4.8 OPERATION: list

3.4.8.1 Overview

The list operation returns a list of configurations known to the Configuration service provider for a certain configuration type in a specific domain.

To appear in the response from the list operation, a configuration must either have been added to the Configuration service provider using the add operation or alternatively be known to the Configuration service by another implementation specific mechanism.

Operation Identifier	list	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	configurationType : (ConfigurationType) domain : (List<MAL::Identifier>) serviceKey : (ServiceKey)
OUT	RESPONSE	objInstIds : (List<COM::ObjectId>)

3.4.8.2 Structures

3.4.8.2.1 The configurationType argument shall hold the type of configuration to be listed.

3.4.8.2.2 The domain request argument shall contain the domain of the configuration objects to return.

3.4.8.2.3 The domain field supports the wildcard value of '*' only in the last part of the domain; for other wildcard placement, an INVALID error shall be returned. (See subsection 3.5.6.5g) in reference [2].)

3.4.8.2.4 If the requested configuration type is SERVICE, then an optional filter may be supplied in the serviceKey field where the ServiceKey composite holds the service area, service, and version values to match on.

3.4.8.2.5 The filter shall be applied with a logical AND for each field of the service key.

3.4.8.2.6 Wildcard values of '0' are not accepted in the serviceKey fields; an INVALID error shall be returned if the wildcard value of '0' appears in the serviceKey field.

3.4.8.2.7 For other types of configuration, the serviceKey field shall be ignored and may be set to NULL in the request.

3.4.8.2.8 The operation shall return the list of matched configuration object identifiers known to the Configuration service provider.

3.4.8.2.9 If no configurations matched, then an empty list shall be returned.

3.4.8.3 Errors

The operation may return the following error:

ERROR: INVALID

Wildcard values were specified in the service key filter or in the domain field (except in the last part of the domain).

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

3.4.9 OPERATION: getCurrent

3.4.9.1 Overview

The getCurrent operation returns the currently selected configuration of a service provider, either of the complete provider (ProviderConfiguration) or a specific service (ServiceConfiguration) of that provider, as far as the Configuration service provider is concerned.

This means that if the provider of a specific service has modified its configuration by some other means, and the configuration has not been stored using the storeCurrent operation, then this operation will return the unmodified configuration.

If a provider-level configuration is required by this operation, and service configurations different from the ones in the provider configuration have been activated by the activate operation, then this operation will return the ProviderConfiguration followed by the list of modified ServiceConfiguration objects.

The current configuration of a service provider is most likely the configuration that was last activated using the activate operation; however, implementations of the Configuration service may have other means (outside the scope of this service) of selecting the current configuration of a service provider.

The operation can also be used to determine the initial configuration of a service provider before it has been activated. It is therefore a useful operation to call by a service provider during start-up to determine its initial configuration. If this is the case, a ConfigurationSwitched event is recommended to be published.

Operation Identifier	getCurrent	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	serviceProvider : (COM::ObjectKey) serviceKey : (ServiceKey)
OUT	RESPONSE	objInstId : (List<COM::ObjectId>)

3.4.9.2 Structures

3.4.9.2.1 The serviceProvider field shall contain the domain and object instance identifier of the ServiceProvider COM object being queried.

3.4.9.2.2 If the serviceKey field is not NULL, then the operation shall return the configuration of the selected service, as specified in the field.

3.4.9.2.3 For retrieval of the provider level configuration the serviceKey field shall be set to NULL in the request.

3.4.9.2.4 An UNKNOWN error shall be returned if the combination of service provider and service filter fields do not match an existing service provider, service key, or configuration.

3.4.9.2.5 No wildcards are supported; an INVALID error shall be returned if wildcards are present.

3.4.9.2.6 If a service configuration was requested, in which the serviceKey was not NULL, the response shall contain a list of a single item of the matched ServiceConfiguration COM object identifier.

3.4.9.2.7 If a provider configuration was requested, in which the serviceKey was NULL, the response shall contain a list of the matched ProviderConfiguration COM object identifiers followed by the list of ServiceConfiguration object identifiers active for that provider.

3.4.9.3 Errors

The operation may return one of the following errors:

a) ERROR: INVALID

The request contained one or more wildcards.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

b) ERROR: UNKNOWN

Requested service provider and service key combination is unknown.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

3.4.10 OPERATION: exportXML

3.4.10.1 Overview

The exportXML operation returns the actual Configuration information in the XML format from the configuration object stored in the Archive.

The returned XML is in the standardised format, and one of two levels of detail, compact or complete, can be selected. Compact contains just the COM object instance identifiers with the respective domains and object types; complete augments the compact with the additional set of values inside the respective service objects.

The XML standardised format is the XML representation of the referenced COM objects as defined by the XML encoding given in reference [4]. Example XML documents can be found in annex F.

If the implementation of the Configuration service is not using a COM archive, then an error is returned.

It should be noted that this operation supports only COM configurations.

Operation Identifier	exportXML	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	confObjId : (COM::ObjectId) returnComplete : (MAL::Boolean)
OUT	RESPONSE	xmlConfiguration : (MAL::File)

3.4.10.2 Structures

3.4.10.2.1 If the implementation of the Configuration service is not using a COM archive, then an UNSUPPORTED_OPERATION error shall be returned.

3.4.10.2.2 The confObjId argument shall contain the type, domain, and COM object instance identifier of the configuration object to return the XML representation of.

3.4.10.2.3 An UNKNOWN error shall be returned if the confObjId field does not match an existing COM object.

3.4.10.2.4 An INVALID error shall be returned if the confObjId does not refer to either a ProviderConfiguration or a ServiceConfiguration object.

3.4.10.2.5 An INVALID error shall be returned if the confObjId refers to either a hard-coded or a non-COM configuration.

3.4.10.2.6 The returnComplete Boolean shall be set to True if the returned XML is to be in the complete standardised format; otherwise it will be in the compact standardised format.

3.4.10.2.7 The returned File object shall contain the configuration XML.

3.4.10.2.8 The Configuration object shall not be deleted from the COM Archive.

3.4.10.3 Errors

The operation may return one of the following errors:

a) ERROR: INVALID

Requested configuration is not a valid configuration object type.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

b) ERROR: UNKNOWN

Requested object is unknown.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

c) ERROR: UNSUPPORTED_OPERATION

The operation requires the use of a COM archive.

