BS ISO 20210:2015

BSI Standards Publication

# Space data and information transfer systems — Mission Operations Message Abstraction Layer - JAVA API

bsi.

...making excellence a habit.™

Document Sharing Hub

## National foreword

This British Standard is the UK implementation of ISO 20210:2015.

The UK participation in its preparation was entrusted to Technical Committee ACE/68, Space systems and operations.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2015.
Published by BSI Standards Limited 2015

ISBN 978 0 580 87910 4

ICS 49.140

**Compliance with a British Standard cannot confer immunity from legal obligations.**

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 December 2015.

**Amendments/corrigenda issued since publication**

| Date | Text affected |
|---|---|

Documents
Document Sharing Hub

Full document details: BS ISO 20210:2015

INTERNATIONAL STANDARD

# ISO 20210

First edition
2015-08-15

# Space data and information transfer systems — Mission Operations Message Abstraction Layer - JAVA API

*Systèmes de transfert des informations et données spatiales — Couche d'abstraction des messages des opérations de mission — JAVA API*

Reference number
ISO 20210:2015(E)

Documents
Document Sharing Hub
© ISO 2015

Full document details: BS ISO 20210:2015

**COPYRIGHT PROTECTED DOCUMENT**

© ISO 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org


© ISO 2015 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO 20210:2015

# Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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

ISO 20210 was prepared by the Consultative Committee for Space Data Systems (CCSDS) (as CCSDS 523.1-M-1, April 2013) and was adopted (without modifications except those stated in clause 2 of this International Standard) by Technical Committee ISO/TC 20, *Aircraft and space vehicles*, Subcommittee SC 13, *Space data and information transfer systems*.

© ISO 2015 – All rights reserved


Documents
Document Sharing Hub
Full document details: BS ISO 20210:2015

Documents
Document Sharing Hub

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Practices

# MISSION OPERATIONS MESSAGE ABSTRACTION LAYER—JAVA API

## RECOMMENDED PRACTICE

## CCSDS 523.1-M-1

## MAGENTA BOOK

April 2013

Documents
Document Sharing Hub

Full document details: BS ISO 20210:2015

# AUTHORITY

| | |
|---|---|
| Issue: | Recommended Practice, Issue 1 |
| Date: | April 2013 |
| Location: | Washington, DC, USA |

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-3), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

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



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# STATEMENT OF INTENT

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

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

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

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



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# FOREWORD

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

http://www.ccsds.org/

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



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

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

Member Agencies

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

Observer Agencies

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



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# DOCUMENT CONTROL

| Document | Title | Date | Status |
|---|---|---|---|
| CCSDS 523.1-M-1 | Mission Operations Message Abstraction Layer—JAVA API, Recommended Practice, Issue 1 | April 2013 | Current issue |


© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS

Section Page

**1 INTRODUCTION ..... 1-1**

1.1 PURPOSE OF THIS RECOMMENDED PRACTICE ..... 1-1
1.2 SCOPE ..... 1-1
1.3 APPLICABILITY ..... 1-1
1.4 RATIONALE ..... 1-1
1.5 DOCUMENT STRUCTURE ..... 1-1
1.6 DEFINITIONS ..... 1-2
1.7 CONVENTIONS ..... 1-2
1.8 REFERENCES ..... 1-4

**2 OVERVIEW ..... 2-1**

2.1 GENERAL ..... 2-1
2.2 MO SERVICE FRAMEWORK JAVA MAPPING ..... 2-3
2.3 MAPPING FROM MAL DOCUMENT ..... 2-5

**3 MAL API ..... 3-1**

3.1 GENERAL ..... 3-1
3.2 MAL PACKAGE ..... 3-3
3.3 DATA STRUCTURES PACKAGE ..... 3-40
3.4 CONSUMER PACKAGE ..... 3-58
3.5 PROVIDER PACKAGE ..... 3-94
3.6 BROKER PACKAGE ..... 3-134

**4 SERVICE MAPPING ..... 4-1**

4.1 OVERVIEW ..... 4-1
4.2 DEFINITION ..... 4-1
4.3 CONSUMER ..... 4-8
4.4 PROVIDER ..... 4-25
4.5 DATA STRUCTURES ..... 4-40
4.6 ELEMENT FACTORY CLASSES ..... 4-50
4.7 MULTIPLE ELEMENT BODY CLASSES ..... 4-51
4.8 HELPER AND ELEMENT FACTORY CLASSES ..... 4-53

**5 TRANSPORT API ..... 5-1**

5.1 GENERAL ..... 5-1
5.2 CLASSES AND INTERFACES ..... 5-1



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS (continued)

Section Page

**6 ACCESS CONTROL API ..... 6-1**

6.1 GENERAL ..... 6-1
6.2 CLASSES AND INTERFACES ..... 6-1

**7 ENCODING API ..... 7-1**

7.1 OVERVIEW ..... 7-1
7.2 CLASSES AND INTERFACES ..... 7-1

**ANNEX A DEFINITION OF ACRONYMS (INFORMATIVE) ..... A-1**
**ANNEX B INFORMATIVE REFERENCES (INFORMATIVE) ..... B-1**
**ANNEX C SECURITY, SANA, AND PATENT CONSIDERATIONS (INFORMATIVE) ..... C-1**
**ANNEX D CODE EXAMPLE (INFORMATIVE) ..... D-1**

Figure

2-1 Mission Operations Services Concept Document Set ..... 2-1
2-2 Relationship of MO Books ..... 2-2
2-3 Overview of the Mission Operations Service Framework ..... 2-3
2-4 MO Framework Java Mapping ..... 2-4
3-1 Relationships between the API Main Interfaces ..... 3-2
4-1 Relationships between the Stub Classes and Interfaces ..... 4-8
4-2 Relationships between the Skeleton Classes and Interfaces (Delegation Pattern) ..... 4-26
4-3 Relationships between the Skeleton Classes and Interfaces (Inheritance Pattern) ..... 4-26
4-4 Multi-Binding Service Provider ..... 4-27

Table

1-1 Variable Value Case Rules ..... 1-4
3-1 API Main Interfaces ..... 3-1
3-2 MALContextFactory ‘registerFactoryClass’ Parameter ..... 3-4
3-3 MALContextFactory ‘deregisterFactoryClass’ Parameter ..... 3-4
3-4 MALContextFactory ‘createMALContext’ Parameter ..... 3-6
3-5 MALContextFactory ‘registerArea’ Parameter ..... 3-6
3-6 MALContextFactory ‘lookupArea’ Parameters ..... 3-7
3-7 MALContextFactory ‘registerError’ Parameters ..... 3-8
3-8 MALContextFactory ‘lookupError’ Parameter ..... 3-8
3-9 MALContext ‘getTransport’ Parameters ..... 3-11



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS (continued)

Table Page

3-10 MALService Attributes ..... 3-12
3-11 MALService Constructor Parameters ..... 3-13
3-12 MALService 'setArea' Parameter ..... 3-15
3-13 MALService 'addOperation' Parameter ..... 3-15
3-14 MALOperation Attributes ..... 3-16
3-15 MALOperation Constructor Parameters ..... 3-16
3-16 MALOperation 'getOperationStage' Parameter ..... 3-18
3-17 MALOperation 'setService' Parameter ..... 3-19
3-18 Interaction Stages Constants ..... 3-20
3-19 Operation stages ..... 3-21
3-20 MAL<<Ip>>Operation Constructor Parameters ..... 3-21
3-21 MALOperationStage Attributes ..... 3-23
3-22 MALOperationStage Constructor Parameters ..... 3-24
3-23 MALOperation 'setOperation' Parameter ..... 3-25
3-24 MALService Attributes ..... 3-26
3-25 MALArea Constructor Parameters ..... 3-26
3-26 MALArea 'addService' Parameter ..... 3-27
3-27 MALService Constructor Parameters ..... 3-28
3-28 MALException Constructor Parameters ..... 3-30
3-29 MALInteractionException Constructor Parameter ..... 3-31
3-30 MALElementFactoryRegistry 'registerElementFactory' Parameters ..... 3-32
3-31 MALElementFactoryRegistry 'lookupElementFactory' Parameter ..... 3-33
3-32 MALElementFactoryRegistry 'deregisterElementFactory' Parameter ..... 3-33
3-33 MALEncoder and MALDecoder Variables ..... 3-34
3-34 MALEncoder 'encode[Nullable]<<Attribute>>' Parameter ..... 3-35
3-35 MALListEncoder 'createListEncoder' Parameter ..... 3-36
3-36 MALListDecoder 'createListDecoder' Parameter ..... 3-36
3-37 MALEncoder 'encode[Nullable]Element' Parameter ..... 3-37
3-38 MALDecoder 'decode[Nullable]Element' Parameter ..... 3-37
3-39 MALEncoder 'encode[Nullable]Attribute' Parameter ..... 3-38
3-40 Element 'encode' Parameter ..... 3-41
3-41 Element 'decode' Parameter ..... 3-42
3-42 Enumeration Constructor Parameter ..... 3-44
3-43 MAL::Attribute Types Mapped to a Java Type ..... 3-45
3-44 Union Variables ..... 3-46
3-45 Union Constructor Parameter ..... 3-46
3-46 Blob Byte Array Constructor Parameter ..... 3-49
3-47 Blob URL Constructor Parameter ..... 3-50
3-48 MAL::Attribute Types Represented by a Specific Class ..... 3-53
3-49 Initial Value Assigned by the <<Attribute>> Empty Constructor ..... 3-54
3-50 MALConsumerManager 'createConsumer' Parameters ..... 3-59



