INTERNATIONAL STANDARD

ISO 17807

First edition
2013-06-01

Space data and information transfer systems — Asynchronous message service

Systèmes de transfert des informations et données spatiales — Service de messagerie asynchrone

Reference number
ISO 17807:2013(E)

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2. www.iso.org/directives

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received. www.iso.org/patents

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

ISO 17807 was prepared by the Consultative Committee for Space Data Systems (CCSDS) (as CCSDS 735.1-B-1, September 2011) 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*.

Space data and information transfer systems — Asynchronous message service

1 Scope

This International Standard defines a CCSDS Asynchronous Message Service (AMS) for mission data system communications. The service and its protocols implement an architectural concept under which the *modules* of mission systems—distinct sequential flows of application control logic, whether called processes, tasks, or threads—may be designed as if they were to operate in isolation, each one producing and consuming mission information without explicit awareness of which other modules are currently operating. Communication relationships among such modules are self-configuring; this tends to minimize complexity in the development and operations of modular data systems.

A system built on this model is a 'society' of generally autonomous interoperating modules that may fluctuate freely over time in response to changing mission objectives, module functional upgrades, and recovery from individual module failure. The purpose of AMS, then, is to reduce mission cost and risk by providing standard, reusable infrastructure for the exchange of information among data system modules in a manner that is simple to use, highly automated, flexible, robust, scalable, and efficient.

This International Standard specifies the protocol procedures and data units that accomplish automatic configuration of AMS communication relationships, dynamic reconfiguration of those relationships during operations, and the use of those relationships to accomplish the exchange of mission information among data system modules.

The scope and field of application are furthermore detailed in subclause 1.2 of the enclosed CCSDS publication.

2 Requirements

Requirements are the technical recommendations made in the following publication (reproduced on the following pages), which is adopted as an International Standard:

CCSDS 735.1-B-1, September 2011, Asynchronous message service.

For the purposes of international standardization, the modifications outlined below shall apply to the specific clauses and paragraphs of publication CCSDS 735.1-B-1.

Pages i to vi

This part is information which is relevant to the CCSDS publication only.

Page 1-8

Add the following information to the reference indicated:

[3] Document CCSDS 301.0-B-4, November 2010, is equivalent to ISO 11104:2013.

3 Revision of publication CCSDS 735.1-B-1

It has been agreed with the Consultative Committee for Space Data Systems that Subcommittee ISO/TC 20/SC 13 will be consulted in the event of any revision or amendment of publication CCSDS 735.1-B-1. To this end, NASA will act as a liaison body between CCSDS and ISO.

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

ASYNCHRONOUS MESSAGE SERVICE

RECOMMENDED STANDARD

CCSDS 735.1-B-1

BLUE BOOK
September 2011

(Blank page)

AUTHORITY

Issue:	Recommended Standard, Issue 1
Date:	September 2011
Location:	Washington, DC, USA

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

This document is published and maintained by:

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

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **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:

- o 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.
- o 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.
- o 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

As the computers used to conduct space flight mission operations both in flight and on the ground increase in capability, the software operating on those computers tends to increase in functional scope and thus take on greater operational significance. With that increase in scope comes increasing size and complexity, which may be partially mitigated by decomposition into modules whose functionality can be readily defined and tested. However, this modularity in turn entails a growing reliance on effective communication among modules. That is, mere decomposition cannot diminish the functional complexity implied by complex requirements. It can only partition that complexity into manageable portions, while the resulting web of communication relationships among modules introduces new complexity of a different order: rather than a relatively simple system of a few increasingly large and complex modules, a modern mission is increasingly likely to require a large and complex system of relatively simple modules.

Increasing complexity tends to increase the likelihood of failure. The increasing complexity of mission systems based on communication among modules therefore tends to increase the chance of such systems' failing even as the success of those systems becomes increasingly critical to the achievement on mission objectives. Measures that can minimize the chance of failure in complex systems—exhaustive regression testing and configuration management, flight rules constraining the exercise of unproven system capabilities and the introduction of improvements—increase cost if they are taken and increase risk if they are not.

These considerations have led to the present recommendation for a standard system of communication—*messaging*—among mission software modules. The objective of this proposed messaging standard is to reduce mission cost and risk by confining much of the complexity of modern mission systems to relatively static and proven reusable infrastructure.

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

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

http://www.ccsds.org/

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

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

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- 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.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 735.1-B-1	Asynchronous Message Service, Recommended Standard, Issue 1	September 2011	Original issue

CONTENTS

CONTENTS (continued)

(Blank page)

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

This document defines a CCSDS Asynchronous Message Service (AMS) for mission data system communications. The service and its protocols implement an architectural concept under which the *modules* of mission systems—distinct sequential flows of application control logic, whether called processes, tasks, or threads—may be designed as if they were to operate in isolation, each one producing and consuming mission information without explicit awareness of which other modules are currently operating. Communication relationships among such modules are self-configuring; this tends to minimize complexity in the development and operations of modular data systems.

A system built on this model is a 'society' of generally autonomous interoperating modules that may fluctuate freely over time in response to changing mission objectives, module functional upgrades, and recovery from individual module failure. The purpose of AMS, then, is to reduce mission cost and risk by providing standard, reusable infrastructure for the exchange of information among data system modules in a manner that is simple to use, highly automated, flexible, robust, scalable, and efficient.

This Recommended Standard specifies the protocol procedures and data units that accomplish automatic configuration of AMS communication relationships, dynamic reconfiguration of those relationships during operations, and the use of those relationships to accomplish the exchange of mission information among data system modules.

1.2 APPLICABILITY

This Recommended Standard specifies protocols and associated services that enable communication among modules of mission data systems, specifically:

– between modules of a ground data system;

– between modules of a flight data system;

– between modules of different ground data systems;

– between modules of the flight data systems of different spacecraft;

– between modules of flight data systems and modules of ground data systems, over interplanetary distances.

1.2.1 ORGANIZATION OF THE RECOMMENDED STANDARD

This Recommended Standard is organized as follows:

– Section 2 provides an overview of AMS, its intended use, and a description of the main interactions involved in message exchange.

– Section 3 defines the services provided by AMS along with the associated primitives and parameters.
– Section 4 defines the AMS protocol procedures.
– Section 5 defines the AMS protocol data units.
– Section 6 defines standardized user operations that may be built on AMS.
– Section 7 defines eight Conformance Classes of AMS implementation.
– Section 8 describes the AMS Management Information Base (MIB).
– Annex A discusses the currently recognized underlying transport services for AMS.
– Annex B summarizes the functional significance of various classes of AMS subject numbers and the manner in which this significance affects protocol procedures.
– Annex C defines a Service Conformance Statement Proforma.
– Annex D defines a Protocol Implementation Conformance Statement (PIC) Proforma.
– Annex E discusses security, SANA, and patent considerations related to the specification.
– Annex F provides a list of informative references.
– Annex G provides a list of acronyms and definitions.

1.3 CONVENTIONS AND DEFINITIONS

1.3.1 BIT NUMBERING CONVENTION

In this document, the following convention is used to identify each bit in an *N*-bit field. The first bit in the field to be transmitted (i.e., the most left justified when drawing a figure) is defined to be ‘Bit 0’; the following bit is defined to be ‘Bit 1’ and so on up to ‘Bit *N*–1’. When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, i.e., ‘Bit 0’.

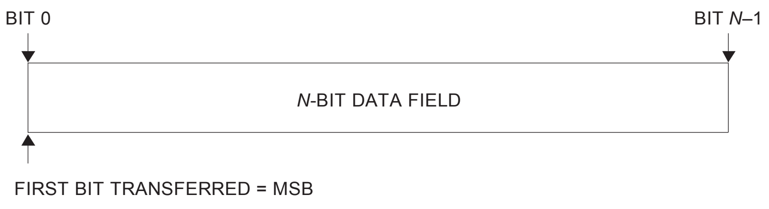


Figure 1-1: Bit Numbering Convention

In accordance with modern data communications practice, spacecraft data fields are often grouped into eight-bit 'words' which conform to the above convention. Throughout this Recommended Standard, the following nomenclature is used to describe this grouping:

8-BIT WORD = 'OCTET'

Figure 1-2: Octet Convention

By CCSDS convention, all 'spare' or 'unused' bits shall be permanently set to value 'zero'.

1.3.2 NOMENCLATURE

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

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

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

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

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

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

1.3.3 INFORMATIVE TEXT

In the normative sections of this document, sections 3 through 8 and annexes A through D, informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

– Overview;

– Background;

– Rationale;

– Discussion.

1.3.4 DEFINITIONS

1.3.4.1 Definitions from OSI Basic Reference Model

This Recommended Standard makes use of a number of terms defined in reference [1]. The use of those terms in this Recommended Standard shall be understood in a generic sense, i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

– entity;

– Protocol Data Unit (PDU);

– service;

– Service Access Point (SAP);

– Service Data Unit (SDU).

1.3.4.2 Definitions from Open Systems Interconnection (OSI) Service Definition Conventions

This Recommended Standard makes use of a number of terms defined in reference [2]. The use of those terms in this Recommended Standard shall be understood in a generic sense, i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

– Indication;

– Primitive;

– Request;

– Response.

1.3.4.3 Terms Defined in This Recommended Standard

NOTE – In view of the detailed technical rigor of these definitions, the first-time reader is encouraged to skip them and instead begin at the much more accessible section 2, below, referring back to the definitions in 1.3.4.3 as necessary.

Within the context of this document the following definitions apply:

continuum: A closed set of entities that utilize the Meta-AMS protocol (discussed below) for the purpose of configuring AMS-based communication among themselves. Each continuum is identified by a *continuum name* and corresponding non-negative continuum number. A ‘continuum ID’ is either a continuum name or a continuum number, whichever is more appropriate in context.

application: A data system implementation, typically taking the form of a set of source code text files, that relies on AMS procedures to accomplish its purposes. Each application is identified by an *application name*.

authority: An administrative entity or persona that may have responsibility for the configuration and operation of an instance of an application. Each authority is identified by an *authority name*.

venture: An instance of an application, i.e., a functioning projection of the application, for which some authority is responsible, onto a set of one or more running computers. As such it is identified by the combination of some application name and some authority name, and it is alternatively identified by some non-negative, non-zero *venture number* that corresponds to that combination of application and authority names. A 'venture ID' is either an application-name/authority-name combination or a venture number, whichever is more appropriate in context

message: An octet array of known size which, when copied from the memory of one module of a venture to that of another (*exchanged*), conveys information that can further the purposes of that venture.

application data: The array of zero or more octets embedded in a message containing the specific information that the message conveys.

role: Some part of the functionality of an application. Each role is identified by a *role name* and corresponding non-negative role number. A 'role ID' is either a role name or a role number, whichever is more appropriate in context.

module: A communicating entity that implements some part of the functionality of some AMS venture—that is, performs some application role—by, among other activities, exchanging messages with other modules. Associated with each module is the name of the role it performs within the application. (Multiple modules may perform the same role in an application, so the role name of a module need not uniquely identify the module within its venture.) In order to accomplish AMS message exchange a module generates AMS service requests and consumes AMS service indications; the module that is the origin of a given AMS service request or the destination of a given AMS service indication is termed *the operative module*.

message space: The set of all of the modules of one AMS venture that are also members of a given AMS continuum; that is, a message space is the intersection of a venture and a continuum. Each message space is uniquely identified within its continuum by the combination of the name of the application and the name of the authority that is responsible for the venture, and by the corresponding venture number.