Error	Error #	ExtraInfo Type
UNSUPPORTED_OPERATION	Defined in MAL	Not Used

3.4.11 OPERATION: add

3.4.11.1 Overview

The add operation makes a new Configuration available on the Configuration service. The Configuration must already exist in the COM archive to be added to the Configuration service.

If the implementation of the Configuration service is not using a COM archive then an error is returned.

This operation can be used to add COM, non-COM, and hard-coded configurations (the mechanism to store the hard-coded configuration in the COM archive is implementation specific).

Operation Identifier	add	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Signature
IN	SUBMIT	serviceProvider : (COM::ObjectKey) configObjIds : (List<COM::ObjectId>)

3.4.11.2 Structures

3.4.11.2.1 If the implementation of the Configuration service is not using a COM archive, then an UNSUPPORTED_OPERATION error shall be returned.

3.4.11.2.2 The first argument shall contain the domain and object identifier of the ServiceProvider COM object to which the configurations are being added.

3.4.11.2.3 The second argument shall contain a list of service and/or provider configurations to add to the list of configurations available for the specific service provider.

3.4.11.2.4 If either the service provider or the configuration objects are unknown, then an UNKNOWN error shall be returned.

3.4.11.2.5 If any of the supplied configuration objects are not provider or service configuration objects, then an INVALID error shall be returned.

3.4.11.2.6 If an error is raised, then no new configurations shall be added as a result of this operation call.

3.4.11.3 Errors

The operation may return one of the following errors:

a) ERROR: INVALID

1) One of the supplied configuration object instance identifiers is either not a Service or a Provider configuration object.

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

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

b) ERROR: UNSUPPORTED_OPERATION

The operation requires the use of a COM archive.

Error	Error #	ExtraInfo Type
UNSUPPORTED_OPERATION	Defined in MAL	Not Used

c) ERROR: UNKNOWN

1) One of the supplied service or configuration object instance identifiers is unknown.

2) A list of 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>

3.4.12 OPERATION: remove

3.4.12.1 Overview

The remove operation removes a provider configuration from the list of configurations available for that provider in the Configuration service. The operation does not remove the configuration objects from the COM archive, but merely removes the objects from the Configuration service provider.

If the implementation of the Configuration service is not using a COM archive, then an error is returned.

This operation can be used to remove COM, non-COM, and hard-coded configurations from a provider.

Operation Identifier	remove	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Signature
IN	SUBMIT	serviceProvider : (COM::ObjectKey) configObjIds : (List<COM::ObjectId>)

3.4.12.2 Structures

3.4.12.2.1 If the implementation of the Configuration service is not using a COM archive, then an UNSUPPORTED_OPERATION error shall be returned.

3.4.12.2.2 The first argument shall contain the domain and object identifier of the ServiceProvider COM object from which the configurations are being removed.

3.4.12.2.3 The second argument shall contain a list of service and/or provider configurations to remove from the list of configurations available for the specific service provider.

3.4.12.2.4 If either the service provider or a provided object identifier does not match an existing configuration object, then this operation shall fail with an UNKNOWN error.

3.4.12.2.5 If any of the supplied configuration objects are not provider or service configuration objects, then an INVALID error shall be returned.

3.4.12.2.6 If an error is raised, then no configurations shall be removed as a result of this operation call.

3.4.12.2.7 Matched configuration objects shall not be removed from the COM archive, only from the list of configuration objects in the provider.

3.4.12.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN

1) Either the service provider or one of the supplied configuration object instance identifiers is unknown.

2) A list of the indexes of the error values is contained in the extra information field.

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

b) ERROR: INVALID

1) One of the supplied configuration object instance identifiers is either not a Service or not a Provider configuration object.

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

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

c) ERROR: UNSUPPORTED_OPERATION

The operation requires the use of a COM archive.

Error	Error #	ExtraInfo Type
UNSUPPORTED_OPERATION	Defined in MAL	Not Used

3.4.13 OPERATION: storeCurrent

3.4.13.1 Overview

The storeCurrent operation requests the creation of a new Configuration object containing the current configuration of a specific Service or Provider configuration and stores the new Configuration object in the COM archive. Optionally, the configuration can be added to the list of available configurations.

The actual service provider is responsible for the creation of the new Configuration objects in the COM archive, as it is the provider that contains the configuration being stored.

This operation's expected use is for storing a configuration that has been modified 'on line' in the service provider. For instance, a need to modify the thresholds of the parameter checks on ground can arise during operations. Before being 'officialised', the new thresholds are usually implemented locally and tested. If the modifications are deemed correct, then the configuration may be stored in the archive, and it becomes a new official configuration of the relevant service. This configuration can then be used by other providers of the same service via the add and activate operations.

If the implementation of the Configuration service is not using a COM archive, then an error is returned.

This operation can be used to request the store of both COM and non-COM configurations. It does not make sense to support hard-coded configurations, as by definition they are fixed in nature.

Operation Identifier	storeCurrent	
Interaction Pattern	INVOKE	
Pattern Sequence	Message	Body Signature
IN	INVOKE	serviceProvider : (COM::ObjectKey) serviceKey : (ServiceKey) autoAdd : (MAL::Boolean)
OUT	ACK	Empty
OUT	RESPONSE	objInstId : (COM::ObjectId)

3.4.13.2 Structures

3.4.13.2.1 If the implementation of the Configuration service is not using a COM archive, then an UNSUPPORTED_OPERATION error shall be returned.

3.4.13.2.2 The serviceProvider field shall contain the domain and object instance identifier of the ServiceProvider COM object that must store its current configuration.

3.4.13.2.3 If the service provider is not known, an UNKNOWN error shall be returned.

3.4.13.2.4 If the serviceKey field is not NULL, then only the specified service of the provider shall be stored.

3.4.13.2.5 Wildcard values of '0' are not accepted in the serviceKey fields; an INVALID error shall be returned if wildcard values of '0' appear in the serviceKey fields.

3.4.13.2.6 If the serviceKey field is not NULL and the referenced service is not supported by the service provider, an UNKNOWN error shall be returned.