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS (continued)

Table Page

3-51 QoS Properties ..... 3-60
3-52 MALConsumer 'send' Parameters ..... 3-62
3-53 MALConsumer 'submit' Parameters ..... 3-63
3-54 MALConsumer 'request' Parameters ..... 3-64
3-55 MALConsumer 'invoke' Parameters ..... 3-66
3-56 MALConsumer 'progress' Parameters ..... 3-67
3-57 MALConsumer 'register' Parameters ..... 3-68
3-58 MALConsumer 'deregister' Parameters ..... 3-70
3-59 MALConsumer 'asyncSubmit' Parameters ..... 3-71
3-60 MALConsumer 'asyncRequest' Parameters ..... 3-72
3-61 MALConsumer 'asyncInvoke' Parameters ..... 3-73
3-62 MALConsumer 'asyncProgress' Parameters ..... 3-75
3-63 MALConsumer 'asyncRegister' Parameters ..... 3-76
3-64 MALConsumer 'asyncDeregister' Parameters ..... 3-77
3-65 MALConsumer 'continueInteraction' Parameters ..... 3-78
3-66 MALConsumer 'setTransmitErrorListener' Parameter ..... 3-79
3-67 MALInteractionListener 'submitAckReceived' Parameters ..... 3-81
3-68 MALInteractionListener 'submitErrorReceived' Parameters ..... 3-81
3-69 MALInteractionListener 'requestResponseReceived' Parameters ..... 3-82
3-70 MALInteractionListener 'requestErrorReceived' Parameters ..... 3-83
3-71 MALInteractionListener 'invokeAckReceived' Parameters ..... 3-83
3-72 MALInteractionListener 'invokeAckErrorReceived' Parameters ..... 3-84
3-73 MALInteractionListener 'invokeResponseReceived' Parameters ..... 3-85
3-74 MALInteractionListener 'invokeResponseErrorReceived' Parameters ..... 3-85
3-75 MALInteractionListener 'progressAckReceived' Parameters ..... 3-86
3-76 MALInteractionListener 'progressAckErrorReceived' Parameters ..... 3-87
3-77 MALInteractionListener 'progressUpdateReceived' Parameters ..... 3-87
3-78 MALInteractionListener 'progressUpdateErrorReceived' Parameters ..... 3-88
3-79 MALInteractionListener 'progressResponseReceived' Parameters ..... 3-89
3-80 MALInteractionListener 'progressResponseErrorReceived' Parameters ..... 3-89
3-81 MALInteractionListener 'registerAckReceived' Parameters ..... 3-90
3-82 MALInteractionListener 'registerErrorReceived' Parameters ..... 3-91
3-83 MALInteractionListener 'notifyReceived' Parameters ..... 3-91
3-84 MALInteractionListener 'notifyErrorReceived' Parameters ..... 3-92
3-85 MALInteractionListener 'deregisterAckReceived' Parameters ..... 3-92
3-86 MALProviderManager 'createProvider' Parameters ..... 3-95
3-87 MALProvider 'createPublisher' Parameters ..... 3-99
3-88 MALProvider 'setTransmitErrorListener' Parameter ..... 3-100
3-89 MALInteractionHandler 'malInitialize' Parameter ..... 3-103
3-90 MALInteractionHandler 'handleSend' Parameters ..... 3-103
3-91 MALInteractionHandler 'handleSubmit' Parameters ..... 3-104



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS (continued)

Table Page

3-92 MALInteractionHandler 'handleRequest' Parameters ..... 3-105
3-93 MALInteractionHandler 'handleInvoke' Parameters ..... 3-105
3-94 MALInteractionHandler 'handleProgress' Parameters ..... 3-106
3-95 MALInteractionHandler 'malFinalize' Parameter ..... 3-107
3-96 MALInteractionHandler 'get/setQoSProperty' Parameters ..... 3-108
3-97 MALSubmit 'sendError' Parameter ..... 3-109
3-98 MALResponse 'sendResponse' Parameter ..... 3-111
3-99 MALResponse 'sendError' Parameter ..... 3-112
3-100 MALInvoke 'sendAcknowledgement' Parameters ..... 3-113
3-101 MALProgress 'sendUpdate' Parameter ..... 3-115
3-102 MALProgress 'sendUpdateError' Parameter ..... 3-116
3-103 MALPublisher 'publish' Parameters ..... 3-117
3-104 MALPublisher 'register' Parameters ..... 3-118
3-105 MALPublisher 'asyncRegister' Parameters ..... 3-120
3-106 MALPublisher 'asyncDeregister' Parameter ..... 3-121
3-107 MALPublishInteractionListener 'publishRegisterAckReceived' Parameters ..... 3-123
3-108 MALPublishInteractionListener 'publishRegisterErrorReceived' Parameters ..... 3-123
3-109 MALPublishInteractionListener 'publishErrorReceived' Parameters ..... 3-124
3-110 MALPublishInteractionListener 'publishDeregisterAckReceived' Parameters ..... 3-125
3-111 MALProviderSet Constructor Parameter ..... 3-125
3-112 MALProviderSet 'createPublisherSet' Parameters ..... 3-126
3-113 MALProviderSet 'addProvider' Parameter ..... 3-127
3-114 MALProviderSet 'removeProvider' Parameter ..... 3-128
3-115 MALPublisherSet Constructor Parameters ..... 3-129
3-116 MALPublisherSet 'createPublisher' Parameter ..... 3-129
3-117 MALPublisherSet 'deletePublisher' Parameter ..... 3-130
3-118 MALPublisherSet 'publish' Parameters ..... 3-131
3-119 MALPublisherSet 'register' Parameters ..... 3-131
3-120 MALPublisherSet 'asyncRegister' Parameters ..... 3-132
3-121 MALPublisherSet 'asyncDeregister' Parameter ..... 3-133
3-122 MALBrokerManager 'createBroker' Parameter ..... 3-135
3-123 MALBrokerBinding 'createBrokerBinding' Parameters ..... 3-136
3-124 MALBrokerBinding 'sendNotify' Parameters ..... 3-139
3-125 MALBrokerBinding 'sendNotifyError' Parameters ..... 3-140
3-126 MALBrokerBinding 'sendPublishError' Parameters ..... 3-142
3-127 MALBrokerBinding 'setTransmitErrorListener' Parameter ..... 3-143
3-128 MALBrokerHandler 'malInitialize' Parameter ..... 3-145
3-129 MALBrokerHandler 'handleRegister' Parameters ..... 3-145
3-130 MALBrokerHandler 'handlePublishRegister' Parameters ..... 3-146
3-131 MALBrokerHandler 'handlePublish' Parameters ..... 3-147
3-132 MALBrokerHandler 'handleDeregister' Parameters ..... 3-148



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS (continued)

Table Page

3-133 MALBrokerHandler 'handlePublishDeregister' Parameters ..... 3-148
3-134 MALBrokerHandler 'malFinalize' Parameter ..... 3-149
4-1 Service Mapping Variables ..... 4-3
4-2 Area Classes ..... 4-7
4-3 Service Classes ..... 4-7
4-4 <<Service>>Stub Attribute ..... 4-21
4-5 <<Op>> Publisher Attribute ..... 4-30
4-6 <<Op>>Publisher constructor Parameter ..... 4-30
4-7 <<Op>>Publisher 'publish' Parameters ..... 4-31
4-8 Invoke <<Op>>Interaction Attribute ..... 4-32
4-9 Progress <<Op>>Interaction Attribute ..... 4-34
4-10 <<Service>>InheritanceSkeleton Attributes ..... 4-36
4-11 <<Service>>DelegationSkeleton Attributes ..... 4-39
5-1 MALTransportFactory Attributes ..... 5-1
5-2 MALTransportFactory 'registerFactoryClass' Parameters ..... 5-2
5-3 MALTransportFactory 'deregisterFactoryClass' Parameter ..... 5-2
5-4 MALTransportFactory Constructor Parameter ..... 5-3
5-5 MALTransportFactory 'newFactory'Parameter ..... 5-4
5-6 MALTransportFactory 'createTransport' Parameter ..... 5-5
5-7 MALTransport 'createEndpoint' Parameters ..... 5-6
5-8 MALTransport 'getEndpoint' Parameters ..... 5-7
5-9 MALTransport 'deleteEndpoint' Parameter ..... 5-7
5-10 MALTransport 'isSupportedQoSLevel' Parameter ..... 5-8
5-11 MALTransport 'isSupportedInteractionType' Parameter ..... 5-9
5-12 MALTransport 'createBroker' Parameters ..... 5-10
5-13 MALEndpoint 'createMessage' Parameters ..... 5-15
5-14 MALEndpoint 'sendMessage' Parameters ..... 5-16
5-15 MALEndpoint 'sendMessages' Parameters ..... 5-17
5-16 MALEndpoint 'setMessageListener' Parameter ..... 5-17
5-17 MALMessageBody 'getBodyElement' Parameters ..... 5-20
5-18 MALMessageBody 'getEncodedBodyElement' Parameter ..... 5-20
5-19 MALPublishBody 'getUpdateLists' Parameter ..... 5-23
5-20 MALPublishBody 'getUpdateList' Parameter ..... 5-23
5-21 MALPublishBody 'getUpdate' Parameter ..... 5-24
5-22 MALPublishBody 'getEncodedUpdate' Parameter ..... 5-25
5-23 MALEncodedElement Constructor Parameter ..... 5-27
5-24 MALMessageListener 'onMessage' Parameters ..... 5-29
5-25 MALMessageListener 'onMessages' Parameters ..... 5-29
5-26 MALMessageListener 'onInternalError' Parameters ..... 5-30
5-27 MALTransmitErrorException Constructor Parameters ..... 5-30
5-28 MALTransmitMultipleErrorException Constructor Parameter ..... 5-32