NOTE – Unique naming of continua enables multiple message spaces that are in different continua but are identified by the same application and authority names to be concatenated via Remote AMS (discussed below) into a single venture.

unit (i.e., a unit of organization): An identified subset of the organizational hierarchy of the modules of one AMS venture, declared during venture configuration as specified by the responsible authority for that venture. Each unit is uniquely identified within the venture by *unit name* and corresponding non-negative unit number. A 'unit ID' is either a unit name or a unit number, whichever is more appropriate in context. The *root unit* of a venture is the unit that is coterminous with the venture itself; its unit name is the character string that is of length zero. A unit whose name is identical to the first N bytes (where N is greater than or equal to zero) of the name of another unit of the same venture is said to *contain* that other unit. The membership of a unit that is contained by another unit is a subset of the membership of the containing unit. Units form a strict hierarchy: no module is ever a member of two units such that neither unit is a proper subset of the other.

cell: The set of all modules that are members of one unit of a given venture and are also members of a given continuum; that is, it is the intersection of a unit and a continuum. Since each unit is a subset of a venture, each cell is necessarily a subset of the message space for that venture in that continuum. Each cell is uniquely identified within its message space by its unit's name and number. The root cell of a message space is coterminous with the message space itself. A cell contains some other cell only if its unit contains that other cell's unit. A cell may be an empty set; that is, in a given continuum there may be no modules that are members of the cell's unit. The *registered membership* of a cell is the set of all modules in the cell that are not members of any other cell aside from cells which contain that cell.

NOTES

1 For example, if cell A contains cells B and C, and cell C contains cells D and E, any modules in C that are not in either D or E are in the registered membership of cell C. Those modules are also members of cell A, but because they are in cell C—which does not contain cell A—they are not in cell A's registered membership.

2 The root cell contains every other cell in the message space, and every module in the message space is therefore a member, though not necessarily a registered member, of the root cell.

domain (of an AMS service request): The set of modules to which the request pertains. It comprises all of the modules that are members of the venture in which the operative module is itself a member, with the following exceptions:

– If the service request is one for which continuum ID is not specified, then only modules that are members of the local continuum are members of the service request's domain. Otherwise, if the service request's continuum ID parameter does not indicate 'all continua', then only modules that are members of the continuum identified by the service request's continuum ID parameter are members of the service request's domain.

– Only modules that are members of the organizational unit identified by the service request's unit ID parameter are members of the service request's domain.

– If the service request's role ID parameter does not indicate 'all roles', then only modules performing the role identified by that role ID are members of the service request's domain.

subject number or **subject** (of a message): An integer embedded in the message that indicates the general nature of the information the message conveys, in the context of the AMS venture within which the message is exchanged.

subject name: A text string that serves as the symbolic representation of some subject number. A 'subject ID' is either a subject number or the corresponding subject name, whichever is more appropriate in context.

send: To cause a message to be copied to the memory of a specified module.

publish: To cause a message on a specified subject to be sent to one or more implicitly specified modules, namely, all those that have requested copies of all messages on the specified subject.

announce: To cause a message to be sent to one or more implicitly specified modules, namely, all those modules that are located within a specified continuum (or all continua), that are members of a specified unit (possibly the root unit), and that perform a specified role in the application (possibly 'any role').

subscribe: To issue a *subscription* on a subject, i.e., a request that one copy of every message published on some specified subject by any module in the subscription's *domain* be sent to the subscribing module; the domain of a subscription is the domain of the AMS service request that established the subscription.

invitation: A statement of the manner in which messages on some specified subject may be sent to the inviting module by modules in the *domain* of the invitation; the invitation's domain is the domain of the AMS service request that established the invitation.

reply: A message that is sent back to the sender of some earlier message and that directly responds to that antecedent message in some application-specific way.

context number: A numeric value that (in some application-specific way) enables the information in a reply message to be applied in the context in which the antecedent message was sent. For this purpose, the context number, if any, included in the antecedent message is simply echoed in the reply message.

query: A message that implicitly notifies the receiver that the sender of the message has suspended message exchange altogether until it has received a reply that responds to this message.

Some of the protocol procedures defined in this specification are in part expressed in terms of timeout intervals. All AMS timeout intervals are notional rather than fixed. (However, effective operation of the AMS protocols cannot be assured unless the same fixed values for all timeout intervals are used universally within any given AMS continuum.) Similarly, the system health monitoring procedures defined in this specification are in part expressed in terms of the number of consecutive missing heartbeats that will be interpreted as indicating that the source entity for the missing heartbeats has ceased operation. This limit is again notional rather than fixed; it is identified in this specification by the label ‘N6’, and its nominal value is 3.

The timeout intervals referenced in this specification are defined as follows:

Table 1-1: AMS Timeout Intervals (in Seconds)

Interval Name	Interval ID	Nominal Value	Mandatory value
Configuration server response	N1	5.0	n/a
Registrar response	N2	5.0	n/a
Registrar heartbeat	N3	10.0	n/a
Module heartbeat	N4	n/a	2 X the value of N3
Module loss imputation	N5	n/a	N6 X the value of N4

NOTE – An example of an AMS-based communications configuration that illustrates some of the concepts defined above is provided in 2.1.3.

1.4 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommended Standard. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Standard 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] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[2] *Information Technology—Open Systems Interconnection—Basic Reference Model—Conventions for the Definition of OSI Services*. International Standard, ISO/IEC 10731:1994. Geneva: ISO, 1994.

[3] *Time Code Formats*. Recommendation for Space Data System Standards, CCSDS 301.0-B-4. Blue Book. Issue 4. Washington, D.C.: CCSDS, November 2010.

[4] *AMQP—Advanced Message Queuing Protocol—Protocol Specification*. Version 0-10. N.p.: AMQP Working Group, 2009. <http://amqp.org/>

NOTE – Informative references are contained in annex F.

2 OVERVIEW

2.1 GENERAL

2.1.1 ARCHITECTURAL CHARACTER

A data system based on AMS has the following characteristics:

– Any module may be introduced into the system at any time. That is, the order in which system modules commence operation is immaterial; a module never needs to establish an explicit *a priori* communication 'connection' or 'channel' to any other module in order to pass messages to it or receive messages from it.

– Any module may be removed from the system at any time without inhibiting the ability of any other module to continue sending and receiving messages. That is, the termination of any module, whether planned or unplanned, only causes the termination of other modules that have been specifically designed to terminate in this event.

– When a module must be upgraded to an improved version, it may be terminated and its replacement may be started at any time; there is no need to interrupt operations of the system as a whole.

– When the system as a whole must terminate, the order in which the system's modules cease operation is immaterial.

AMS-based systems are highly robust, lacking any innate single point of failure and tolerant of unplanned module termination. At the same time, communication within an AMS-based system can be rapid and efficient:

– Messages are exchanged directly between modules rather than through any central message dispatching nexus.

– Messages are automatically conveyed using the most suitable underlying transport service to which the sending and receiving modules both have access. For example, messages between two ground system modules running in different computers on a common LAN would likely be conveyed via TCP/IP, while messages between modules running on two flight processors connected to a common bus memory board might be conveyed via a shared-memory message queue.

Finally, AMS is designed to be highly scalable: partitioning message spaces into cells enables a venture to comprise hundreds or thousands of cooperating modules without significant impact on application performance.

2.1.2 MESSAGE EXCHANGE MODELS

AMS is designed to support a variety of message exchange models:

– *Send/receive*. Messages may simply be sent to designated modules in an asynchronous manner.

– *Synchronous query*. Messages may be sent in a synchronous manner, i.e., suspending further activity until a reply is received.

– *Publish/subscribe*. Messages may be published anonymously to a time-varying community of self-selected subscribers.

– *Announcement*. Messages may be 'announced' to a set of modules selected by the message source.

To this end, AMS comprises three distinct and complementary protocols:

– The *Application AMS* (*AAMS*) protocol conveys application data between AMS modules within a single continuum.

– The *Meta-AMS* (*MAMS*) protocol propagates configuration information that enables the exchange of AAMS protocol data units within a single continuum.

– The *Remote AMS* (*RAMS*) protocol conveys application data between AMS modules in different continua.

AAMS message exchange is fundamentally *asynchronous*. That is, each message is sent in a 'postal' rather than 'telephonic' manner: upon sending a message, an AMS module need not wait for arrival of any message (such as a reply to the message it sent) before continuing performance of its functions.

Although message exchange among modules is asynchronous, for some purposes it may be desirable to apply the information in a reply message (received asynchronously) to the context in which the antecedent message was published. To this end, AAMS enables a module to include a context number in any original (non-reply) message; AAMS procedures can be used to reply to any original message, and the reply to a message automatically includes an echo of the context number (if any) embedded in the original message. This number can be used to retrieve some block of contextual information, enabling the original message sender to link the information in a reply message to the application activity which caused the antecedent message to be issued, so that this activity may be continued in a pseudo-synchronous fashion. The specific mechanism used to establish this linkage is an implementation matter.

For some purposes true message synchrony may be necessary as well; that is, it may be desirable for a module to query some other module, sending a message and then suspending message exchange altogether until a reply is received. AAMS procedures additionally support this communication model when it is required.

Most message exchange in an AMS-based data system is conducted on the 'publish-subscribe' model:

– A module uses MAMS procedures to announce that it is subscribing to messages on a specified subject.

– From that time on (until the subscription is canceled), whenever any module in the message space uses AAMS procedures to publish a message on that subject, a copy of the message is automatically delivered to that subscribing module and to all others that have announced similar subscriptions.

This model can greatly simplify application development and integration. In effect, each module plugs itself into a data 'grid', much as producers and consumers of electric power—for example, a hydroelectric plant and a kitchen toaster—plug into an electric power grid. An AMS module can insert into such a data grid whatever data it produces, without having to know much about the consumer(s) of that data, and draw from the grid whatever data it requires without having to know much about the producer(s). The design of a module is largely decoupled from the designs of all other modules in the same way that the design of a toaster is largely decoupled from the design of a power plant.

For some purposes, though, it may still be necessary for a module to send a message privately and explicitly to some specific module, e.g., in reply to a published message. Again, AAMS procedures support this communication model as well when it is required.

Finally, a module may also announce a message privately to some explicitly specified set of modules (thus privately and implicitly to all modules in that set), where the specification of the set may comprise any combination of continuum identifier, unit identifier, and role identifier. The announcement model enables the source of the message to exercise some discretion in multi-point data distribution, directing the message only to the modules it believes need to receive it, while still shielding destination modules from the delivery of messages they prefer not to receive.

2.1.3 AMS COMMUNICATION CONFIGURATION EXAMPLE

In this example, AMS is being used to operate a rover on a distant planet. This application is named 'rover-ops'.

For this purpose, AMS must be running in two distinct continua: 'rover', sited on the rover's flight computer, and 'moc', sited on the rover's Mission Operations Center (MOC) computers on Earth.

NOTE – It is not possible to run the entire system in a single continuum because the Meta-AMS (MAMS) protocol relies on timely responses to messages: the lengthy signal propagation delays of interplanetary communication would prevent the successful completion of MAMS message exchanges. The Remote AMS protocol propagates the results of MAMS-based continuum configuration among continua in a delay-tolerant manner.