3.4.13.2.7 The operation shall publish a ConfigurationStore event containing the selected configuration to be stored.

3.4.13.2.8 The service provider that implements the selected service shall, after the reception of the event, store its current Configuration in the COM Archive.

3.4.13.2.9 Once the relevant service provider has finished storing its configuration, it shall publish a ConfigurationStored event with the stored configuration's ObjectId as its body, or NULL if the store failed.

3.4.13.2.10 If the request is for a hard-coded configuration, then the relevant service provider must fail the store request by returning NULL as a response.

3.4.13.2.11 If the autoAdd field is set to TRUE, then once the stored event has been published, and if it indicates success, the Configuration service provider shall add the new configuration to the list of available configurations for the selected service provider, in effect equivalent to calling the add operation.

3.4.13.2.12 The response shall contain the object identifier of the new configuration object if successful, or NULL if not.

3.4.13.3 Errors

The operation may return one of the following errors:

a) ERROR: INVALID

   Not a valid Service or Provider object.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

b) ERROR: UNSUPPORTED_OPERATION

   The operation requires the use of a COM archive.

Error	Error #	ExtraInfo Type
UNSUPPORTED_OPERATION	Defined in MAL	Not Used

c) ERROR: UNKNOWN

The service provider referenced is not known, or the referenced service is not known by the provider.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

3.4.14 OPERATION: importXML

3.4.14.1 Overview

The importXML operation generates a new Configuration object from an XML file and stores the Configuration in the COM archive. Afterwards, the configuration can be added to the list of available configurations using the add operation.

The operation is only for importing XML configurations that use the standardised format defined by the Configuration service.

If the implementation of the Configuration service is not using a COM archive, then an error is returned.

The importXML can only be used to import COM-based configurations.

Operation Identifier	importXML	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Signature
IN	REQUEST	xmlFile : (MAL::File)
OUT	RESPONSE	objInstId : (COM::ObjectId)

3.4.14.2 Structures

3.4.14.2.1 If the implementation of the Configuration service is not using a COM archive, then an UNSUPPORTED_OPERATION error shall be returned.

3.4.14.2.2 The supplied file contained in the xmlFile argument shall be read and converted to COM objects.

3.4.14.2.3 If there is a problem converting the XML, then an INVALID error shall be returned.

3.4.14.2.4 For every object that is present within the XML file and that does not exist in the COM Archive, the Configuration service shall create a new object with the same content and store the object in the COM Archive.

3.4.14.2.5 If the object already exists in the COM Archive, nothing shall be created.

3.4.14.2.6 If the object already exists in the COM Archive but contains different content, a DUPLICATE error shall be raised.

3.4.14.2.7 The newly generated Configuration object shall always reference existing objects in the Archive.

3.4.14.2.8 The newly generated Configuration object should be checked for consistency. An INVALID error shall be raised if the configuration is not valid.

3.4.14.2.9 If an error is raised, then no objects shall be stored in the COM archive, and the operation shall end.

3.4.14.2.10 The return response shall contain in the objInstId field the object identifier of the new configuration object.

3.4.14.3 Errors

The operation may return one of the following errors:

a) ERROR: UNSUPPORTED_OPERATION

The operation requires the use of a COM archive.

Error	Error #	ExtraInfo Type
UNSUPPORTED_OPERATION	Defined in MAL	Not Used

b) ERROR: DUPLICATE

The supplied XML contains a duplicate object definition that is different from the one held in the COM Archive.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	Not Used

c) ERROR: INVALID

The supplied XML was not valid.

Error	Error #	ExtraInfo Type
INVALID	Defined in COM	Not Used

4 DATA TYPES

4.1 AREA DATA TYPES: COMMON

4.1.1 COMPOSITE: ServiceKey

The ServiceKey structure shall be used to hold information about a service.

Name	ServiceKey		
Extends	MAL::Composite		
Short Form Part	1		
Field	Type	Nullable	Comment
keyArea	MAL::UShort	No	The area of this service taken from the numeric Area identifier of the service specification.
keyService	MAL::UShort	No	The service taken from the numeric Service identifier of the service specification.
keyAreaVersion	MAL::UOctet	No	The version of this service taken from the Area Version of the area specification.

4.2 SERVICE DATA TYPES: DIRECTORY

4.2.1 COMPOSITE: ProviderDetails

4.2.1.1 The ProviderDetails structure shall be used to hold parameters providing information about a provider of a service and its capabilities.

4.2.1.2 The structure shall contain a list of AddressDetails structures, which should be used when the individual services listed by the provider do not supply address information.

NOTE – A provider may support more than one transport technology and therefore can be reached using more than one address.

Name	ProviderDetails		
Extends	MAL::Composite		
Short Form Part	1		
Field	Type	Nullable	Comment
serviceCapabilities	List<ServiceCapability>	No	The service capabilities supported by this service provider
providerAddresses	List<AddressDetails>	No	List of addresses for all services of this service provider unless service specific addresses are supplied in the serviceCapabilities field. If all address information is supplied in the serviceCapabilities field, this list should be zero length.

4.2.2 COMPOSITE: ServiceCapability

The ServiceCapability structure shall be used to hold parameters containing information about a service and the capabilities offered by a provider.

Name	ServiceCapability		
Extends	MAL::Composite		
Short Form Part	2		
Field	**Type**	**Nullable**	**Comment**
serviceKey	ServiceKey	No	The area, service, and version fields.
supportedCapabilitySets	List<MAL::UShort>	Yes	The supported capability set numbers for this service provider. If NULL, then all capability sets supported.
serviceProperties	List<MAL::NamedValue>	Yes	Allows the passing of deployment specific service properties.
serviceAddresses	List<AddressDetails>	Yes	Optional set of address details for this specific service, which shall be used instead of the provider ones when accessing this service. If all address information is supplied in the containing ProviderDetails structure field, this list should be replaced with a NULL.

4.2.3 COMPOSITE: AddressDetails

The AddressDetails structure shall be used to hold parameters containing all information required by the Directory service about a service provider's URI and attributes relating to QoS.