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# CONTENTS (continued)

Table Page

5-29 MALEncodedElementList Constructor Parameters ..... 5-33
5-30 MALEncodedBody Constructor Parameter ..... 5-33
5-31 MALTransmitErrorListener 'onTransmitError' Parameters ..... 5-34
6-1 MALAccessControlFactory 'registerFactoryClass' Parameters ..... 6-2
6-2 MALAccessControlFactory 'deregisterFactoryClass' Parameter ..... 6-2
6-3 MALAccessControlFactory 'createAccessControl' Parameter ..... 6-3
6-4 MALAccessControl 'check' Parameter ..... 6-4
6-5 MALCheckErrorException Constructor Parameters ..... 6-5
7-1 MALElementStreamFactory 'registerFactoryClass' Parameters ..... 7-2
7-2 MALElementStreamFactory 'deregisterFactoryClass' Parameters ..... 7-3
7-3 MALElementStreamFactory Creation Parameters ..... 7-3
7-4 MALElementStreamFactory 'init' Parameters ..... 7-5
7-5 MALElementStreamFactory 'createInputStream' Parameter ..... 7-5
7-6 MALElementStreamFactory 'createOutputStream' Parameter ..... 7-6
7-7 MALElementStreamFactory 'encode' Parameters ..... 7-7
7-8 MALElementInputStream 'readElement' Parameters ..... 7-8
7-9 MALElementOutputStream 'writeElement' Parameters ..... 7-9
7-10 MALEncodingContext Attributes ..... 7-11
C-1 JavaBindingNamespace Initial Values ..... C-3



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

Documents
Document Sharing Hub

Full document details: BS ISO 20210:2015

# 1 INTRODUCTION

## 1.1 PURPOSE OF THIS RECOMMENDED PRACTICE

This Recommended Practice defines a Java API for the Mission Operations (MO) Message Abstraction Layer (MAL) specified in reference [1].

## 1.2 SCOPE

The scope of this Recommended Practice is the definition of all concepts and terms that establish a Java API for consuming and providing MO services on top of the MAL. The MAL Java API is intended to maximize the portability of the MO components across various underlying MAL implementations and transport protocols.

## 1.3 APPLICABILITY

This Recommended Practice serves as a specification of a Java API providing all the concepts defined by the MAL, in particular the interaction patterns, the access control and transport interfaces.

The API supports version 1 of the MAL as specified in reference [1].

The API depends on Java 5.

## 1.4 RATIONALE

The goal of this Recommended Practice is to document how to develop and utilize MAL-based services using Java (reference [2]).

Another objective is to create a guide for promoting portability between various software packages implementing the MAL Java API and applications using the MAL Java API.

## 1.5 DOCUMENT STRUCTURE

This Recommended Practice is organized as follows:

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

b) section 2 gives an overview of the API;

c) section 3 defines the MAL API that represents the MAL service interface;

d) section 4 defines how a MAL service specification is mapped to Java;

e) section 5 defines the transport API that represents the MAL transport interface;



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

f) section 6 defines the access control API that represents the MAL access control interface;

g) section 7 defines the encoding API, an optional set of interfaces that can be used by the transport layer in order to externalize the encoding behaviour.

NOTE – Some application code examples are given in annex C.

## 1.6 DEFINITIONS

**Attribute**: Either

– a field of a Java class called ‘attribute’,

– the data type MAL::Attribute, or

– the interface Attribute defined in the MAL Java API.

## 1.7 CONVENTIONS

### 1.7.1 NOMENCLATURE

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

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

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

c) the word ‘may’ implies an optional specification;

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

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

### 1.7.2 INFORMATIVE TEXT

In the normative sections of this document (sections 3-7), informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

– Overview;

– Background;

– Rationale;

– Discussion.



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

### 1.7.3 USE OF CAPITAL LETTERS

Names of interfaces and classes of the MAL Java API are shown with the first letter of each word capitalized.

### 1.7.4 CODE TEMPLATE VARIABLES

Some code templates are given throughout this document. Those templates are parameterized with variables. A variable is used by placing its name between the enclosing characters '<<' and '>>'. For example, the variable 'method name' is used in the following code line:

```
public void <<method name>>()
```

If a variable contains a list of values, then the values are iterated with the following syntax:

```
<<variable name [i]>> … <<variable name [N]>>
```

The variable 'i' is the iteration index starting from 0.

The variable 'N' is the size of the list minus one.

If the list size is zero, then nothing is written in the code template.

If the list size is greater than zero, then the <<variable name [i]>> part is iterated N times and the <<variable name [N]>> part is finally written in the code template.

For example, the variable 'type' contains a list of types. A method 'm' taking those types as parameters is declared as follows:

```
public void m(<<type [i]>> p<<i>>, … <<type [N]>> p<<N>>)
```

If the list is empty, then the result is:

```
public void m()
```

If the list contains one element 'A' then the result is:

```
public void m(A p0)
```

The names of variables are not case-sensitive: 'field name' and 'FIELD NAME' designate the same variable.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

However, the value of the variable is case sensitive following the rules explained in table 1-1.

**Table 1-1: Variable Value Case Rules**

| Variable name case | Variable value case |
|---|---|
| All the letters in lower case. Example: `<<variable name>>` | No change |
| First letter in upper case and the other in lower case. Example: `<<Variable name>>` | No change except the first letter in upper case |
| All the letters in upper case. Example: `<<VARIABLE NAME>>` | Upper case |
| All the letters in lower case and between an initial and final character '!'. Example: `<<!variable name!>>` | Lower case |

### 1.7.5 JAVA CLASS PACKAGES

In the code templates, the package of classes is not indicated if there is no ambiguity. However, the real Java code needs either to prefix the package name to the class name or to declare the package in an 'import' clause at the beginning of the class definition.

## 1.8 REFERENCES

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

[1] *Mission Operations Message Abstraction Layer*. Recommendation for Space Data System Standards, CCSDS 521.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, October 2010.

[2] James Gosling, et al. *The Java™ Language Specification*. 3rd Ed. Boston: Addison-Wesley, 2005.

[3] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*. RFC 2396. Reston, Virginia: ISOC, August 1998.



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

[4] R. Hinden, B. Carpenter, and L. Masinter. *Format for Literal IPv6 Addresses in URL's*. RFC 2732. Reston, Virginia: ISOC, December 1999.

NOTE – Informative references are listed in annex B.



© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# 2 OVERVIEW

## 2.1 GENERAL

This Recommended Practice provides an interface specification that translates the abstract service definitions of the MAL and MO Services into a Java-specific interface that can be used by programmers. The API must be faithful to the abstract services defined in the specifications and it must, by some means, provide access to the features of the services and all of their options. It plays a useful role in application portability, but plays no role in interoperability.

The following diagram presents the set of standards documentation in support of the Mission Operations Services Concept. The MO MAL Java API belongs to the language mappings documentation.