It is desirable to run this system both 'live' and in simulation, without modifying the software in any way, so two different authorities, named 'live' and 'sim', are identified.

Finally, it is desirable to be able to send a request for current thermal status information to all modules of the thermal subsystem on the rover and no other modules. For this purpose it is assumed that one unit is defined for each rover-ops venture will be the 'thermal' unit.

Figure 2-1 depicts one of the two continua, the **rover** continuum. Each continuum has a single configuration server (discussed later), marked 'C' in this diagram.

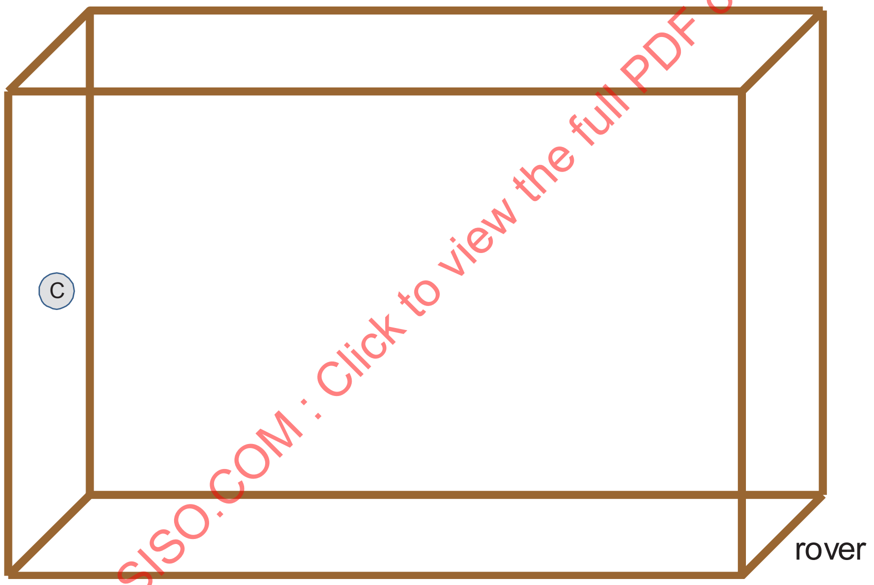


Figure 2-1: The Rover Continuum

In figure 2-2, the **moc** continuum is depicted as well.

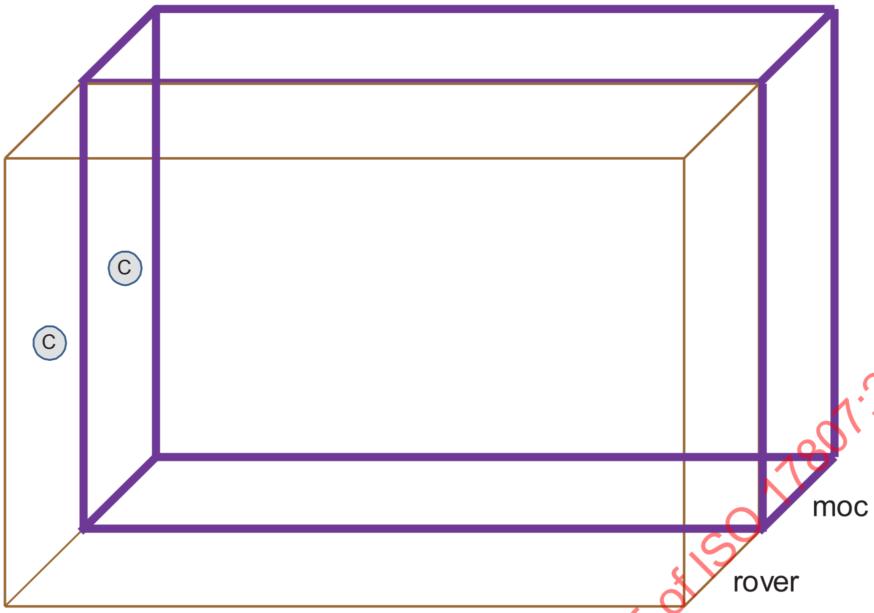


Figure 2-2: Rover and MOC Continua

Figure 2-3 shows a single venture that conducts the **live** operations of the **rover-ops** application, spanning these two continua. Every venture comprises at least one implicit unit, the **root** unit, but since the root unit is coterminous with the venture it is not shown separately here.

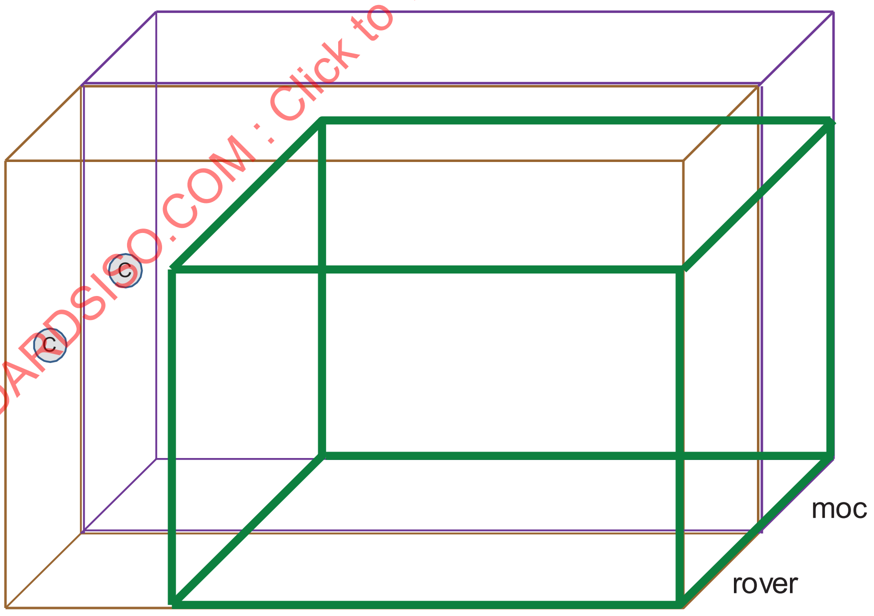


Figure 2-3: Live Rover-Ops Venture

The portion of the **live rover-ops** venture that resides in the **rover** continuum is a single message space as shown in figure 2-4.

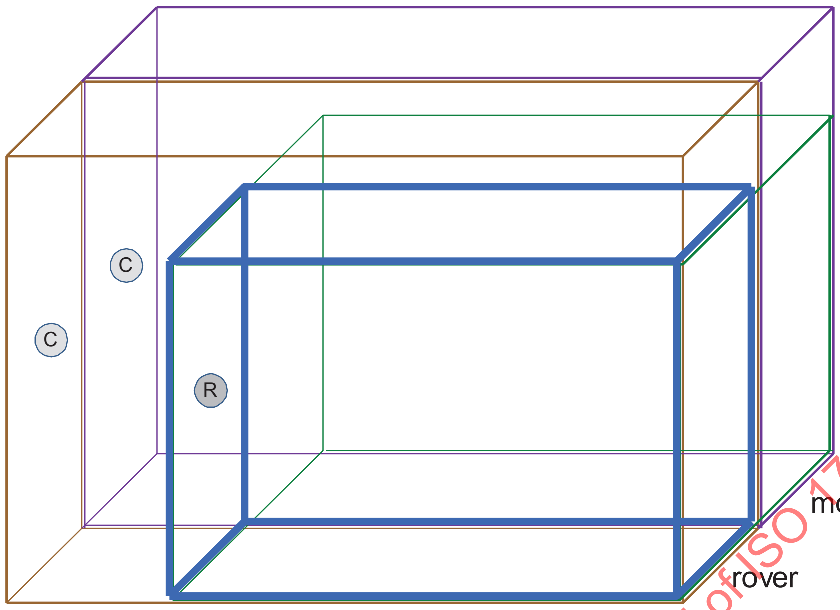


Figure 2-4: Rover Continuum Portion of Live Rover-Ops Venture

Each message space comprises at least one cell, the **root** cell, which is that portion of the **live rover-ops root** unit that resides in the **rover** continuum. Since the **root** cell is coterminous with the message space itself it is not shown separately. Every cell of every message space, including the **root** cell, has a single registrar (discussed later), marked ‘R’ in this diagram.

In figure 2-5, the message space for the **MOC** portion of **live rover-ops** is added.

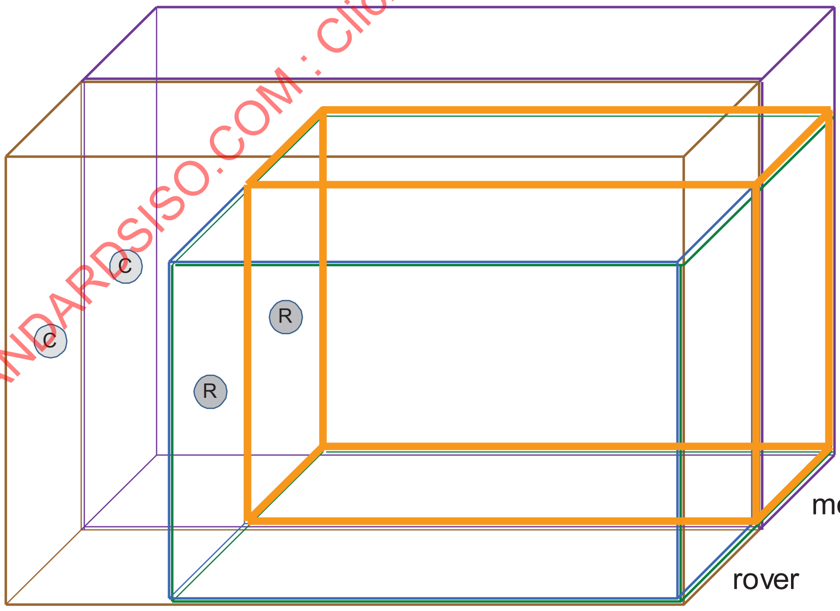


Figure 2-5: Live Rover-Ops Venture with MOC Message Space

Figure 2-6 shows a subset of the **live rover-ops** venture, the **thermal** unit, which spans the **rover** and **moc** continua.

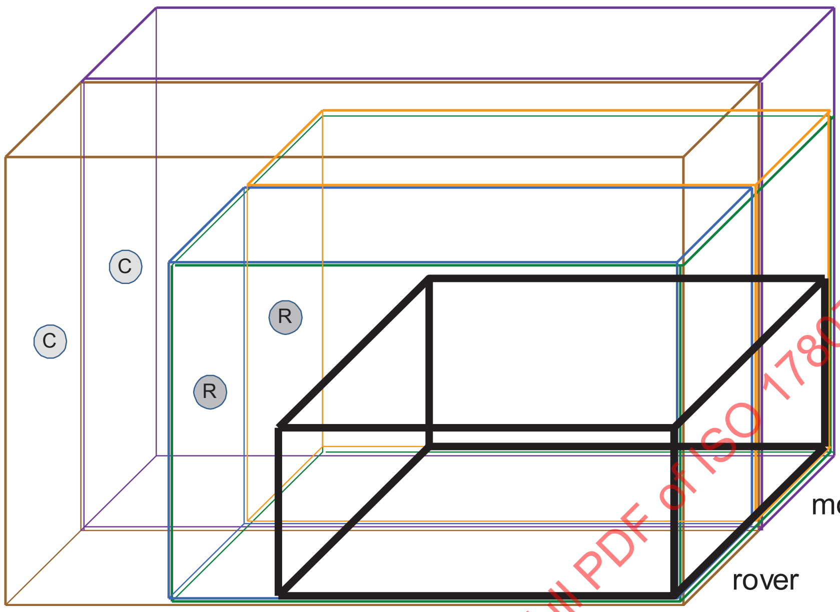


Figure 2-6: Thermal Unit Subset of Live Rover-Ops Venture

The portion of the **live rover-ops thermal** unit that resides in the **rover** continuum is a single cell as shown in figure 2-7. Modules numbered 19 and 6 are registered with the registrar of this cell.