Name	AddressDetails		
Extends	MAL::Composite		
Short Form Part	4		
Field	Type	Nullable	Comment
supportedLevels	List<MAL::QoSLevel>	No	The set of possible QoS levels this service can provide.
QoSproperties	List<MAL::NamedValue>	No	Any QoS properties relevant to this address's URIs and the specified transport.
priorityLevels	MAL::UInteger	No	The number of QoS priority levels that this provider supports.
serviceURI	MAL::URI	Yes	The Service URI that identifies the physical location of this service. NULL if represents a shared data provider (Broker).
brokerURI	MAL::URI	Yes	The broker URI that identifies the physical location of the publish and subscribe interface. NULL if service does not use publish and subscribe operations or if a shared broker is to be used.
brokerProviderObjInstId	MAL::Long	Yes	The object instance identifier of a ServiceProvider COM object that is the shared broker used by this provider.

4.2.4 COMPOSITE: ProviderSummary

The ProviderSummary structure shall be used to hold parameters containing information about a provider of a service and its capabilities.

Name	ProviderSummary		
Extends	MAL::Composite		
Short Form Part	5		
Field	Type	Nullable	Comment
providerKey	COM::ObjectKey	No	The COM object key of this service provider.
providerId	MAL::Identifier	No	The id of this service provider.
providerDetails	ProviderDetails	No	The service capabilities supported by this provider.

4.2.5 COMPOSITE: PublishDetails

The PublishDetails structure shall be used to hold parameters containing all the information required to publish new service-provider details.

Name	PublishDetails		
Extends	MAL::Composite		
Short Form Part	6		
Field	Type	Nullable	Comment
providerId	MAL::Identifier	No	The unique service provider id; allows multiple service providers of the same service type to coexist in the directory service.
domain	List<MAL::Identifier>	No	The domain of the provider.
sessionType	MAL::SessionType	No	The type of session of the provider.
sourceSessionName	MAL::Identifier	Yes	If this is part of a replay session, this field holds the session name of the source session. NULL otherwise.
network	MAL::Identifier	No	The network of the provider.
providerDetails	ProviderDetails	No	The new service provider details.
serviceXML	List<MAL::File>	Yes	The optional XML files to associate with this provider.

4.2.6 COMPOSITE: ServiceFilter

The ServiceFilter structure shall be used to hold parameters containing all information required by the Directory service for service lookup operation. The field filters shall be AND'd together.

Name	ServiceFilter		
Extends	MAL::Composite		
Short Form Part	7		
Field	Type	Nullable	Comment
serviceProviderId	MAL::Identifier	Yes	The required service provider. Can be NULL, in which case matches all values.
domain	List<MAL::Identifier>	Yes	The domain to query. Can be NULL, in which case matches all values.
network	MAL::Identifier	Yes	The network to match. Can be NULL, in which case matches all values.
sessionType	MAL::SessionType	Yes	The session type to match. Can be NULL, in which case matches all values.
sessionName	MAL::Identifier	Yes	The session name to match. Can be NULL, in which case matches all values.
serviceKey	ServiceKey	Yes	The service to filter on. Values can be NULL, which matches all values.
requiredCapabilitySets	List<MAL::UShort>	Yes	List of required capability sets. If NULL, then matches any.

4.3 SERVICE DATA TYPES: LOGIN

4.3.1 COMPOSITE: Profile

The Profile structure shall be used to contain details of the user who is logging on to take a specified role.

Name	Profile		
Extends	MAL::Composite		
Short Form Part	1		
Field	Type	Nullable	Comment
username	MAL::Identifier	No	The name of the user.
role	MAL::Long	Yes	The optional object instance identifier of the role required by the user.

4.4 SERVICE DATA TYPES: CONFIGURATION

4.4.1 ENUMERATION: ConfigurationType

The ConfigurationType enumeration shall be used to hold the possible types of a configuration.

Name	ConfigurationType	
Short Form Part	4	
Enumeration Value	Numerical Value	Comment
PROVIDER	1	The configuration is a Provider configuration.
SERVICE	2	The configuration is a Service configuration.

4.4.2 COMPOSITE: ConfigurationObjectset

The ConfigurationObjectSet structure shall be used to hold a set of object identifiers for a single COM object type in a single domain.

Name	ConfigurationObjectSet		
Extends	MAL::Composite		
Short Form Part	1		
Field	Type	Nullable	Comment
objType	COM::ObjectType	No	The COM object type of the configuration objects.
domain	List<MAL::Identifier>	No	The domain of the configuration objects.
objInstIds	List<MAL::Long>	No	The set of COM object identifiers that form this configuration set.

4.4.3 COMPOSITE: ConfigurationObjectDetails

4.4.3.1 The ConfigurationObjectDetails composite shall be used to hold zero to many ConfigurationObjectSet structures.

4.4.3.2 It shall allow a configuration to reference COM objects from more than one domain or of more than one COM object type.

Name	ConfigurationObjectDetails		
Extends	MAL::Composite		
Short Form Part	2		
Field	Type	Nullable	Comment
configObjects	List<ConfigurationObjectSet>	No	The list of configuration objects.

4.4.4 COMPOSITE: ServiceConfigurationIdentifier

The ServiceConfigurationIdentifier structure shall be used to hold the name and service key of a service configuration object.

Name	ServiceConfigurationIdentifier		
Extends	MAL::Composite		
Short Form Part	3		
Field	**Type**	**Nullable**	**Comment**
configName	MAL::Identifier	No	The name of the service configuration.
serviceKey	ServiceKey	No	The service key of the service configuration.

5 SERVICE SPECIFICATION XML

5.1 OVERVIEW

This section provides a link to definition of the service in XML notation as specified in 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 at:

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

5.2 NORMATIVE XML SERVICE SPECIFICATION

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

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

ANNEX A

PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

(NORMATIVE)

A1 INTRODUCTION

A1.1 OVERVIEW

This annex provides the Protocol Implementation Conformance Statement (PICS) Requirements List (RL) for an implementation of the Mission Operations Common Services standard. The PICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation claiming conformance must satisfy the mandatory requirements referenced in the RL.

An implementation's completed RL is called the PICS. The PICS states which protocol features have been implemented. The following entities can use the PICS:

- the protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;
- the supplier and acquirer or potential acquirer of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
- the user or potential user of the implementation, as a basis for initially checking the possibility of interworking with another implementation (while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSes);
- a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A1.2 NOTATION

A1.2.1 Status Column Symbols

The following are used in the RL to indicate the status of features:

Symbol	Meaning
M	Mandatory
O	Optional

A1.2.2 Support Column Symbols

The support of every item as claimed by the implementer is stated by entering the appropriate answer (Y, N, or N/A) in the support column.