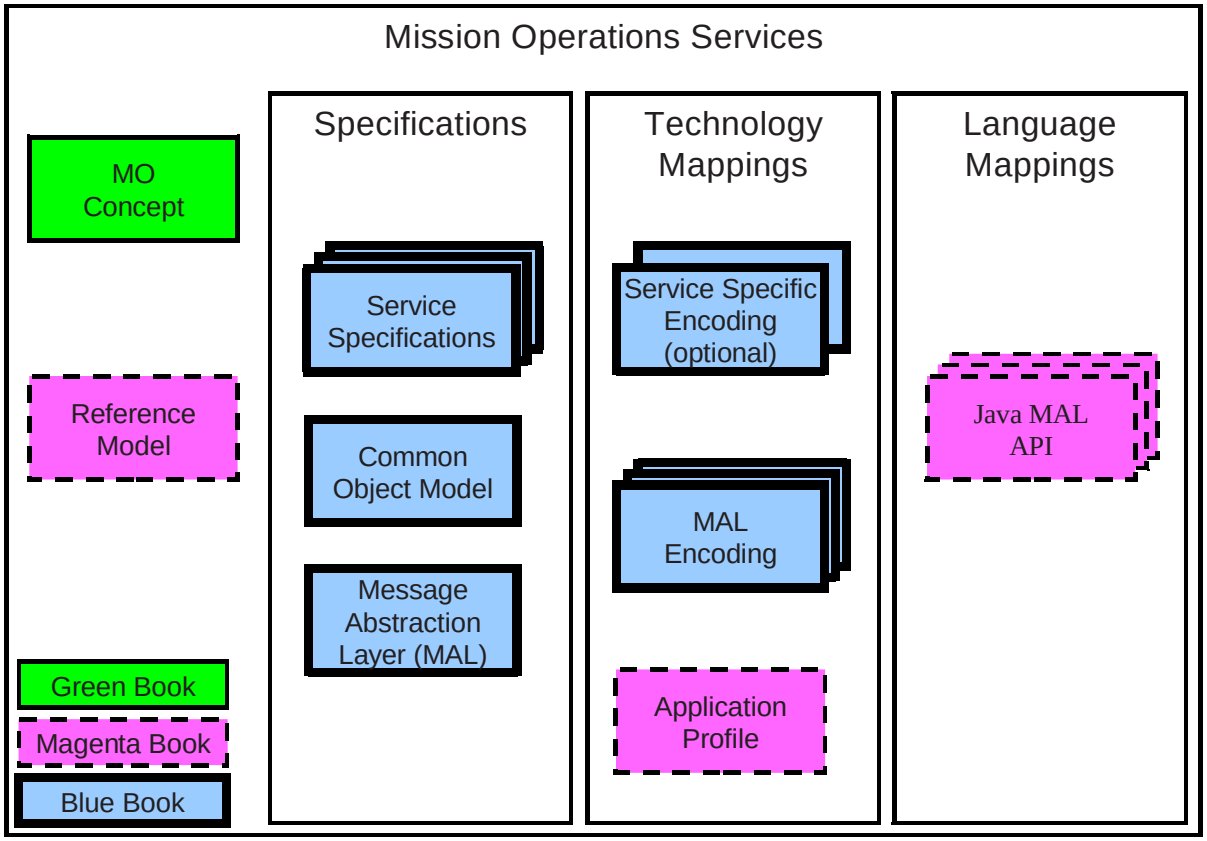


**Figure 2-1: Mission Operations Services Concept Document Set**



© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015
Documents
Document Sharing Hub

For each programming language there is only required a single mapping of the MAL to that language as the abstract services are defined in terms of the MAL and therefore their language-specific API is derived from the mapping:

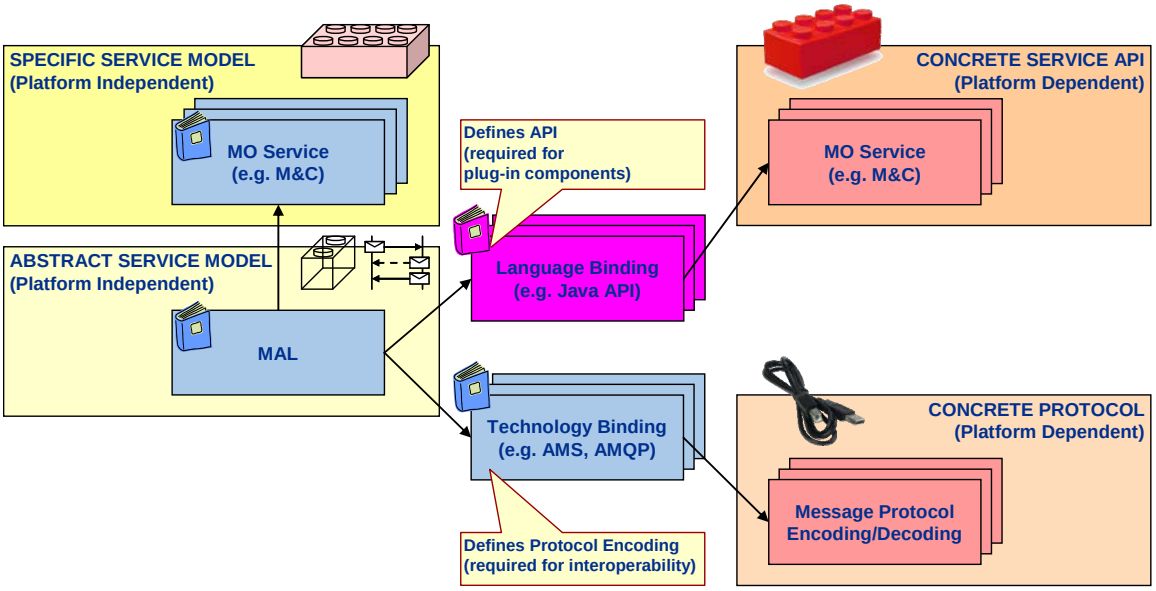


**Figure 2-2: Relationship of MO Books**

This Recommended Practice defines a mapping from the abstract notation of the MAL to an unambiguous Java API, more specifically:

a) how the specific MAL abstract services are mapped to Java;

b) how any service errors or issues shall be communicated to higher layers;

c) the physical representation of the abstract MAL messages at the interface necessary to constitute the operation templates;

d) the mapping of the message structure rules for Java;

e) the physical representation of the abstract MAL data types in Java.

It does not specify:

a) individual application services, implementations or products;

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

c) the methods or technologies required to acquire data;

d) the management activities required to schedule a service;

e) the representation of any service specific Protocol Data Units (PDUs) or operations (this is derived from the Java transform defined in this document).



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

## 2.2 MO SERVICE FRAMEWORK JAVA MAPPING

The CCSDS Spacecraft Monitoring & Control (SM&C) working group has developed a concept for a Mission Operations (MO) Service Framework, which follows the principles of Service Oriented Architectures. It defines two important aspects, the first is a protocol for interaction between two separate entities, and the second is a framework of common services providing functionality common to most uses of the service framework.

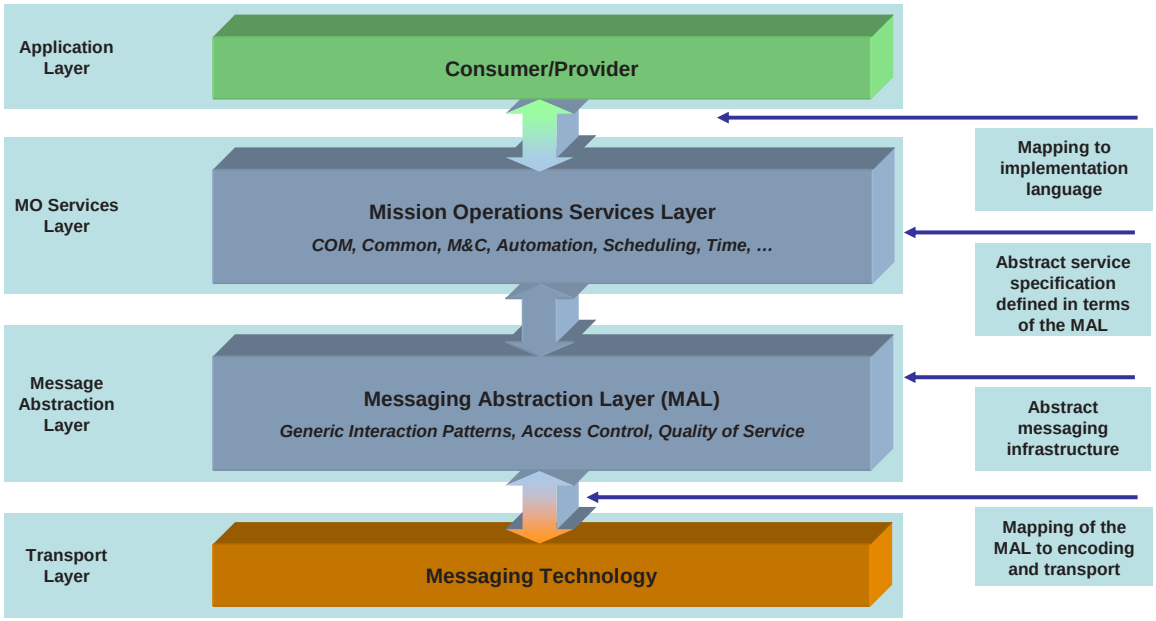


**Figure 2-3: Overview of the Mission Operations Service Framework**

This Recommended Practice defines the following Java mapping of the MO Service Framework:

a) the application layer is mapped to Java application modules consuming and providing MO services;

b) the MO services layer is mapped to service stubs and skeletons enabling the application to consume and provide MO services on top of the MAL;

c) the MAL Interaction Patterns and Quality of Service features are mapped to a MAL module;

d) the MAL access control feature is mapped to an access control module used by the MAL module;

e) the transport layer is mapped to a transport module in charge of using a messaging technology;

f) the transport module can manage the MAL message encoding internally or delegate this function to an encoding module.


Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

The layer diagram displayed in figure 2-4 presents the Java modules resulting from the MO Service Framework mapping. It also gives the name of the programming interfaces used and implemented by the modules.

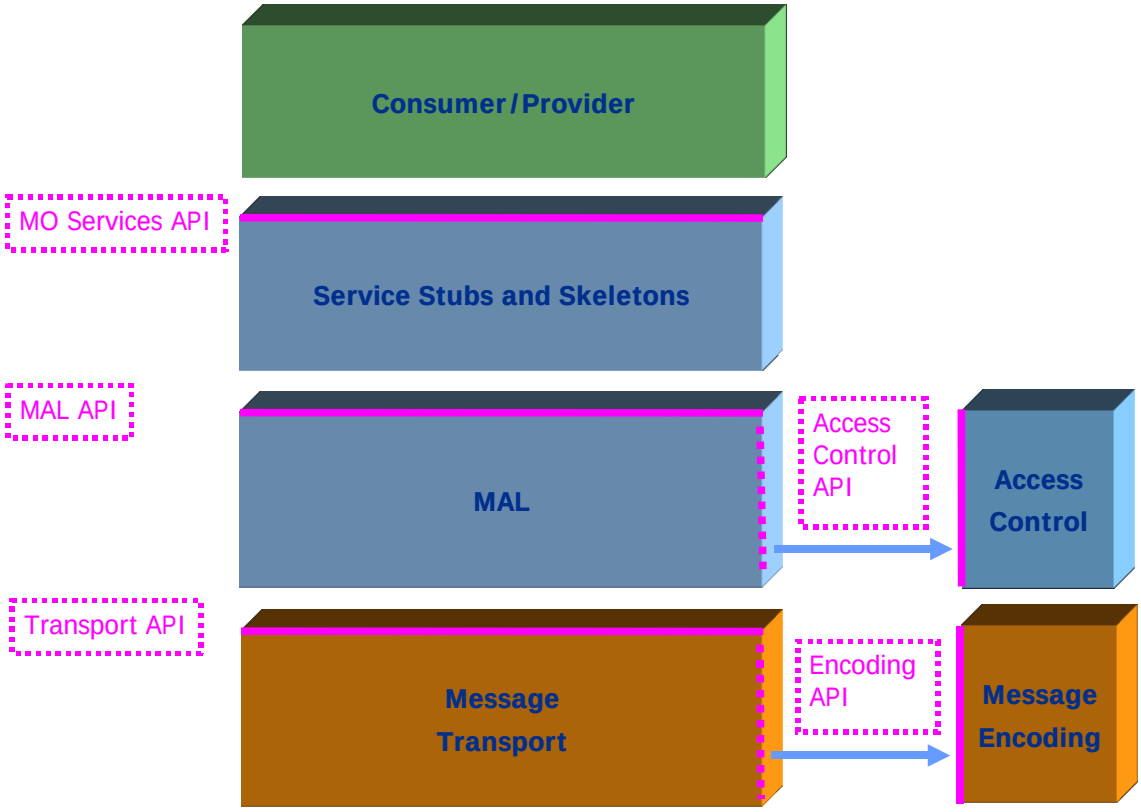


**Figure 2-4: MO Framework Java Mapping**

The programming interfaces are:

a) MAL API;

b) MO Services API generated from the Service Mapping;

c) Transport API;

d) Access Control API;

e) Encoding API.

The MO Services API is derived from the services specifications and the Java Service Mapping rules.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

## 2.3 MAPPING FROM MAL DOCUMENT

This Recommended Practice defines a mapping from the abstract notation of the MAL to an unambiguous Java API. The structure of the MAL document (reference [1]) is mapped to this document as follows:

a) the sections 'Transaction Handling', 'State Transitions', 'Message Composition', 'MAL Service Interface' are mapped to the interaction pattern handling methods of the service consumer, provider and broker defined in the section 'MAL API';

b) the sections 'MAL Data Types' and 'MAL Errors' are mapped to the section 'MAL API' and 'Service Mapping';

   1) the Java representation of the MAL fundamental and attribute data types is defined in the section 'MAL API';

   2) the Java representation of the MAL data structures is generated according to the section 'Service Mapping' except for the structures that need a specific mapping which is defined in the section 'MAL API';

   3) the Java representation of the MAL errors is generated according to the 'Service Mapping';

c) the section 'Service Specification XML' is mapped to the section 'Service Mapping';

d) the section 'Transport Interface' is mapped to the section 'Transport API';

e) the section 'Access Control Interface' is mapped to the section 'Access Control API'.

The Mission Operations Reference Model (reference [B2]) describes an encoding component that is responsible for the conversion from the abstract message format of the MAL into the on-the-wire representation used by the transport layer. The encoding component is mapped to this document as follows:

a) the encoding of the message header is internally handled by the transport layer;

b) the encoding of the message body is handled either internally by the transport layer or externally by an encoding module separated from the transport layer by the encoding API.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

# 3 MAL API

## 3.1 GENERAL

**3.1.1** The MAL API shall define the interfaces and classes required by the MAL service interface defined in reference [1].

**3.1.2** The MAL API shall comprise the following levels:

a) the generic API used by every MAL clients;

b) the data structures and related interfaces;

c) the service consumer API;

d) the service provider API;

e) the broker API.

**3.1.3** The main interfaces provided by the API shall be those listed in table 3-1.

**Table 3-1: API Main Interfaces**

| API level | Interface name | Description |
|---|---|---|
| Generic | MALContextFactory | MALContext factory |
| | MALContext | Context enabling a client to use the communication functions provided by the MAL layer |
| Consumer | MALConsumerManager | MALConsumer factory and activator |
| | MALConsumer | Communication context enabling a client to initiate interaction patterns |
| Provider | MALProviderManager | MALProvider factory and activator. |
| | MALProvider | Execution context of a service provider handling all the interaction patterns and initiating the Publish/Subscribe interaction pattern as a publisher |
| Broker | MALBrokerManager | MALBroker factory and activator |
| | MALBroker | Execution context of a shared broker |

NOTE – Figure 3-1 gives an overview of the relationships between the main interfaces.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

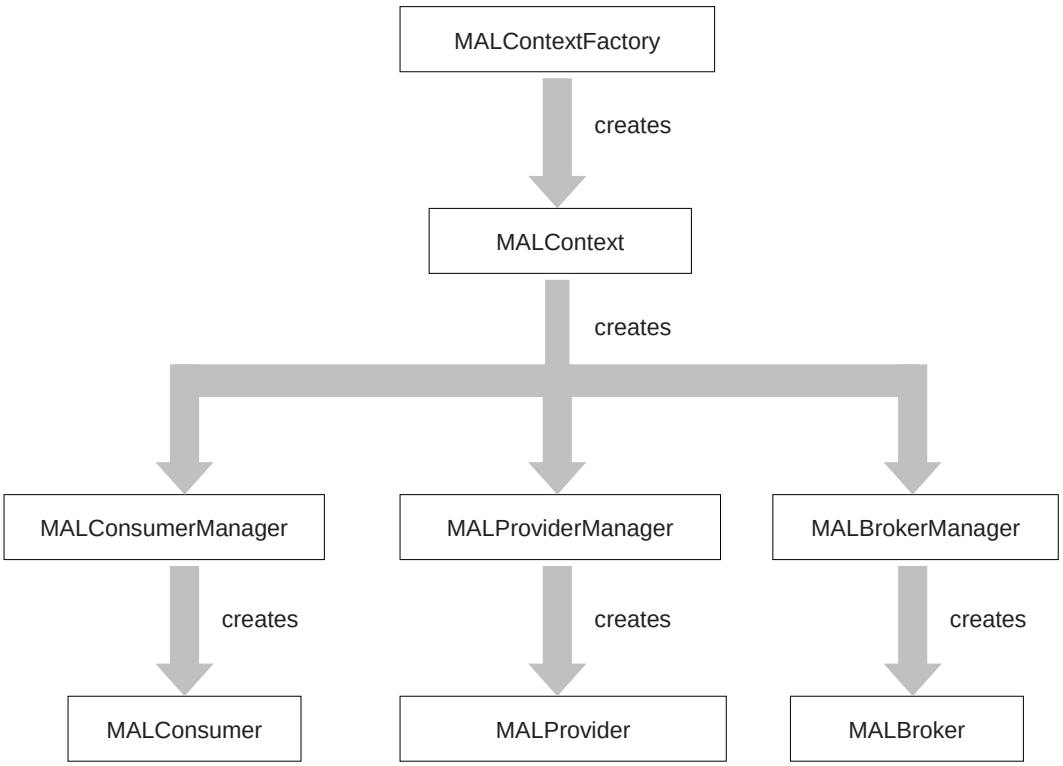


**Figure 3-1: Relationships between the API Main Interfaces**

**3.1.4** The packages provided by the API shall be:

a) the MAL package: org.ccsds.moims.mo.mal;

b) the data structures package: org.ccsds.moims.mo.mal.structures;

c) the consumer package: org.ccsds.moims.mo.mal.consumer;

d) the provider package: org.ccsds.moims.mo.mal.provider;

e) the broker package: org.ccsds.moims.mo.mal.broker.