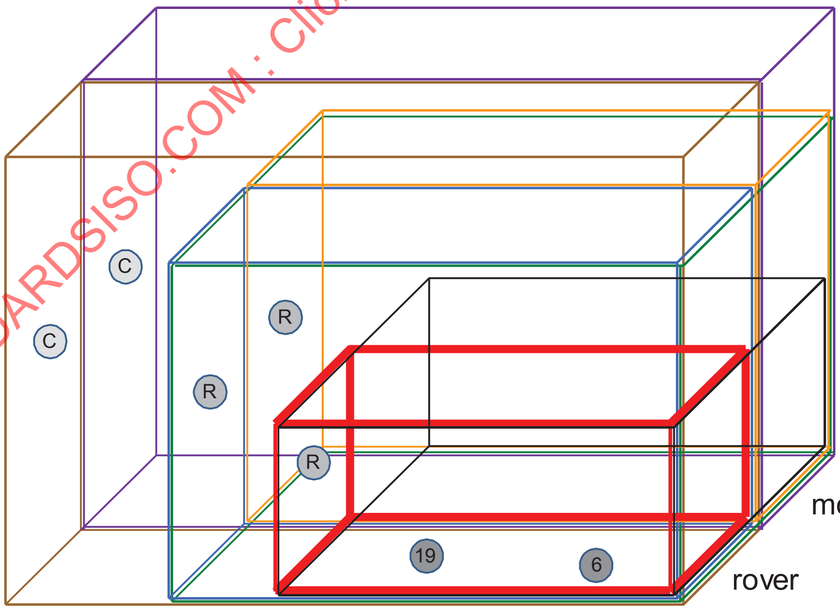


Figure 2-7: Rover Continuum Portion of Live Rover-Ops Thermal Unit

In the figure 2-8, the **live rover-ops thermal** cell residing in the **moc** continuum has been added as well.

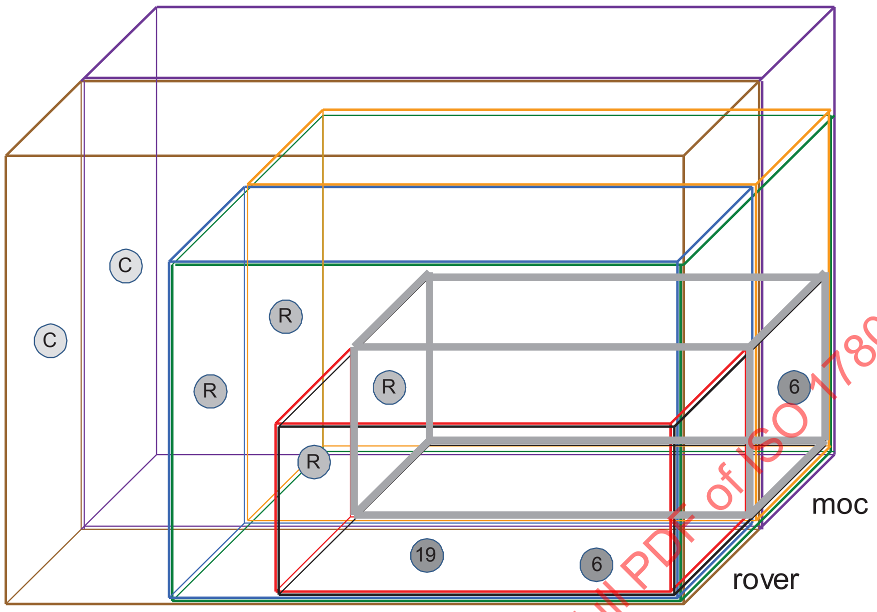


Figure 2-8: Rover and MOC Continua Portions of Live Rover-Ops Thermal Unit

It should be noted that this cell also has a registered module 6, but there is no ambiguity: module numbers are meaningful only within the scope of a cell, and the concatenation of the cell's ID (venture ID, unit ID, and continuum ID) and the module number uniquely identifies the module within the venture.

2.1.4 ARCHITECTURAL PERSPECTIVE

2.1.4.1 General

The manner in which AMS relates to other CCSDS Recommended Standards is discussed here in a continuation of the AMS communication configuration example presented in 2.1.3. In this example, AMS is to be used for monitoring and controlling the thermal subsystem (among others, potentially) of a rover on a distant planet. The design team for this mission has named the AMS application 'rover-ops' and is considering the use of AMS in either or both of two distinct computing environments, the ground data system on Earth and the rover's flight data system. AMS operations in the former environment are viewed as proceeding in a continuum named 'moc' (mission operations center), while AMS operations in the latter environment are viewed as proceeding in a continuum named 'rover'.

NOTE – Underlying protocols identified in the following subsections are examples. AMS can also run over Proximity-1 as well as any number of other protocols that meet the requirements of 3.2.

2.1.4.2 Ground Data System Example

2.1.4.2.1 Scenario

Telemetry packets acquired from the rover are passed to a module that is registered in the role of 'telemetry-sink' (module 3), which is a general-purpose publisher of telemetry information. Subject 'temperature' is used for the publication of telemetry pertaining to the thermal subsystem. Two modules in the ground data system subscribe to 'temperature': a real-time thermal state monitoring application (module 6) and a thermal state trend analysis application (module 8). Whenever a thermal telemetry packet is acquired from the rover, its contents are published by module 3 in an AAMS message on subject 'temperature' causing copies of this message to be immediately received by both module 6 and module 8.

2.1.4.2.2 Topology

Module 3 is shown here as a registered module in the 'root' cell of the message space for 'live rover-ops' in the 'moc' continuum. Modules 6 and 8 are registered in the 'thermal' cell of this message space.

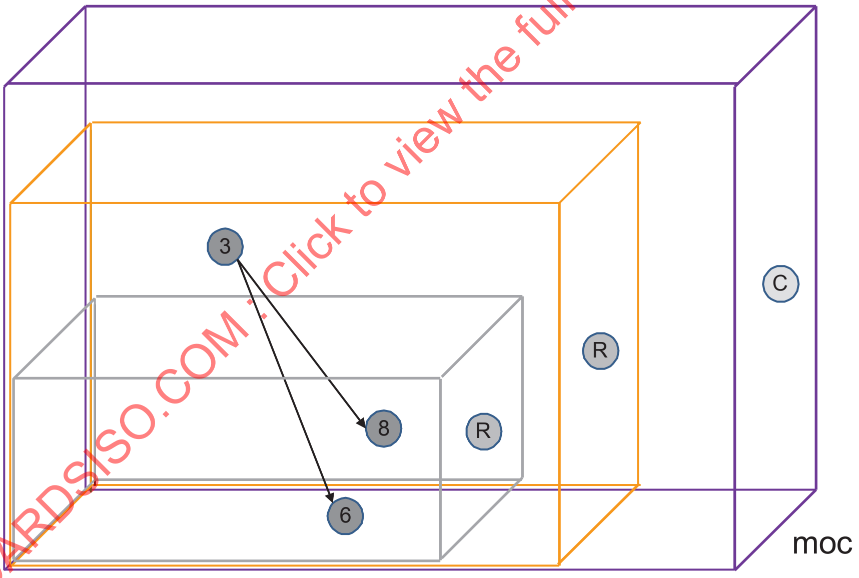


Figure 2-9: Ground Data System Registered Modules

2.1.4.2.3 Module Protocol Stack

In this example, modules 3, 6, and 8 all conform to CCSDS Spacecraft Monitor and Control architecture. The Application AMS protocol running over TCP/IP (in the mission operations center's local area network) serves as the messaging technology underlying the Messaging Abstraction Layer (MAL).

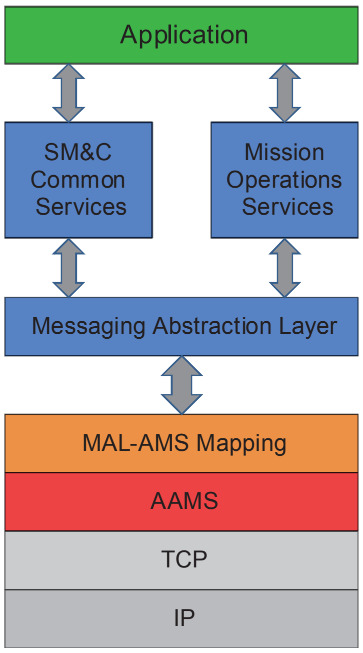


Figure 2-10: AAMS as Underlying Messaging Technology for MAL

2.1.4.3 Flight Data System Example

2.1.4.3.1 Scenario

On-board the rover, thermal sensor monitoring modules 6 and 12 periodically publish thermal sensor data in AAMS messages on subject 'temperature'. Two on-board modules subscribe to 'temperature': a real-time thermal alarm module (module #19) and a module registered in the role of 'telemetry-source' (module #4). Whenever either module 6 or module 12 publishes a 'temperature' message, both module 4 and module 19 immediately receive copies. On receipt of a 'temperature' message, module 4 issues a thermal telemetry packet for transmission to the mission operations center on Earth.

2.1.4.3.2 Topology

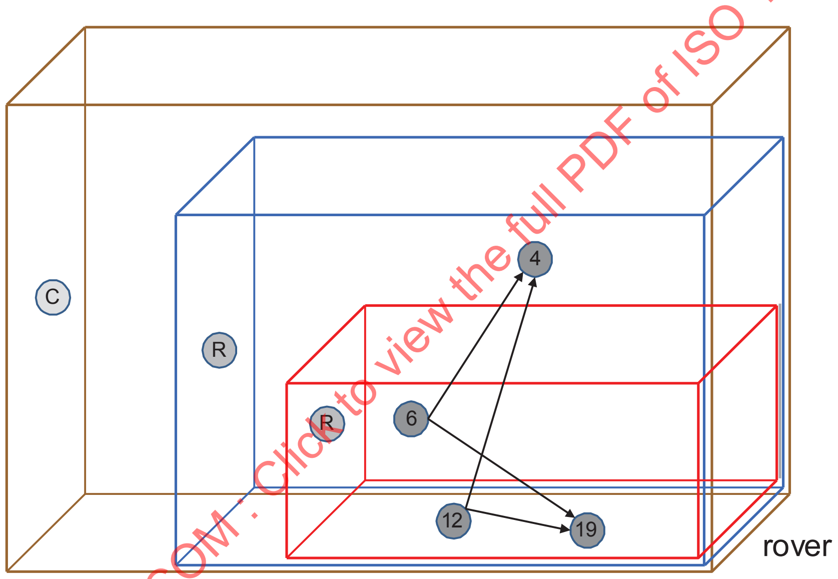


Figure 2-11: Flight Data System Registered Modules

2.1.4.3.3 Module Protocol Stack

In this example, modules 4, 6, 12, and 19 all conform to CCSDS Spacecraft Onboard Information Services (SOIS) architecture. The Application AMS protocol running over the SOIS Subnetwork Packet Service (mapped to the spacecraft's SpaceWire bus) serves as the spacecraft's Message Transfer Service.