Symbol	Meaning
Y	Yes, supported by the implementation
N	No, not supported by the implementation
N/A	Not applicable

A2 GENERAL INFORMATION

A2.1 IDENTIFICATION OF PICS

Ref	Question	Response
1	Date of Statement (DD/MM/YYYY)	
2	CCSDS document number containing the PICS	
3	Date of CCSDS document containing the PICS	

A2.2 IDENTIFICATION OF IMPLEMENTATION UNDER TEST (IUT)

Ref	Question	Response
1	Implementation name	
2	Implementation version	
3	Machine name	
4	Machine version	
5	Operating System name	
6	Operating System version	
7	Special Configuration	
8	Other Information	

A2.3 USER IDENTIFICATION

Supplier	
Contact Point for Queries	
Implementation name(s) and Versions	
Other Information Necessary for full identification —e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)	

A2.4 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the protocol by completing the RL; the resulting completed RL is called a PICS.

A3 MO M&C SERVICES PICS

Item	Protocol Feature	Reference	Status	Support
	Directory service			
1-1	Capability Set 1	3.2.7	M	
1-2	Capability Set 2	3.2.8 – 3.2.10	O	
	Login Service			
2-1	Capability Set 1	3.3.7 and 3.3.8	M	
2-2	Capability Set 2	3.3.9	O	
2-3	Capability Set 2	3.3.10	O	
	Configuration service			
3-1	Capability Set 1	3.4.7	M	
3-2	Capability Set 2	3.4.8 and 3.4.9	O	
3-3	Capability Set 3	3.4.10	O	
3-4	Capability Set 4	3.4.11 and 3.4.12	O	
3-5	Capability Set 5	3.4.13	O	
3-6	Capability Set 6	3.4.14	O	

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

The security considerations of this specification are the same as those of reference [2]. Specifically, authentication and authorisation of a participating consumer or provider is provided by the MAL access control concept and is covered in subsections 3.6, 5.2, and 5.3 of the Reference Model (reference [1]).

Security of a communications link is delegated to the transport layer.

B2 SANA CONSIDERATIONS

The recommendations of this document request SANA populate the registry specified in reference [2] with the schema and XML detailed in section 5 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, the Area Director.

B3 PATENT CONSIDERATIONS

The recommendations of this document have no patent issues.

ANNEX C

EXAMPLE USE CASES

(INFORMATIVE)

C1 INTRODUCTION

C1.1 OVERVIEW

This annex provides four example uses of the Configuration service. This is not an exhaustive list of possible uses; however, it comprises the most common expected uses of the Configuration service.

These are example uses of the Configuration service; they are not prescriptive. Users/implementers of the service can use the service as they deem best or appropriate for their deployments, agencies, or needs.

C1.2 OBTAINING A PROVIDER CONFIGURATION

To obtain a provider configuration, a consumer of the Configuration service will first need to locate the Configuration service, possibly using the Directory service. Next, it will get the COM object identifier of the current configuration of the required service provider(s) from Configuration service using the getCurrent operation.

The actual configuration is then retrieved from either the archive (using the COM archive operations) or requested as an XML file (using the Configuration service exportXML operation). It should be noted that this example requires the use of a COM archive.

Choices for retrieval are as a large XML file containing references and parameter definition values, a small XML file containing just references, or a set of COM object references from the COM archive. The two alternatives, XML or COM objects, are shown in the diagram as wrapped in ‘alt’ boxes.

The sequence for this use case is shown in figure C-1 below:

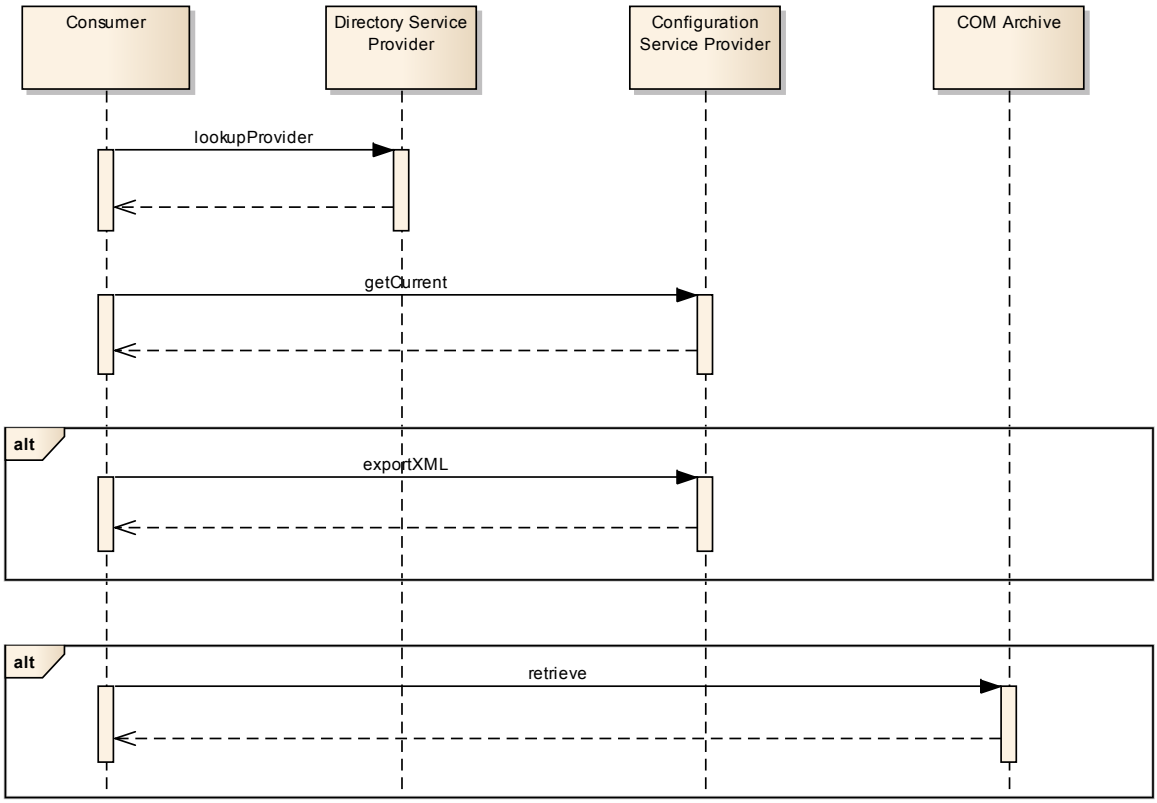

Figure C-1: Obtaining a Service Provider Configuration Example Sequence

C1.3 RECONFIGURING A SERVICE PROVIDER

To reconfigure a service provider, a consumer of the Configuration service will first need to locate the Configuration service, possibly using the Directory service.