**3.1.5** The MAL package name shall be added to the SANA registry JavaBindingNamespace and shall refer to the MAL Java API document ‘CCSDS 523.1-M-1’.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

## 3.2 MAL PACKAGE

### 3.2.1 OVERVIEW

This subsection defines the content of the generic MAL package:

`org.ccsds.moims.mo.mal`

### 3.2.2 MALCONTEXTFACTORY

#### 3.2.2.1 Definition

**3.2.2.1.1** A MALContextFactory class shall be defined in order to enable a MAL client to instantiate and configure a MALContext.

**3.2.2.1.2** The MALContextFactory class shall provide static repositories:

a) the MALContext factory class repository shall map the MALContextFactory implementation classes to their class names;

b) the MALArea repository shall map the area names and numbers to the MALArea descriptions;

c) the error repository shall map the error codes to the error names.

**3.2.2.1.3** The MALContextFactory class shall provide a MALElementFactoryRegistry.

#### 3.2.2.2 Class Initialization

The MALContextFactory class initialization shall:

a) create the static repositories;

b) instantiate the MALElementFactoryRegistry;

c) call the static method 'init' provided by the MALHelper by passing the MALElementFactoryRegistry.

#### 3.2.2.3 Factory Class Registration

**3.2.2.3.1** The MALContextFactory class shall provide a static method 'registerFactoryClass' in order to register the class of a specific MALContextFactory.

NOTE – The method 'registerFactoryClass' may be useful in environments where several class loaders are involved.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.2.3.2** The signature of the method 'registerFactoryClass' shall be:

```
public static void registerFactoryClass(Class factoryClass)
  throws java.lang.IllegalArgumentException
```

**3.2.2.3.3** The parameter of the method 'registerFactoryClass' shall be assigned as described in table 3-2.

**Table 3-2: MALContextFactory 'registerFactoryClass' Parameter**

| Parameter | Description |
|---|---|
| factoryClass | Class extending MALContextFactory |

**3.2.2.3.4** The method 'registerFactoryClass' shall store the class in the factory class repository.

**3.2.2.3.5** The method 'registerFactoryClass' shall throw an IllegalArgumentException if the parameter 'factoryClass' is NULL or does not extend MALContextFactory.

### 3.2.2.4 Factory Class Deregistration

**3.2.2.4.1** The MALContextFactory class shall provide a static method 'deregisterFactoryClass' in order to deregister the class of a specific MALContextFactory.

**3.2.2.4.2** The signature of the method 'deregisterFactoryClass' shall be:

```
public static void deregisterFactoryClass(Class factoryClass)
  throws java.lang.IllegalArgumentException
```

**3.2.2.4.3** The parameter of the method 'deregisterFactoryClass' shall be assigned as described in table 3-3.

**Table 3-3: MALContextFactory 'deregisterFactoryClass' Parameter**

| Parameter | Description |
|---|---|
| factoryClass | Class extending MALContextFactory |

**3.2.2.4.4** The method 'deregisterFactoryClass' shall remove the class from the factory class repository.

**3.2.2.4.5** The method 'deregisterFactoryClass' shall throw an IllegalArgumentException if the parameter 'factoryClass' is NULL or does not extend MALContextFactory.



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.2.4.6** The method 'deregisterFactoryClass' shall do nothing if the class is not found in the factory class repository.

#### 3.2.2.5 MALContextFactory Creation

**3.2.2.5.1** A static method 'newFactory' shall be provided in order to return a MALContextFactory instance.

**3.2.2.5.2** The signature of the method 'newFactory' shall be:

```
public static MALContextFactory newFactory() throws MALException
```

**3.2.2.5.3** The method 'newFactory' shall use the system property 'org.ccsds.moims.mo.mal.factory.class' to determine the name of the MALContextFactory implementation class to load and instantiate.

**3.2.2.5.4** The method 'newFactory' shall lookup the MALContextFactory implementation class from the factory class repository.

**3.2.2.5.5** If the MALContextFactory implementation class is not found in the factory class repository, then it shall be loaded from the current class loader.

**3.2.2.5.6** If the MALContextFactory implementation class is not found in the current class loader, then a MALException shall be raised with the ClassNotFoundException as the cause.

**3.2.2.5.7** The method 'newFactory' shall not return the value NULL.

**3.2.2.5.8** If no MALContextFactory can be returned, then a MALException shall be raised.

#### 3.2.2.6 MALContext Instantiation

**3.2.2.6.1** The MALContextFactory class shall provide an abstract public method 'createMALContext' in order to instantiate a MALContext.

**3.2.2.6.2** The signature of the method 'createMALContext' shall be:

```
public abstract MALContext createMALContext(java.util.Map properties)
  throws MALException
```

**3.2.2.6.3** The parameter of the method 'createMALContext' shall be assigned as described in table 3-4.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**Table 3-4: MALContextFactory 'createMALContext' Parameter**

| Parameter | Description |
|---|---|
| properties | Properties required by the specific MALContext implementation |

**3.2.2.6.4** The parameter 'properties' may be NULL.

**3.2.2.6.5** The method 'createMALContext' shall not return the value NULL.

**3.2.2.6.6** If no MALContext can be returned, then a MALException shall be raised.

**3.2.2.6.7** The method 'createMALContext' shall be implemented by the specific factory class.

### 3.2.2.7 MALArea Registration

**3.2.2.7.1** The MALContextFactory class shall provide a static method 'registerArea' in order to register MALAreas.

**3.2.2.7.2** The signature of the method 'registerArea' shall be:

```
public static void registerArea(MALArea area)
  throws java.lang.IllegalArgumentException, MALException
```

**3.2.2.7.3** The parameter of the method 'registerArea' shall be assigned as described in table 3-5.

**Table 3-5: MALContextFactory 'registerArea' Parameter**

| Parameter | Description |
|---|---|
| area | MALArea to register |

**3.2.2.7.4** If the parameter 'area' is NULL, then an IllegalArgumentException shall be raised.

**3.2.2.7.5** If a MALArea having the same name and the same version has already been registered and if the registered MALArea is not the same object as the MALArea to register, then a MALException shall be raised.



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

#### 3.2.2.8 MALArea Query

**3.2.2.8.1** The MALContextFactory class shall provide two static methods 'lookupArea' in order to look up a MALArea from:

a) the name and the version of the area;

b) the number and the version of the area.

**3.2.2.8.2** The signatures of the method 'lookupArea' shall be:

```
public static MALArea lookupArea(Identifier areaName, UOctet areaVersion)
  throws java.lang.IllegalArgumentException

public static MALArea lookupArea(UShort areaNumber, UOctet areaVersion)
  throws java.lang.IllegalArgumentException
```

**3.2.2.8.3** The parameters of the method 'lookupArea' shall be assigned as described in table 3-6.

**Table 3-6: MALContextFactory 'lookupArea' Parameters**

| Parameter | Description |
|---|---|
| areaName | Name of the area to look up |
| areaNumber | Number of the area to look up |
| areaVersion | Version of the area to look up |

**3.2.2.8.4** If any of the parameters 'areaName', 'areaNumber', or 'areaVersion' is NULL, then an IllegalArgumentException shall be raised.

**3.2.2.8.5** The method 'lookupArea' shall return NULL if the MALArea is not found.

#### 3.2.2.9 Error Registration

**3.2.2.9.1** The MALContextFactory class shall provide a static method 'registerError' in order to register errors.

**3.2.2.9.2** The signature of the method 'registerError' shall be:

```
public static void registerError(UInteger errorNumber,
  Identifier errorName)
  throws java.lang.IllegalArgumentException, MALException
```

**3.2.2.9.3** The parameters of the method 'registerError' shall be assigned as described in table 3-7.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**Table 3-7: MALContextFactory 'registerError' Parameters**

| Parameter | Description |
|---|---|
| errorNumber | Error code |
| errorName | Name of the error |

**3.2.2.9.4** If either of the parameters 'errorNumber' or 'errorName' is NULL, then an IllegalArgumentException shall be raised.

**3.2.2.9.5** If an error having the same code is already registered, then a MALException shall be raised.

#### 3.2.2.10 Error Query

**3.2.2.10.1** The MALContextFactory class shall provide a static method 'lookupError' in order to look up an error name from an error code.