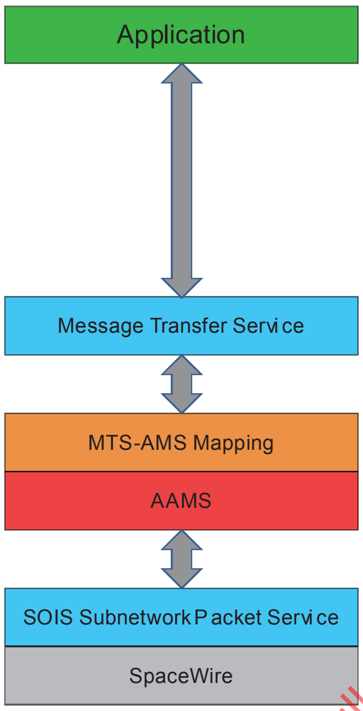


Figure 2-12: AAMS over SOIS Subnetwork Packet Service

2.1.4.4 End-to-End Mission Data System Example

2.1.4.4.1 Scenario

The mission design team have decided to use end-to-end AMS messaging to operate the rover. The mission-specific 'telemetry source' and 'telemetry sink' modules are no longer required: instead, standard RAMS gateway modules in the MOC and on the rover will manage the forwarding of command and telemetry information over the space link. Whenever either module 6 or module 12 publishes a 'temperature' message, both module 19 and module 1 (the 'live rover-ops' RAMS gateway in the 'rover' continuum) immediately receive copies. The rover's RAMS gateway uses reliable Delay/Disruption Tolerant Networking (DTN) protocols to convey the 'temperature' message to its counterpart in the MOC, which then forwards that message to the two local subscribers, modules 6 and 8.

2.1.4.4.2 Topology

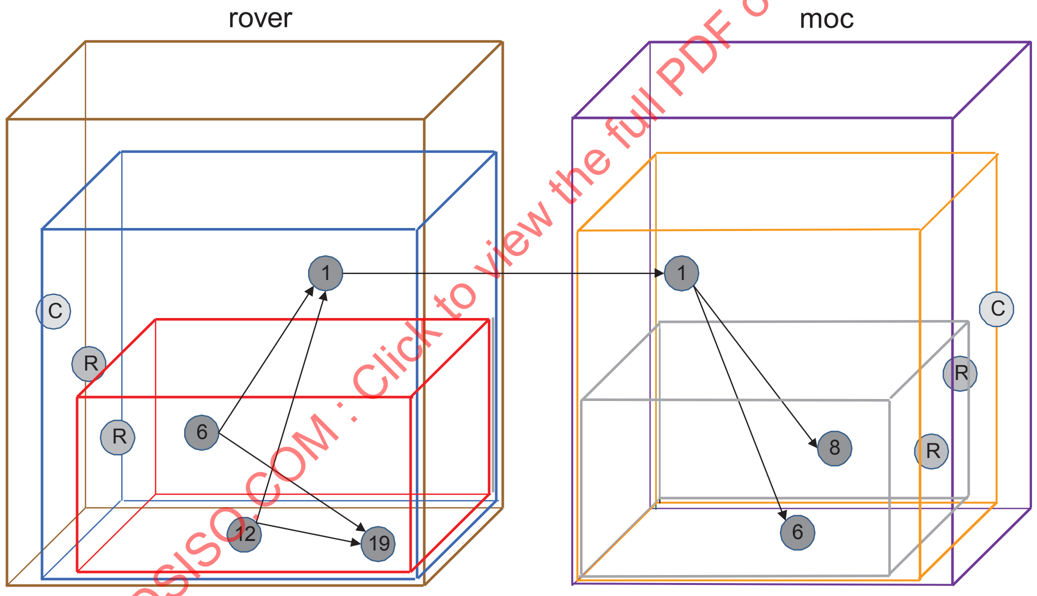


Figure 2-13: End-to-End Mission Data System Registered Modules

2.1.4.4.3 Ground Gateway Protocol Stack

In the mission operations center, the RAMS gateway uses TCP/IP for exchange of AAMS messages with other modules in its message space but uses the DTN Bundle Protocol and Licklider Transmission Protocol over CCSDS encapsulation packets and telemetry/telecommand links for exchange of RAMS messages with its counterpart on the rover.

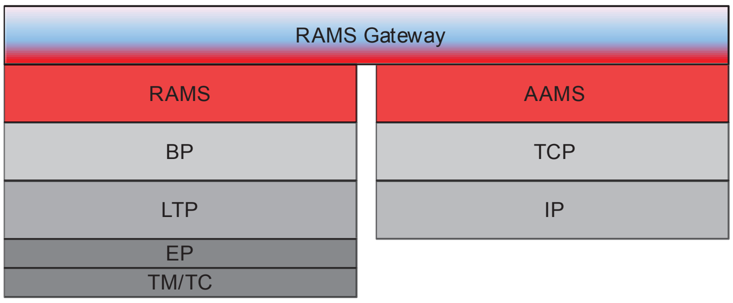


Figure 2-14: RAMS Gateway with AAMS over TCP/IP and DTN over CCSDS

2.1.4.4.4 Flight Gateway Protocol Stack

On-board the rover, the RAMS gateway uses the SOIS Subnetwork Packet Service over SpaceWire for exchange of AAMS messages with other modules in its message space but uses the DTN Bundle Protocol and Licklider Transmission Protocol over CCSDS encapsulation packets and telemetry/telecommand links for exchange of RAMS messages with its counterpart in the mission operations center.

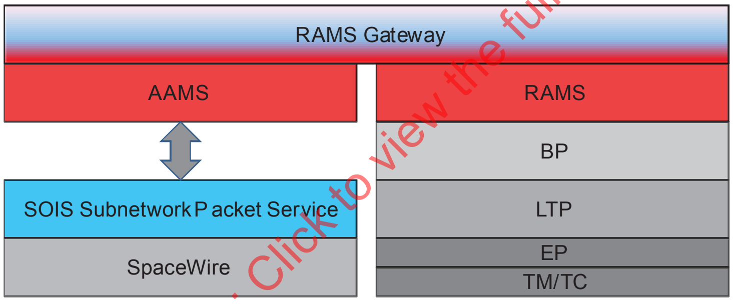


Figure 2-15: RAMS Gateway with AAMS over SOIS Subnetwork Packet Service and DTN over CCSDS

2.1.4.4.5 Enhanced Flight Module Protocol Stack

Application modules on-board the rover may use AMS either for end-to-end messaging, through Spacecraft Monitoring and Control (SM&C) services, or for local messaging.

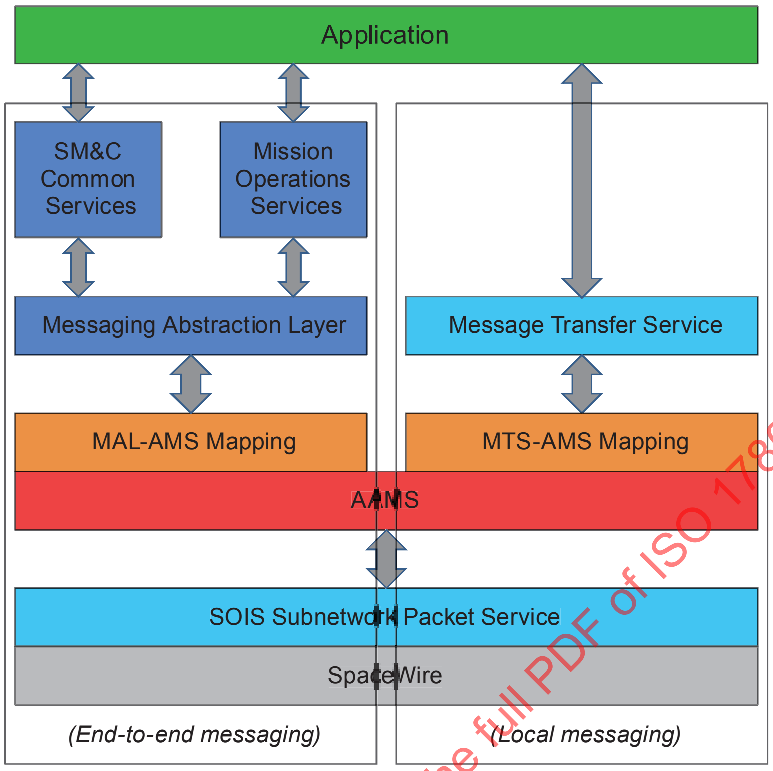


Figure 2-16: Enhanced Flight Module Protocol Stack

2.2 ARCHITECTURAL ELEMENTS

2.2.1 GENERAL

The architectural elements involved in the asynchronous message service protocols are depicted in figure 2-17 and described below.

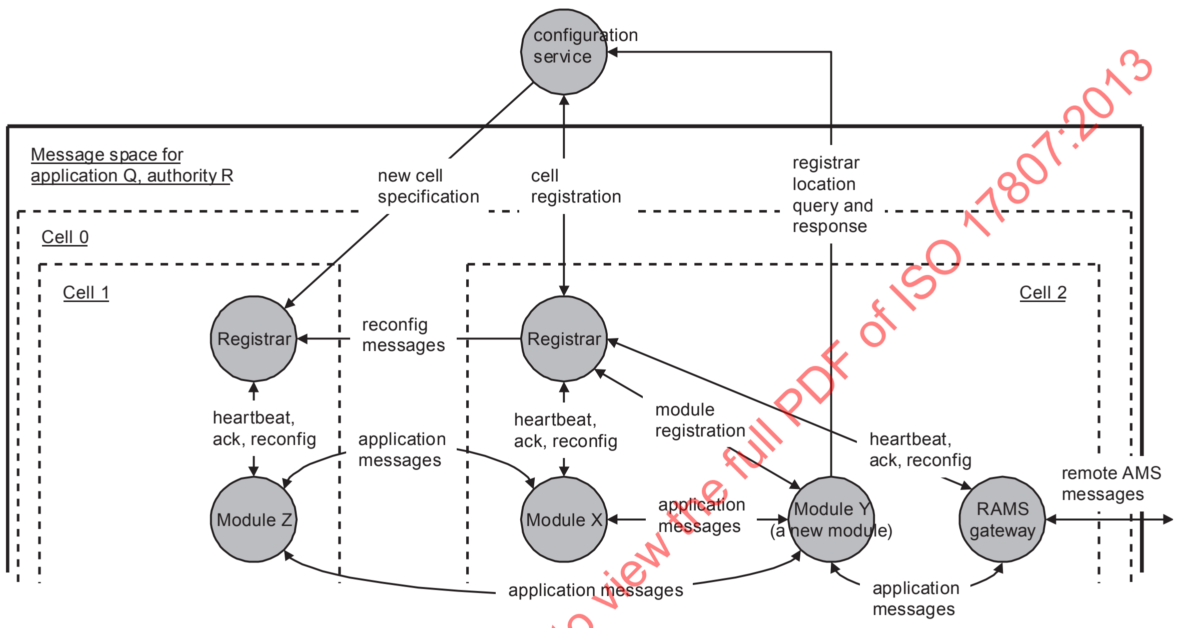


Figure 2-17: Architectural Elements of AMS (a Single Continuum)

2.2.2 COMMUNICATING ENTITIES

All AMS communication is conducted among three types of communicating entities: modules (defined earlier), registrars, and configuration servers.