Next, it will obtain a list of possible Provider Configurations from the Configuration service provider, including which one is currently active (list operation or/and getCurrent) for the service provider that it wishes to switch the configuration of.

NOTE – To use the getCurrent alternative (as shown by the 'alt' wrapping box), a COM archive is required.

Then the consumer should select one from the list and call activate on Configuration service. The activate request results in publication of a COM event announcing the configuration change request (ConfigurationSwitch event). The COM event uses the related link to give the service provider id of the service provider required to be switched.

The referenced service provider is expected to publish a matching event giving their success or failure result (ConfigurationSwitched event). How the referenced service provider obtains

the selected configuration is deployment dependent; for example, it may be using a COM archive and would be able to load the requested configuration from there.

The sequence for this use case is shown in figure C-2 below:

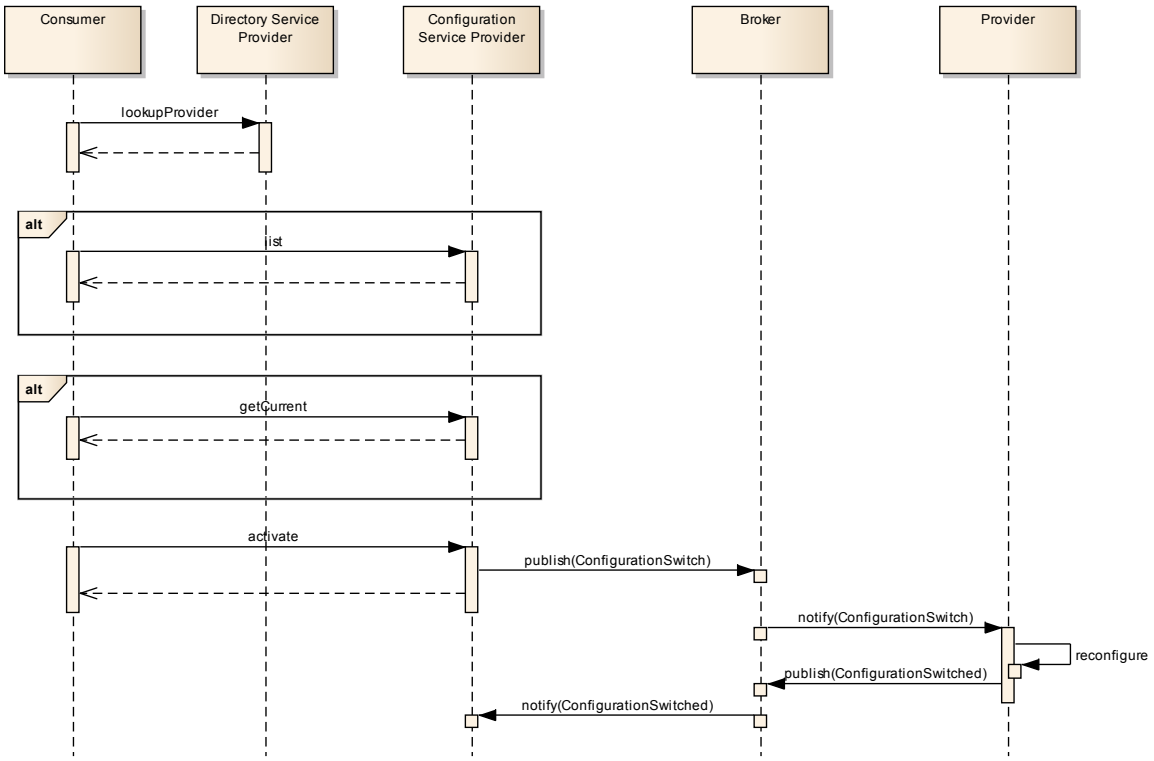

Figure C-2: Reconfiguring a Service Provider Example Sequence

C1.4 SERVICE START-UP IN A PROVIDER

The current configuration of a service provider is most likely the configuration that was last activated using the activate operation; however, deployments may have other requirements or means (outside the scope of this service) for determining the current configuration of a service provider at start up. For example, a deployment of a service provider may always default to a known or specific configuration rather than use the last activated configuration.

It is also possible that in a deployment a provider may start up and remain in an un-initialised state (without configuration), in which case it may require an external entity to call ‘activate’ to switch it to the required configuration. The external entity may also use ‘getCurrent’ to determine which configuration is best for the ‘activate’ call; however, that is deployment and implementation specific.

In this example use, the ‘current’ configuration is the configuration that is returned by the getCurrent operation.

At start up, the service provider will first need to locate the Configuration service, possibly using the Directory service. Then, in this example, it will obtain its last active Provider Configuration from the Configuration service provider using the getCurrent operation. Finally, the service provider will retrieve the actual configuration information from the COM archive and configure itself accordingly.