**3.2.2.10.2** The signature of the method 'lookupError' shall be:

```
public static Identifier lookupError(UInteger errorNumber)
  throws java.lang.IllegalArgumentException
```

**3.2.2.10.3** The parameter of the method 'lookupError' shall be assigned as described in table 3-8.

**Table 3-8: MALContextFactory 'lookupError' Parameter**

| Parameter | Description |
|---|---|
| errorNumber | Code of the error to look up |

**3.2.2.10.4** If the parameter 'errorNumber' is NULL, then an IllegalArgumentException shall be raised.

**3.2.2.10.5** The method 'lookupError' shall return NULL if the error is not found.

#### 3.2.2.11 MALElementFactoryRegistry Getter

**3.2.2.11.1** The MALContextFactory class shall provide a static method 'getElementFactoryRegistry' in order to get the MALElementFactoryRegistry.

**3.2.2.11.2** The signature of the method 'getElementFactoryRegistry' shall be:

```
public static MALElementFactoryRegistry getElementFactoryRegistry()
```



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.2.11.3** The method 'getElementFactoryRegistry' shall return the MALElementFactoryRegistry that has been instantiated during the MALContextFactory class initialization.

### 3.2.3 MALCONTEXT

#### 3.2.3.1 Definition

**3.2.3.1.1** A MALContext interface shall be defined in order to enable a client to consume and/or provide services.

**3.2.3.1.2** A MALContext shall be a factory of:

a) MALConsumerManager;

b) MALProviderManager;

c) MALBrokerManager.

#### 3.2.3.2 MALContext Creation

A MALContext shall be created by calling the method 'createMALContext' provided by the MALContextFactory.

NOTE – Because of the infrastructure and communication setup done when a MALContext is created, a MALContext is a heavyweight object. Most clients will initiate and handle all the interactions with a single MALContext. However, some applications may use several MALContext objects.

#### 3.2.3.3 MALConsumerManager Creation

**3.2.3.3.1** A method 'createConsumerManager' shall be defined in order to enable to create and activate multiple MALConsumerManagers.

**3.2.3.3.2** The signature of the method 'createConsumerManager' shall be:

```
public MALConsumerManager createConsumerManager() throws MALException
```

**3.2.3.3.3** The method 'createConsumerManager' shall not return the value NULL.

**3.2.3.3.4** If no MALConsumerManager can be returned, then a MALException shall be raised.

**3.2.3.3.5** If the MALContext is closed, then a MALException shall be raised.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

#### 3.2.3.4 MALProviderManager Creation

**3.2.3.4.1** A method 'createProviderManager' shall be defined in order to enable creation and activation of multiple MALProviderManagers.

**3.2.3.4.2** The signature of the method 'createProviderManager' shall be:

```
public MALProviderManager createProviderManager() throws MALException
```

**3.2.3.4.3** The method 'createProviderManager' shall not return the value NULL.

**3.2.3.4.4** If no MALProviderManager can be returned, then a MALException shall be raised.

**3.2.3.4.5** If the MALContext is closed, then a MALException shall be raised.

#### 3.2.3.5 MALBrokerManager Creation

**3.2.3.5.1** A method 'createBrokerManager' shall be defined in order to enable creation and activation of multiple MALBrokerManagers.

**3.2.3.5.2** The signature of the method 'createBrokerManager' shall be:

```
public MALBrokerManager createBrokerManager() throws MALException
```

**3.2.3.5.3** The method 'createBrokerManager' shall not return the value NULL.

**3.2.3.5.4** If no MALBrokerManager can be returned, then a MALException shall be raised.

**3.2.3.5.5** If the MALContext is closed, then a MALException shall be raised.

#### 3.2.3.6 Get a Transport

**3.2.3.6.1** Two methods 'getTransport' shall be defined in order to get the reference of a MALTransport:

a) the first method 'getTransport' shall take as a parameter the name of the protocol to be supported by the MALTransport;

b) the second method 'getTransport' shall take as a parameter a URI which protocol is to be supported by the MALTransport.

**3.2.3.6.2** The signatures of the method 'getTransport' shall be:

```
public MALTransport getTransport(String protocol)
  throws java.lang.IllegalArgumentException, MALException

public MALTransport getTransport(URI uri)
  throws java.lang.IllegalArgumentException, MALException
```



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.3.6.3** The parameters of the method 'getTransport' shall be assigned as described in table 3-9.

**Table 3-9: MALContext 'getTransport' Parameters**

| Parameter | Description |
|---|---|
| protocol | Name of the protocol to be supported |
| uri | URI which protocol is to be supported |

**3.2.3.6.4** If either of the parameters 'protocol' or 'uri' is NULL, then an IllegalArgumentException shall be raised.

**3.2.3.6.5** The method 'getTransport' shall return a unique MALTransport instance for a given protocol.

**3.2.3.6.6** If there is still no instance of the MALTransport for the specified protocol, then such an instance shall be created.

**3.2.3.6.7** If no MALTransport can be returned, then a MALException shall be raised.

**3.2.3.6.8** If the MALContext is closed, then a MALException shall be raised.

### 3.2.3.7 Get the Access Control

**3.2.3.7.1** A method 'getAccessControl' shall be defined in order to get the reference of the MALAccessControl used by the MALContext.

**3.2.3.7.2** The signature of the method 'getAccessControl' shall be:

```
public MALAccessControl getAccessControl() throws MALException
```

**3.2.3.7.3** If no MALAccessControl can be returned, then a MALException shall be raised.

### 3.2.3.8 Close the MALContext

**3.2.3.8.1** A method 'close' shall be defined in order to synchronously close all the MALConsumerManager, MALProviderManager, and MALBrokerManager instances that have been created by this MALContext.

NOTE – In order to release the resources allocated by a MALContext, clients should close it when it is no longer used. The MALContext should be explicitly closed because it may be referenced by internal threads, making the Java Virtual Machine (JVM) garbage collector unable to release the allocated resources.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.3.8.2** The signature of the method 'close' shall be:

```
public void close() throws MALException
```

**3.2.3.8.3** If an internal error occurs, then a MALException shall be raised.

### 3.2.4 MALSERVICE

#### 3.2.4.1 Definition

**3.2.4.1.1** A MALService class shall be defined in order to represent the specification of a service (reference [1]).

**3.2.4.1.2** The MALService class shall define the attributes specified in table 3-10.

**Table 3-10: MALService Attributes**

| Attribute | Type |
|---|---|
| number | UShort |
| name | Identifier |
| version | UOctet |
| area | MALArea |
| elementFactoryRegistry | MALElementFactoryRegistry |

#### 3.2.4.2 Null Service Number

The MALService class shall define the following constant:

```
public static final UShort NULL_SERVICE_NUMBER = new UShort(0);
```

#### 3.2.4.3 Creation

**3.2.4.3.1** The MALService constructor signature shall be:

```
public MALService(
      UShort number,
      Identifier name
) throws java.lang.IllegalArgumentException
```

**3.2.4.3.2** The MALService constructor parameters shall be assigned as described in table 3-11.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**Table 3-11: MALService Constructor Parameters**

| Parameter | Description |
|---|---|
| number | Number of the service |
| name | Name of the service |

**3.2.4.3.3** If either of the parameters 'number' or 'name' is NULL, then an IllegalArgumentException shall be raised.

**3.2.4.3.4** The attribute 'number' shall be assigned with the value of the parameter 'number'.

**3.2.4.3.5** The attribute 'name' shall be assigned with the value of the parameter 'name'.

**3.2.4.3.6** The attribute 'version' shall be assigned with the value of the parameter 'version'.

### 3.2.4.4 Getters

**3.2.4.4.1** The MALService class shall provide the following getters:

a) Name getter:

```
public Identifier getName()
```

b) Area getter:

```
public MALArea getArea()
```

c) Number getter:

```
public UShort getNumber()
```

d) Element factory registry getter:

```
public MALElementFactoryRegistry getElementFactoryRegistry()
```

e) Operation getters:

1) Getter from the operation number:

```
public MALOperation getOperationByNumber(UShort opNumber)
```

2) Getter from the operation name:

```
public MALOperation getOperationByName(Identifier opName)
```

3) Getter from the capability set:

```
public MALOperation[] getOperationsByCapabilitySet(
  UShort capabilitySet)
```



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

4) Send IP operation getter:

```
public MALSendOperation[] getSendOperations()
```

5) Submit IP operation getter:

```
public MALSubmitOperation[] getSubmitOperations()
```

6) Request IP operation getter:

```
public MALRequestOperation[] getRequestOperations()
```

7) Invoke IP operation getter:

```
public MALInvokeOperation[] getInvokeOperations()
```

8) Progress IP operation getter:

```
public MALProgressOperation[] getProgressOperations()
```

9) Publish-Subscribe IP operation getter:

```
public MALPubSubOperation[] getPubSubOperations()
```

**3.2.4.4.2** The methods ‘getOperationByNumber’ and ‘getOperationByName’ shall return NULL if the operation is not found.