A *registrar* is a communicating entity that catalogues information regarding the registered membership of a single cell of a message space. It responds to queries for this information, and it updates this information as changes are announced.

A *configuration server* is a communicating entity that manages a database of configuration information for a single continuum. In particular, it catalogues information regarding the message spaces that the continuum comprises, notably the locations of all registrars. It responds to queries for this information, and it updates this information as changes are announced.

2.3 OVERVIEW OF INTERACTIONS

2.3.1 TRANSPORT SERVICES FOR APPLICATION MESSAGES

AAMS is an OSI protocol stack layer 7 (application) service that might best be characterized as messaging 'middleware'. As such, it relies on the capabilities of underlying Transport-layer protocols to accomplish the actual copying of a message from the memory of the sending module to the memory of the receiving module. It additionally relies on those capabilities to accomplish the transmission of the MAMS messages to and from registrars and configuration servers that enable the dynamic self-configuration of AMS message spaces.

For any given AMS continuum, some common transport service must be utilized for MAMS traffic by all communicating entities involved in the operations of all message spaces in the continuum. The transport service selected for this purpose is termed the continuum's *Primary Transport Service (PTS)*.

The PTS clearly can also be used for application message exchange among all modules in a continuum, as it must be universally available for MAMS message exchange. In some cases, however, improved application performance can be achieved by using a different transport service for message exchange between modules that share access to some especially convenient communication medium, such as a shared-memory message queue. These performance-optimizing transport services are termed *Supplementary Transport Services (STSes)*.

The service access point at which a module can receive messages via a given transport service is termed the module's *delivery point* for that service. Multiple delivery points (for different transport services) may be characterized by the same *service mode*, that is, by the same levels of transmission order preservation and diligence in message delivery. For a given service mode, the list of all delivery points that provide that mode of service to a given module—in descending order of preference (generally, descending order of network performance)—is termed the *delivery vector* for that service mode, for that module.

2.3.2 REGISTRAR REGISTRATION

Every message space always comprises at least one cell (the root cell), and each module resides within (is registered in) some cell; in the simplest case all modules of the message space reside in the root cell. Each cell is served by a single registrar, which is responsible for monitoring the health of all modules in the cell's registered membership and for propagating six kinds of message space configuration changes: module registrations and terminations, subscription and invitation assertions, and subscription and invitation cancellations. On reception of one of these reconfiguration messages from a module in its own cell's registered membership, the registrar immediately propagates the message to the other modules in that registered membership and then to the registrars of other cells in the message space, as appropriate; on receiving such a message from a remote cell's registrar, the registrar propagates it to the modules in its own cell's registered membership.

The registrars themselves register with the configuration server for the continuum within which the message space is contained. A list of all possible network locations for the configuration server, ordered by some mechanism that reflects the continuum's globally recognized strict relative order among candidate configuration servers, must be 'well known'—that is, included in the AMS MIBs exposed to all registrars for all message spaces in the continuum—and each continuum must have a configuration server in operation at one of those locations at all times in order to enable registrars and modules to register. (The manner in which this latter requirement is satisfied is an implementation matter.)

All registrars and modules of the same message space must register through the same configuration server. The registrars and modules for any number of different message spaces may register with the same configuration server.

2.3.3 MODULE REGISTRATION

Each module has a single associated *Meta-AMS Delivery Point (MADP)* at which the module is prepared to receive MAMS messages. A new module joins a message space by registering itself within some cell of the message space, i.e., by announcing its role name and its MADP to the cell's registrar and thereby inserting itself into the cell's registered membership. However, knowledge of how to communicate with that registrar cannot be hard-coded into the module because the relevant registrar might be running at different network locations at different times.

For this reason, the first step in registering a new module is to contact the configuration server at one of its well-known possible network locations; as with the registrars, a list of all possible network locations for the configuration server, in order of descending preference, must be included in the AMS MIBs exposed to all application modules. The configuration server tells the new module how to contact its registrar. The module obtains a *module number* (which uniquely identifies it within the cell's registered membership) from the registrar and thereby registers. The registrar ensures that all other modules in the message space learn the new module's role name, module number, and MADP. Declaration of those modules' own role names, module numbers, and MADPs to the new module is then accomplished in one of two ways, depending on implementation: either the modules themselves send this information to the new module or else the registrar retrieves this information from its local cache and sends it directly to the new module.

2.3.4 MONITORING MODULE HEALTH

In order to acquire and maintain accurate knowledge of the configuration of a message space (for application purposes, and also to avoid wasting resources on attempts to send messages to nonexistent modules or per concluded subscriptions), it is important for each registrar always to detect the terminations of modules in its cell's registered membership. When a module terminates under application control it automatically notifies its registrar that it is stopping. However, if a module crashes, or the host on which a module resides is simply powered off or rebooted, no such notification is transmitted to the registrar. For this reason, every module automatically sends a 'heartbeat' message to its registrar every N4 seconds. (See table 1-1 for definitions of N3, N4, and N6.) The registrar interprets N6 successive missing heartbeats as an indication that the module has terminated.

Whenever it detects the termination of a module (either an overt termination or a termination imputed from heartbeat failure), the registrar informs all other modules in the cell's registered membership and, via other registrars, all other modules in the message space of the module's demise.

When the termination is imputed from a heartbeat failure, the registrar also tries to send a message to the terminated module telling it that it has been presumed dead; if this module is in fact still running (perhaps it had merely hung trying to write on a blocked file descriptor), it terminates immediately on reception of this message. This minimizes system confusion resulting from other application behavior that may have been triggered by the imputed termination.

NOTE – The value of N4 is defined as twice the value of N3, which is a global implementation parameter within any single continuum. If a very large value of N3 is selected, then heartbeat exchange is in effect disabled; that is, the monitoring of module health, registrar health, and configuration server health is optional. AMS entity health monitoring may be unnecessary under certain circumstances, e.g., in tightly controlled spacecraft on-board environments where other mechanisms for the detection of module failure are available. In these environments, the initiation of AMS procedures for handling imputed entity termination—in effect, the simulation of heartbeat failure—is an implementation matter.

2.3.5 MONITORING REGISTRAR HEALTH

In addition to monitoring the heartbeats of all modules in its cell's registered membership, each registrar issues its own heartbeats to those modules on the same cycle. Each module interprets N6 successive missing registrar heartbeats as an indication that the registrar itself has crashed. On detecting a registrar crash, the module presumes that the registrar has been restarted since it crashed; it re-queries the configuration server to determine the new network location of the registrar, reconnects to the registrar, and resumes exchanging heartbeats.

This presumption is reasonable because the reciprocal heartbeat monitoring relationship between a registrar and its modules is replicated in the relationship between the configuration server and all registrars, but on a slightly shorter cycle. The configuration server interprets N6 successive missing registrar heartbeats as an indication that the registrar has crashed; on detecting such a crash it can automatically take some implementation-specific action to cause the registrar to be restarted, possibly on a different host, so by the time the registrar's modules detect its demise it should already be running again.

Since the module heartbeat interval is N4 seconds, within the first N5 seconds after restart the registrar will have received heartbeat messages from every module in the cell's registered membership that is still running and will therefore know accurately the configuration of the cell.

NOTE – The worst-case interval between failure and recovery of registration service within a cell is N5 seconds. In the event that N3 is a very large number (disabling heartbeat exchange, as discussed above), implementation-specific mechanisms for simulating the effects of heartbeat failure will reduce this recovery interval to far less than N5 seconds.

This accurate configuration information must be delivered to new modules at the time they start up (so that they in turn are qualified to orient a newly restarted registrar to the cell's configuration in the event that the registrar crashes). For this reason, during the first N5 seconds (or other implementation-specific recovery interval, as noted above) after the registrar starts it accepts only MAMS messages from modules that are already registered in the cell (i.e., have been assigned module numbers); if it accepted and processed a registration message from a new module before being certain of the status of all old ones, it would run the risk of delivering incorrect information to the new module.

2.3.6 CONFIGURATION SERVICE FAIL-OVER

It is of course also possible for a configuration server to be killed (or for its host to be rebooted, etc.). Each registrar interprets N6 successive missing configuration server heartbeats as an indication that the configuration server has crashed. On detecting such a crash, the registrar begins cycling through all of the well-known possible network locations for the continuum's configuration server, trying to re-establish communication after the server's restart, possibly at an alternate network location. Moreover, so long as the configuration server is down no new module can join any message space in the continuum, since new modules will have no way of knowing the network locations of their intended registrars; they too will be cycling through all of the well-known possible network locations for the continuum's configuration server, trying to establish initial communications.

When the configuration server is restarted at one of its well-known possible network locations, however, all registrars will eventually[1] find it and re-announce themselves to it; new module registration activity will thereupon resume.

It is possible, in this sort of failure scenario, that multiple configuration servers may be operating concurrently for a brief time; for example, the perceived failure of a configuration server might have been caused by a transient network connectivity failure rather than an actual server crash. To resolve this sort of situation, each running configuration server periodically sends an 'I am running' MAMS message to every lower-ranking configuration server network location in the well-known list of configuration server locations. When a configuration server receives such a message, it immediately terminates; all registrars and modules that were communicating with it will detect its disappearance and search again for the highest-ranking reachable configuration server, eventually bringing the continuum back to orderly operations.

2.3.7 CONFIGURATION RESYNCHRONIZATION

Finally, every registrar can optionally be configured to re-advertise to the entire message space the detailed configuration of its cell's registered membership (all active modules, all subscriptions and invitations) at some user-specified frequency, e.g., once per minute. This capability is referred to as *configuration resynchronization*. Configuration resynchronization of course generates additional message traffic, and it may be unnecessary in extremely simple or extremely stable operating environments. But it does ensure that every change in message space configuration will eventually be propagated to every module in the message space, even if some MAMS messages are lost and even if an arbitrary number of registrars had crashed at the time the change occurred.

Taken together, these measures make AMS applications tolerant of several types of faults:

– When a module crashes, its registrar detects the loss of heartbeat within N6 heartbeat intervals and notifies the rest of the message space. Application message transmission among the remaining modules in the message space is unaffected.

– When a registrar crashes, its configuration server detects the loss of heartbeat within N6 heartbeat intervals and takes action to restart the registrar. During the time that the cell has no registrar, transmission of application messages among modules of the message space is unaffected, but the heartbeat failures of crashed modules are not detected and reconfiguration messages issued by the modules in the cell (registrations, terminations, subscription and invitation assertions, and subscription and invitation cancellations) are not propagated to any other modules. However, after the registrar is restarted it will eventually detect the losses of heartbeat from all crashed modules and will issue obituaries to the message space; in addition, if configuration

[1] The AMS Green Book contains notes on an algorithm for estimating the worst-case elapsed time prior to re-establishment of module registration activity.

resynchronization is enabled it will eventually re-propagate the lost reconfiguration messages.

– When a configuration server crashes, registration of new modules ceases until the configuration server is restarted. But no application modules fail (at least, not because of communication failure), and on restart of the configuration server all new module registration eventually resumes.

2.3.8 SECURITY

NOTE – *The Application of CCSDS Protocols to Secure Systems* (CCSDS 350.0-G-2) should be consulted for definitions of the terms used in this subsection.