The sequence for this use case is shown in figure C-3 below:

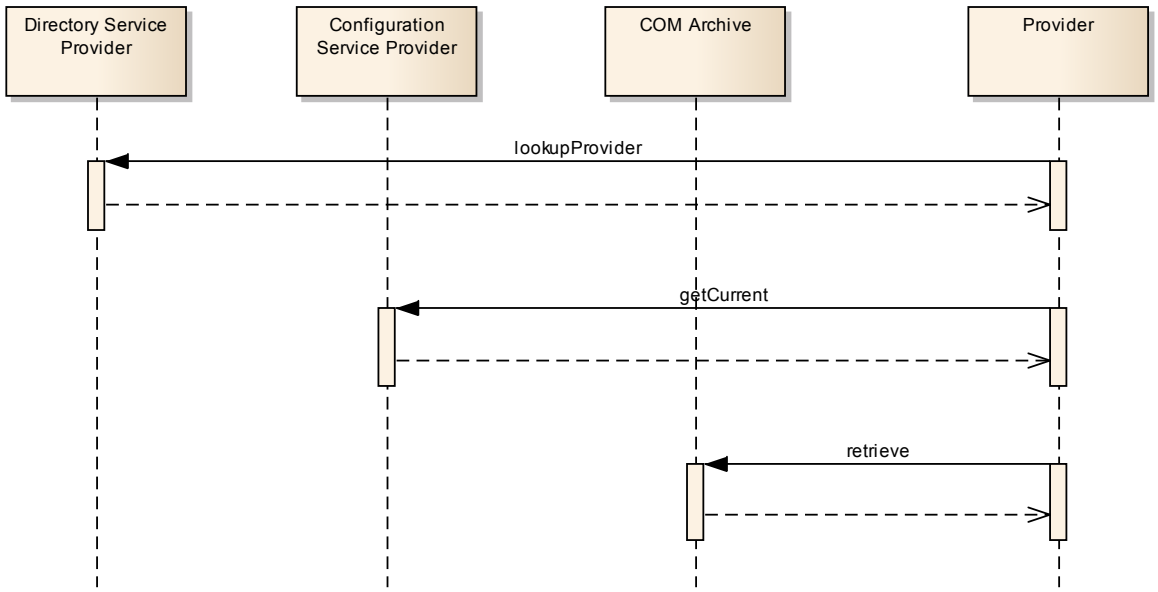

Figure C-3: Service Start Up in Provider Example Sequence

C1.5 STORING A SERVICE PROVIDER CONFIGURATION

To store the current configuration of a service provider, a consumer of the Configuration service will first need to locate the Configuration service, possibly using the Directory service.

Then it will call storeCurrent on the Configuration service provider passing in the provider object identifier of the service provider that should store its current configuration, that is, let it know which provider service to persist the configuration of.

The storeCurrent request results in publication of a COM event announcing the configuration store request (ConfigurationStore event), the related link of the event contains the service provider identifier.

The referenced provider is expected to store its configuration objects to the COM archive and then publish a matching event giving its success or failure result of the store attempt (ConfigurationStored event). This is the first COM Archive store operation shown; it contains the relevant configuration of the provider.

Upon reception of the store result event, the Configuration service provider creates in the COM archive a new configuration object that links to the new configuration of the service provider. The actual configuration objects are not duplicated, as they are already in the archive; they are just referenced by the Configuration object. This is the second COM Archive store operation shown and is the store of the object that links the configuration created to the provider to the list of available configurations for that provider.

Finally, the new configuration is added to the list of available configurations for that service provider.

The sequence for this use case is shown in figure C-4 below:

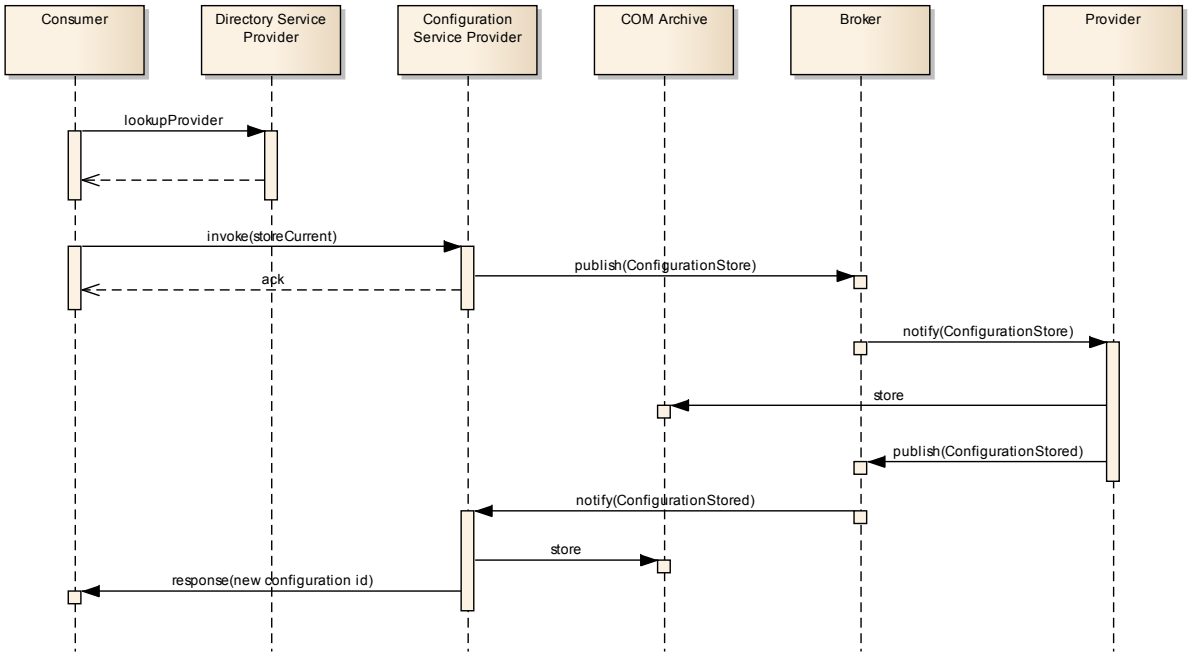

Figure C-4: Storing a Provider Configuration Example Sequence

ANNEX D

DEFINITION OF ACRONYMS

(INFORMATIVE)

API	application program interface
AMS	CCSDS Asynchronous Messaging System
CCS	central checkout system
CCSDS	Consultative Committee for Space Data Systems
COM	Common Object Model
MAL	Message Abstraction Layer
M&C	monitor and control
MCS	mission control system
MO	Mission Operations
PICS	protocol implementation conformance statement
QoS	quality of service
RL	requirements list
SDU	service data unit
SPP	CCSDS Space Packet Protocol
UML	Unified Modelling Language
URI	uniform resource identifier
XML	eXtensible Markup Language

ANNEX E

INFORMATIVE REFERENCES

(INFORMATIVE)

[E1] *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 contained in 1.9.