**3.2.4.4.3** The methods ‘getOperationsByCapabilitySet’ and ‘get<<Ip>>Operations’ shall return an empty array if no result is found.

### 3.2.4.5 Set the Area

**3.2.4.5.1** A method ‘setArea’ shall be defined in order to set the value of the attribute ‘area’.

**3.2.4.5.2** The method ‘setArea’ shall be called only by the MALArea method ‘addService’.

**3.2.4.5.3** The method ‘setArea’ control access level shall be ‘package’.

**3.2.4.5.4** The signature of the method ‘setArea’ shall be:

```
void setArea(MALArea area) throws java.lang.IllegalArgumentException
```

**3.2.4.5.5** The parameter of the method ‘setArea’ shall be assigned as described in table 3-12.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**Table 3-12: MALService 'setArea' Parameter**

| Parameter | Description |
|---|---|
| area | MALArea that owns this MALService |

**3.2.4.5.6** If the parameter 'area' is NULL, then an IllegalArgumentException shall be raised.

**3.2.4.5.7** The attribute 'area' shall be assigned with the value of the parameter 'area'.

#### 3.2.4.6 Add an Operation

**3.2.4.6.1** A method 'addOperation' shall be defined in order to enable population of the list of operations provided by a service.

**3.2.4.6.2** The signature of the method 'addOperation' shall be:

```
public void addOperation(MALOperation operation)
  throws java.lang.IllegalArgumentException
```

**3.2.4.6.3** The parameter of the method 'addOperation' shall be assigned as described in table 3-13.

**Table 3-13: MALService 'addOperation' Parameter**

| Parameter | Description |
|---|---|
| operation | MALOperation to be added into this MALService |

**3.2.4.6.4** If the parameter 'operation' is NULL, then an IllegalArgumentException shall be raised.

**3.2.4.6.5** The method 'setService' provided by the MALOperation shall be called with the reference of this MALService as a parameter.

### 3.2.5 MALOPERATION

#### 3.2.5.1 Definition

**3.2.5.1.1** A MALOperation class shall be defined in order to represent the specification of an operation provided by a service (reference [1]).

**3.2.5.1.2** The MALOperation class shall be abstract.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.5.1.3** The MALOperation class shall be extended by one class defined for each interaction pattern.

**3.2.5.1.4** The MALOperation class shall define the attributes specified in table 3-14.

**Table 3-14: MALOperation Attributes**

| Attribute | Type |
|---|---|
| number | UShort |
| name | Identifier |
| replayable | Boolean |
| interactionType | InteractionType |
| capabilitySet | UShort |
| service | MALService |

### 3.2.5.2 Creation

**3.2.5.2.1** The constructor signature of the MALOperation class shall be:

```
public MALOperation(
      UShort number,
      Identifier name,
      Boolean replayable,
      InteractionType interactionType,
      UShort capabilitySet)
  throws java.lang.IllegalArgumentException
```

**3.2.5.2.2** The MALOperation constructor parameters shall be assigned as described in table 3-15.

**Table 3-15: MALOperation Constructor Parameters**

| Parameter | Description |
|---|---|
| number | Number of the operation |
| name | Name of the operation |
| replayable | Boolean that indicates whether the operation is replayable or not |
| interactionType | Interaction type of the operation |
| capabilitySet | Capability set number of the operation |


Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.5.2.3** If any of the parameters ‘number’, ‘name’, ‘replayable’, ‘interactionType’, or ‘capabilitySet’ is NULL, then an IllegalArgumentException shall be raised.

**3.2.5.2.4** The attribute ‘number’ shall be assigned with the value of the parameter ‘number’.

**3.2.5.2.5** The attribute ‘name’ shall be assigned with the value of the parameter ‘name’.

**3.2.5.2.6** The attribute ‘replayable’ shall be assigned with the value of the parameter ‘replayable’.

**3.2.5.2.7** The attribute ‘interactionType’ shall be assigned according to the parameter ‘interactionType’.

**3.2.5.2.8** The attribute ‘capabilitySet’ shall be assigned with the value of the parameter ‘capabilitySet’.

#### 3.2.5.3 Getters

The MALOperation class shall provide the following getters:

a) Name getter:

```
public Identifier getName()
```

b) Number getter:

```
public UShort getNumber()
```

c) InteractionType getter:

```
public InteractionType getInteractionType()
```

d) Replayable getter:

```
public Boolean isReplayable()
```

e) Service getter:

```
public MALService getService()
```

f) CapabilitySet getter:

```
public UShort getCapabilitySet()
```

#### 3.2.5.4 Get a MALOperationStage

**3.2.5.4.1** An abstract method ‘getOperationStage’ shall be defined in order to return a MALOperationStage from its number.



Documents
Document Sharing Hub
© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**3.2.5.4.2** The signature of the method 'getOperationStage' shall be:

```
public abstract MALOperationStage getOperationStage(UOctet stageNumber)
  throws java.lang.IllegalArgumentException
```

**3.2.5.4.3** The parameter of the method 'getOperationStage' shall be assigned as described in table 3-16.

**Table 3-16: MALOperation 'getOperationStage' Parameter**

| Parameter | Description |
|---|---|
| stageNumber | Number of the MALOperationStage |

**3.2.5.4.4** If the parameter 'stageNumber' is NULL, then an IllegalArgumentException shall be raised except if the interaction is SEND.

**3.2.5.4.5** The method 'getOperationStage' shall return NULL only if the stage does not exist for that operation IP.

**3.2.5.4.6** The method 'getOperationStage' shall be implemented by the specific MAL<<Ip>>Operation classes.

### 3.2.5.5 Set the Service

**3.2.5.5.1** A method 'setService' shall be defined in order to set the value of the attribute 'service'.

**3.2.5.5.2** The method 'setService' shall be called by the MALService constructor.

**3.2.5.5.3** The 'setService' control access level shall be 'package'.

**3.2.5.5.4** The signature of the method 'setService' shall be:

```
void setService(MALService service)
  throws java.lang.IllegalArgumentException
```

**3.2.5.5.5** The parameter of the method 'setService' shall be assigned as described in table 3-17.



Documents
Document Sharing Hub

© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**Table 3-17: MALOperation 'setService' Parameter**

| Parameter | Description |
|---|---|
| service | Service that owns this operation |

**3.2.5.5.6** If the parameter 'service' is NULL, then an IllegalArgumentException shall be raised.

**3.2.5.5.7** The attribute 'service' shall be assigned with the value of the parameter 'service'.

### 3.2.6 MAL<<IP>>OPERATION

#### 3.2.6.1 Definition

**3.2.6.1.1** A MAL<<Ip>>Operation class shall be defined for each interaction pattern.

**3.2.6.1.2** The MAL<<Ip>>Operation class names shall be:

a) MALSendOperation;

b) MALSubmitOperation;

c) MALRequestOperation;

d) MALInvokeOperation;

e) MALProgressOperation;

f) MALPubSubOperation.

#### 3.2.6.2 Interaction Stages Constants

**3.2.6.2.1** The MAL<<Ip>>Operation shall define constants giving the interaction stage values for the stages listed in table 3-18.



© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015

**Table 3-18: Interaction Stages Constants**

| IP | stage name | stage value (HEX) |
|---|---|---|
| Send | - | - |
| Submit | SUBMIT | 0x1 |
| | SUBMIT_ACK | 0x2 |
| Request | REQUEST | 0x1 |
| | REQUEST_RESPONSE | 0x2 |
| Invoke | INVOKE | 0x1 |
| | INVOKE_ACK | 0x2 |
| | INVOKE_RESPONSE | 0x3 |
| Progress | PROGRESS | 0x1 |
| | PROGRESS_ACK | 0x2 |
| | PROGRESS_UPDATE | 0x3 |
| | PROGRESS_RESPONSE | 0x4 |
| PubSub | REGISTER | 0x1 |
| | REGISTER_ACK | 0x2 |
| | PUBLISH_REGISTER | 0x3 |
| | PUBLISH_REGISTER_ACK | 0x4 |
| | PUBLISH | 0x5 |
| | NOTIFY | 0x6 |
| | DEREGISTER | 0x7 |
| | DEREGISTER_ACK | 0x8 |
| | PUBLISH_DEREGISTER | 0x9 |
| | PUBLISH_DEREGISTER_ACK | 0xA |

**3.2.6.2.2** The constants shall be declared as specified below:

```
public static final byte _<<STAGE NAME>>_STAGE = 0x<<stage value>>
public static final UOctet <<STAGE NAME>>_STAGE = new UOctet(_<<STAGE
NAME>>_STAGE)
```

### 3.2.6.3 Creation

**3.2.6.3.1** For every interaction type except PUBLISH-SUBSCRIBE, the constructor signature of the MAL<<Ip>>Operation shall be:

```
public MAL<<Ip>>Operation(
      UShort number,
      Identifier name,
      Boolean replayable,
      UShort capabilitySet,
```



© ISO 2015 – All rights reserved
Full document details: BS ISO 20210:2015