Optionally, AMS can be configured to confine service access to application modules that can prove they are authorized to participate. For this purpose, asymmetric MAMS encryption may be used as follows:

– The AMS MIB exposed to the configuration server contains a list of all applications for which registration service may be offered, identified by application name. Associated with each application name is the AMS public encryption key for that application.

– The AMS MIB exposed to every registrar in each message space contains a list of all functional role names defined for the message space's application; this list limits the role names under which modules may register in that message space. Associated with each role name is the AMS public encryption key for the application module(s) that may register in that role.

– The AMS MIBs exposed to all registrars and application modules in the message space contain the AMS public encryption key of the configuration server.

– The AMS MIBs exposed to the configuration server and to all registrars and application modules in the message space contain the private encryption keys that are relevant to those entities.

As described later, this information is used to authenticate registrar registration and exclude spurious registrars from the message space, to authenticate module registration attempts and deny registration to unauthorized application modules, and to assure the authenticity, confidentiality, and integrity of MAMS traffic exchanged between modules and their registrars.

In addition, the confidentiality and integrity of AMS message exchange may be protected at subject granularity. The AMS MIB exposed to each module of a given message space may contain, for any subset of the message subjects (identified by name and number) used in the message space's application:

- a list of the role names of all modules that are authorized senders of messages on this subject;
- a list of the role names of all modules that are authorized receivers of messages on this subject;
- encryption parameters, including a symmetric encryption key, enabling encryption of messages on this subject.

This information may be used to support secure transmission of messages on selected subjects.

The structure of security information elements and the manner in which MIBs are populated with these elements are implementation matters.

2.3.9 SUBJECT CATALOG

The structure of the application data conveyed by messages on a given subject is application-specific; application data structure is not defined by the AMS protocol. However, the AMS MIB exposed to all modules of a given message space may optionally contain one or more of the following items of information for each message subject (identified by name and number) used in the message space:

- a description of this message subject, discussing the semantics of this type of message;
- a detailed specification of the structure of the application data conveyed by messages on this subject;
- a specification of the manner in which a correctly assembled message is *marshaled* (that is, converted from processor-internal representation to an array of octets) for network transmission in a platform-neutral manner and, on reception, un-marshaled into a format that is suitable for processing by the application.

When AAMS is requested to send a message on a given subject, the application data that is presented for transmission is always in a format that is suitable for processing by the application. In the event that this format is not suitable for network transmission in a platform-neutral manner, as indicated by the presence in the MIB of a marshaling specification for this subject, AAMS will marshal the application data as required before transmitting the message.

When AAMS receives a message on a subject for which a marshaling specification is present in the MIB, AAMS will un-marshal the message's application data into a format that is suitable for processing by the application before delivering the message.

The structure of subject catalog information elements, the manner in which MIBs are populated with these elements (e.g., by linkage to an external information model), and the manner in which message application data are marshaled and un-marshaled are implementation matters.

Message subjects, as noted earlier, are integers with application-defined semantics. This minimizes the cost of including subject information (in effect, message type) in every message, and it can make processing in an AMS implementation simpler and faster: subscription and invitation information may be recorded in (possibly sparse) arrays that are indexed by subject number.

2.3.10 REMOTE AMS MESSAGE EXCHANGE

Because issuance of an asynchronous message need not suspend the operation of the issuing module until a response is received, AAMS's message exchange model enables a high degree of concurrency in the operations of data system modules; it also largely insulates applications from variations in signal propagation time between points in the AMS continuum.

However, some critical MAMS communication is unavoidably synchronous in nature: in particular, a newly registering module must wait for responses from the configuration server and the registrar before proceeding with application activity. For this reason, the AAMS protocol is most suitable for use in operational contexts characterized by generally uninterrupted network connectivity and relatively small and predictable signal propagation times, such as the Internet or a stand-alone local area network. It is usually advantageous for all entities of any single AMS continuum to be running within one such 'low-latency' network.

AAMS application messages may readily be exchanged between modules in different AMS continua, however, by means of the *Remote AMS* (*RAMS*) procedures. RAMS procedures are executed by special-purpose application modules called *RAMS gateways*. Each RAMS gateway has interfaces to two communication environments: the AMS message space it serves and the *RAMS network*—the mesh or tree of mutually aware RAMS gateways—that enables AAMS messages produced in one message space to be forwarded to other message spaces of the same venture. RAMS gateways operate as follows:

– Each RAMS gateway opens persistent RAMS network communication channels to the RAMS gateways of other message spaces for the same venture, in other continua. The interconnected RAMS gateways use these communication channels to forward message petition assertions and cancellations among themselves.

– Each RAMS gateway subscribes locally to all subjects that are of interest in any of the linked message spaces.

– On receiving its copy of a message on any of these subjects, the RAMS gateway module uses the RAMS network to forward the message to every other RAMS gateway whose message space contains at least one module that has subscribed to messages on that subject.

– On receiving a message via the RAMS network from some other RAMS gateway, the RAMS gateway module forwards the message to all subscribers in its own message space.

RAMS operations generalize the AMS architecture as shown in figure 2-18 below:

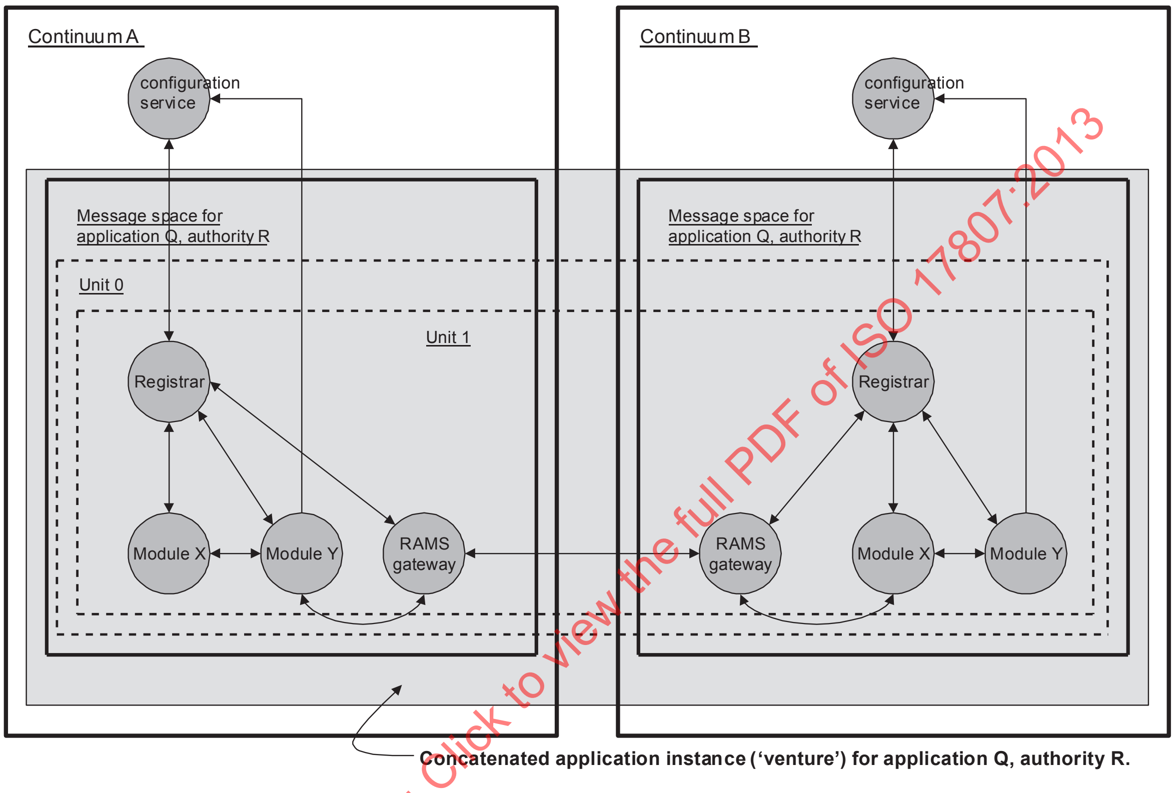

Continuum A
configuration service
Message space for application Q, authority R
Unit 0
Unit 1
Registrar
Module X
Module Y
RAMS gateway
Continuum B
configuration service
Message space for application Q, authority R
Registrar
RAMS gateway
Module X
Module Y
Concatenated application instance ('venture') for application Q, authority R.

Figure 2-18: General Remote AMS Architecture

In this way the RAMS protocol enables the free flow of published application messages across arbitrarily long deep space links while protecting efficient utilization of those links: only a single copy of any message is ever transmitted on any RAMS network communication channel, no matter how many subscribers will receive copies when the message reaches its destination message space.

The nature of the RAMS network communication channels depends upon the implementation of the RAMS network. In order to communicate over the RAMS network for a given venture, each RAMS gateway must know the *RAMS network location*—expressed as an endpoint in the protocol used to implement the RAMS network—at which every neighboring RAMS gateway for that venture receives RAMS network traffic. Additional Management Information Base elements may be required for this purpose, but the configuration of RAMS networks is an implementation matter that is beyond the scope of this Recommended Standard.

Again, this extension of the publish/subscribe model to interplanetary communications is invisible to application modules. Application functionality is unaffected by these details of network configuration, and the only effects on behavior are those that are intrinsic to variability in message propagation latency.

NOTE – Clearly, RAMS cannot insulate applications from the effect of message propagation latency variation on conversational message exchange.

3 SERVICE DESCRIPTIONS

3.1 SERVICES PROVIDED TO THE APPLICATION

3.1.1 SUMMARY OF PRIMITIVES

3.1.1.1 The AMS service shall consume the following request primitives:

a) **Register.request**;

b) **Unregister.request**;

c) **Assert_invitation.request**;

d) **Cancel_invitation.request**;

e) **Assert_subscription.request**;

f) **Cancel_subscription.request**;

g) **Publish.request**;

h) **Send.request**;

i) **Query.request**;

j) **Reply.request**;

k) **Announce.request.**

3.1.1.2 The AMS service shall deliver the following indication primitives:

a) **Message.indication**;

l) **Query.indication**;

b) **Reply.indication**;

c) **Fault.indication**;

d) **Register.indication**;

e) **Unregister.indication**;

f) **Assert_invitation.indication**;

g) **Cancel_invitation.indication**;

h) **Assert_subscription.indication**;

i) **Cancel_subscription.indication**;

j) **Module_is_dead.indication**.

3.1.2 SERVICE PRIMITIVE PARAMETERS

NOTE – The availability and use of parameters for each primitive are indicated in the definitions of primitives below, where parameters that are optional are identified with square brackets [thus]. The following parameter definitions apply.

3.1.2.1 The *continuum ID* parameter (name or number) shall identify the continuum (or continua) of AMS service to which the service primitive pertains. The continuum name that is the character string of length zero indicates 'all known continua' or 'any known continuum', whichever is less restrictive in the context in which this continuum name is used; the reserved continuum number 0 corresponds to this continuum name. There shall be a one-to-one correspondence between AMS continua and authoritative AMS configuration servers. Continuum ID is required for Remote AMS (RAMS) service operations, where it enables messages passed between continua to be correctly dispatched.

3.1.2.2 The *application name* parameter shall identify the application served by a message space's venture.

3.1.2.3 The *authority name* parameter shall identify the administrative entity or persona that is responsible for a message space's venture. The combination of application name and authority name shall uniquely identify a message space within its continuum.