ANNEX F

CONFIGURATION SERVICE XML FORMAT

(INFORMATIVE)

F1 INTRODUCTION

This annex provides example XML fragments showing the XML format for provider or service configurations. These XML fragments can be imported and exported by the Configuration service's importXml and exportXml operations as defined in 3.4 of the Configuration service specification.

The structure of the XML is based on *Mission Operations—Message Abstraction Layer Binding to HTTP Transport and XML Encoding* (reference [4]).

The XML definitions come in two forms as described in 3.4. The first is a complete version in which the underlying objects defining the configuration are included in the XML document, and the second is a compact version in which the underlying objects are not included as part of the document. Each XML document may be generated at the provider level or the service level.

The following examples illustrate each of these cases.

F1.1 EXAMPLE XML: SERVICE LEVEL – COMPACT

```
<cp:Configuration xmlns:cp="CommonPrototype">
 <InlineServiceConfiguration>
  <ObjectId>
   <ObjectKey>
    <Domain>
     <MAL:IdentifierList>
      <MAL:Identifier>domain</MAL:Identifier>
     </MAL:IdentifierList>
    </Domain>
    <InstanceId>2</InstanceId>
   </ObjectKey>
   <ObjectType area="3" number="1" service="5" version="1" />
  </ObjectId>
  <ObjectBody>
   <ServiceKey keyArea="211" keyAreaVersion="1" keyService="2" />
  </ObjectBody>
  <ObjectDetails>
   <InlineConfigurationObjects>
    <ObjectId>
     <ObjectKey>
      <Domain>
       <MAL:IdentifierList>
        <MAL:Identifier>domain</MAL:Identifier>
       </MAL:IdentifierList>
      </Domain>
```

```
      <InstanceId>7</InstanceId>
     </ObjectKey>
     <ObjectType area="3" number="3" service="5" version="1" />
    </ObjectId>
    <ObjectBody>
     <InlineConfigurationObjectDetails>
      <configObjects>
       <InlineConfigurationObjectSet>
        <ObjectType area="211" number="1" service="2" version="1" />
        <Domain>
         <MAL:IdentifierList>
          <MAL:Identifier>domain</MAL:Identifier>
         </MAL:IdentifierList>
        </Domain>
        <ObjectIds>
         <InstanceId>3</InstanceId>
        </ObjectIds>
       </InlineConfigurationObjectSet>
      </configObjects>
     </InlineConfigurationObjectDetails>
    </ObjectBody>
   </InlineConfigurationObjects>
  </ObjectDetails>
 </InlineServiceConfiguration>
</cp:Configuration>
```

F1.2 EXAMPLE XML: SERVICE LEVEL – COMPLETE

```
<cp:Configuration xmlns:cp="CommonPrototype">
 <InlineServiceConfiguration>
  <ObjectId>
   <ObjectKey>
    <Domain>
     <MAL:IdentifierList>
      <MAL:Identifier>domain</MAL:Identifier>
     </MAL:IdentifierList>
    </Domain>
    <InstanceId>2</InstanceId>
   </ObjectKey>
   <ObjectType area="3" number="1" service="5" version="1" />
  </ObjectId>
  <ObjectBody>
   <ServiceKey keyArea="211" keyAreaVersion="1" keyService="2" />
  </ObjectBody>
  <ObjectDetails>
   <InlineConfigurationObjects>
    <ObjectId>
     <ObjectKey>
      <Domain>
       <MAL:IdentifierList>
        <MAL:Identifier>domain</MAL:Identifier>
       </MAL:IdentifierList>
      </Domain>
      <InstanceId>7</InstanceId>
     </ObjectKey>
     <ObjectType area="3" number="3" service="5" version="1" />
    </ObjectId>
    <ObjectBody>
```

```
     <InlineConfigurationObjectDetails>
      <configObjects>
       <InlineConfigurationObjectSet>
        <ObjectType area="211" number="1" service="2" version="1" />
        <Domain>
         <MAL:IdentifierList>
          <MAL:Identifier>domain</MAL:Identifier>
         </MAL:IdentifierList>
        </Domain>
        <ObjectIds>
         <InstanceId>3</InstanceId>
        </ObjectIds>
        <objectBodies>
         <Parameter name="testParameter" value="7" />
        </objectBodies>
       </InlineConfigurationObjectSet>
      </configObjects>
     </InlineConfigurationObjectDetails>
    </ObjectBody>
   </InlineConfigurationObjects>
  </ObjectDetails>
 </InlineServiceConfiguration>
</cp:Configuration>
```

F1.3 EXAMPLE XML: PROVIDER LEVEL COMPLETE

```
<Configuration xmlns="CommonPrototype">
 <InlineProviderConfiguration>
  <ObjectId>
   <ObjectKey>
    <Domain>
     <MAL:IdentifierList>
      <MAL:Identifier>domain</MAL:Identifier>
     </MAL:IdentifierList>
    </Domain>
   </ObjectKey>
   <ObjectType area="3" service="5" version="1" number="2" />
  </ObjectId>
  <ObjectBody>
   <MAL:Identifier>pConf8</MAL:Identifier>
  </ObjectBody>
  <ObjectDetails>
   <InlineConfigurationObjects>
    <ObjectId>
     <ObjectKey>
      <Domain>
       <MAL:IdentifierList>
        <MAL:Identifier>domain</MAL:Identifier>
       </MAL:IdentifierList>
      </Domain>
     </ObjectKey>
     <ObjectType area="3" service="5" version="1" number="3" />
    </ObjectId>
    <ObjectBody>
     <InlineConfigurationObjectDetails>
      <configObjects>
       <InlineConfigurationObjectSet>
        <ObjectType xmlns="COM" area="211" service="2"
```

```
          version="1" number="1" />
         <Domain xmlns="COM">
          <MAL:IdentifierList>
           <MAL:Identifier>domain</MAL:Identifier>
          </MAL:IdentifierList>
         </Domain>
         <objectBodies>
          <Parameter name="testParameter" value="8" />
         </objectBodies>
        </InlineConfigurationObjectSet>
       </configObjects>
      </InlineConfigurationObjectDetails>
     </ObjectBody>
    </InlineConfigurationObjects>
   </ObjectDetails>
  </InlineProviderConfiguration>
</Configuration>
```