3.1.2.4 The *unit ID* parameter (name or number) shall identify some identified subset of the organizational hierarchy of the modules in a message space's venture. The unit name that is the character string of length zero indicates the root unit, and the reserved unit number 0 corresponds to this unit name.

3.1.2.5 The *role ID* parameter (name or number) shall indicate the functional nature (role performed within the application) of a module. The role name that is the character string of length zero indicates 'all roles' or 'any role', whichever is less restrictive in the context in which the role name is used; the reserved role number 0 corresponds to this role name. The role name 'RAMS' identifies RAMS gateway functionality as discussed below; the reserved role number '1' corresponds to this role name.

3.1.2.6 The *Meta-AMS delivery point (MADP) specification* parameter shall be a *transport service endpoint specification* characterizing the manner in which a module is prepared to receive MAMS messages. The MADP specification shall identify a functional endpoint of the AMS continuum's primary transport service. The syntax in which a transport service endpoint specification is represented is transport service-specific; definitions of valid endpoint specification syntax for all recognized transport services are given in annex A.

3.1.2.7 The *module number* parameter shall uniquely identify a module within the cell in which it is registered. Module number 0 shall be reserved, signifying 'none'.

3.1.2.8 The *subject ID* (name or number) parameter shall indicate the general nature of the application data in a message. The subject name that is the character string of length zero indicates 'all subjects'; the reserved subject number 0 corresponds to this subject name.

3.1.2.9 The *delivery specification* parameter shall characterize the manner in which a module is prepared to receive AAMS and MAMS messages. The delivery specification shall comprise a list of one or more *delivery point specifications*. Delivery point specifications shall be listed in declining order of preference; that is, the delivery point on which the module most prefers to receive messages shall be specified first, followed by the next-most-preferred delivery point, and so on. Each delivery point specification shall associate a *transport service name* with a transport service endpoint specification as described earlier. The transport service name shall be the name of the AMS continuum's primary transport service or the name of one of the continuum's supplementary transport services. Standard delivery preference rules declared in the Management Information Base shall, in turn, identify the *service mode* (defined below) that characterizes each named transport service. In this way, the delivery specification not only explicitly indicates the network locations to which messages are to be sent but also implicitly indicates the quality of service with which messages must be sent to those locations.

3.1.2.10 The *service mode* parameter shall indicate the service mode governing issuance of messages on a given subject. The service mode shall indicate (a) the *sequence* in which messages are delivered (Transmission order—nominally implying that message reordering upon arrival is performed automatically as necessary—or Arrival order) and (b) the *diligence* with which delivery of messages on this subject is attempted (Assured—nominally implying automatic acknowledgement and retransmission as necessary—or Best-effort).

3.1.2.11 The *priority* parameter shall indicate the relative urgency of messages issued on a given subject. Priority shall be an integer in the range 1 to 15, where 1 signifies the greatest urgency and 15 the least.

3.1.2.12 The *flow label* parameter shall identify the message stream or 'flow' to which messages on a given subject contribute. Flow label is opaque to AMS but when passed through to transport services may be used to invoke additional, managed quality-of-service features. For example, a flow label might identify—or be mapped to, in an implementation-dependent manner—an IP differentiated services code point, a delivery deadline specification, or a SOIS Packet Service channel. Flow label shall be an integer in the range 0 to 65535.

3.1.2.13 The *context* parameter shall be an integer that identifies the application context in which a message was sent, if any. The significance and interpretation of this number are opaque to AMS. Conceptually, the context parameter functions within AMS as a message identifier. The context in which a reply is sent must be identical to the context in which the reply's antecedent message was sent.

3.1.2.14 The *application data length* parameter shall indicate the length (in octets) of the *application data* parameter. The maximum value of this parameter shall be 65000.

3.1.2.15 The *application data* parameter shall be an array of 0—65000 octets comprising the application data in a message, in a format that is suitable for processing by the application.

3.1.2.16 The *term* parameter shall indicate the length of time following message transmission within which a reply message should be received. If the term expires before the reply is received, the query has 'timed out'. Selection of a term parameter value that indicates that the term is indefinite, i.e., it never expires, is an implementation matter.

3.1.2.17 The *fault expression* parameter shall indicate the nature of an operational fault encountered by AMS. The syntax of fault expressions is an implementation matter.

3.1.2.18 The *service access point (SAP)* parameter shall indicate the AMS service access point at which the service primitive is enacted. The nature of the SAP and the means by which it is exposed to the AMS application are implementation matters.

3.1.3 REGISTER.REQUEST

3.1.3.1 Function

The **Register.request** primitive shall be used by the module to commence its participation in a message space.

3.1.3.2 Semantics

Register.request shall provide parameters as follows:

```
Register.request (SAP, application name,
                  authority name,
                  unit ID,
                  role ID,
                  [MADP specification],
                  [delivery specification])
```

3.1.3.3 When Generated

Register.request is generated by the module at any time while the module is not currently participating in its message space.

3.1.3.4 Effect on Reception

Reception of **Register.request** shall, if approved, cause AMS to add the module to the indicated message space cell.

3.1.3.5 Additional Comments

If MADP specification is omitted, a default MADP specification based on preferences noted in the AMS MIB shall be computed. If delivery specification is omitted, a default delivery specification based on the standard delivery preferences noted in the module's AMS MIB shall be computed.

3.1.4 UNREGISTER.REQUEST

3.1.4.1 Function

The **Unregister.request** primitive shall be used by the module to terminate its participation in a message space.

3.1.4.2 Semantics

Unregister.request shall provide parameters as follows:

Unregister.request (SAP, module number)

3.1.4.3 When Generated

Unregister.request is generated by the module at any time while the module is participating in its message space.

3.1.4.4 Effect on Reception

Reception of **Unregister.request** shall cause AMS to remove the module from the indicated message space cell.

3.1.4.5 Additional Comments

The module number provided in the **Unregister.request** primitive must be the one that was provided in the **Register.indication** primitive that notified the module of its own successful registration.

NOTE – The manner in which a particular AMS service indication may be traced back to the particular AMS service request that caused its delivery (e.g., by some sort of token or other request identifier) is an implementation matter.

3.1.5 ASSERT_INVITATION.REQUEST

3.1.5.1 Function

The **Assert_invitation.request** primitive shall be used by the module to indicate the manner in which private (that is, non-published) messages on a specific subject may be delivered to the module.

3.1.5.2 Semantics

Assert_invitation.request shall provide parameters as follows:

```
Assert_invitation.request (SAP, subject ID,
                          service mode,
                          [priority,]
                          [flow label,]
                          continuum ID of invitee(s),
                          unit ID of invitee(s),
                          role ID of invitee(s))
```

3.1.5.3 When Generated

Assert_invitation.request is generated by the module at any time while the module is participating in its message space.

3.1.5.4 Effect on Reception

Reception of **Assert_invitation.request** shall, if approved, cause AMS to notify all modules in the message space of the module's willingness to accept private messages on the indicated subject from modules in the domain of the service request.

3.1.5.5 Additional Comments

The priority and flow label specified in the **Assert_invitation.request** are advisory, expressing the module's preferences for delivery of messages on this subject. Where omitted, the default value for priority shall be eight, and the default value for flow label shall be zero. Invitations are distinct from subscriptions: they enable private transmission of messages (*send*, *reply*, *query, announce*) on the indicated subject and **do not** mandate delivery of a copy of each published message on this subject. The 'invitee(s)' parameters in this service request template identify the domain of the request (the modules from which messages will be accepted), **not** the module that is issuing the request.

3.1.6 CANCEL_INVITATION.REQUEST

3.1.6.1 Function

The **Cancel_invitation.request** primitive shall be used by the module to indicate that private messages on a specific subject may no longer be delivered to the module.

3.1.6.2 Semantics

Cancel_invitation.request shall provide parameters as follows:

```
Cancel_invitation.request(SAP, subject ID,
                          continuum ID of invitee(s),
                          unit ID of invitee(s),
                          role ID of invitee(s))
```

3.1.6.3 When Generated

Cancel_invitation.request is generated by the module at any time while the module is participating in its message space.

3.1.6.4 Effect on Reception

Reception of **Cancel_invitation.request** shall cause AMS to notify all modules in the message space that the module has rescinded its willingness to accept private messages on the indicated subject from modules in the domain of the service request.

3.1.6.5 Additional Comments

The ‘invitee(s)’ parameters in this service request template identify the domain of the request (the modules from which messages will be accepted), **not** the module that is issuing the request.

3.1.7 ASSERT_SUBSCRIPTION.REQUEST

3.1.7.1 Function

The **Assert_subscription.request** primitive shall be used by the module to subscribe to published messages on a specific subject.

3.1.7.2 Semantics

Assert_subscription.request shall provide parameters as follows:

Assert_subscription.request (SAP, subject ID,
service mode,
[priority,]
[flow label,]
continuum ID of publisher(s),
unit ID of publisher(s),
role ID of publisher(s))

3.1.7.3 When Generated

Assert_subscription.request is generated by the module at any time while the module is participating in its message space.

3.1.7.4 Effect on Reception

Reception of **Assert_subscription.request** shall, if approved, cause AMS to notify all modules in the message space of the module's subscription to messages on the indicated message subject from modules in the domain of the service request.

3.1.7.5 Additional Comments

The priority and flow label specified in the **Assert_subscription.request** are advisory, expressing the module's preferences for delivery of messages on this subject. Where omitted, the default value for priority shall be eight, and the default value for flow label shall be zero. The 'publisher(s)' parameters in this service request template identify the domain of the request, **not** the module that is issuing the request.

3.1.8 CANCEL_SUBSCRIPTION.REQUEST

3.1.8.1 Function

The **Cancel_subscription.request** primitive shall be used by the module to terminate its subscription to published messages on a specific subject.

3.1.8.2 Semantics

Cancel_subscription.request shall provide parameters as follows:

```
Cancel_subscription.request  (SAP, subject ID,
                              continuum ID of publisher(s),
                              unit ID of publisher(s),
                              role ID of publisher(s))
```

3.1.8.3 When Generated

Cancel_subscription.request is generated by the module at any time while the module is participating in its message space.

3.1.8.4 Effect on Reception

Reception of **Cancel_subscription.request** shall cause AMS to notify all modules in the message space that the module has rescinded its subscription to messages on the indicated message subject from modules in the domain of the service request.

3.1.8.5 Additional Comments

The ‘publisher(s)’ parameters in this service request template identify the domain of the request, **not** the module that is issuing the request.

3.1.9 PUBLISH.REQUEST

3.1.9.1 Function

The **Publish.request** primitive shall be used by the module to publish a message.

3.1.9.2 Semantics

Publish.request shall provide parameters as follows:

```
Publish.request (SAP, subject ID,
                 [priority],
                 [flow label],
                 application data length,
                 [application data],
                 [context])
```

3.1.9.3 When Generated

Publish.request is generated by the module at any time while the module is participating in its message space.

3.1.9.4 Effect on Reception

Reception of **Publish.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the application data as necessary) and send one copy of that message to every module in the message space that currently subscribes to messages on the indicated message subject from the operative module.

3.1.9.5 Additional Comments

Context, if specified, identifies context information that is meaningful to the publishing module. Priority and flow label, where specified, override any priority and flow label preferences expressed in the applicable subscriptions as asserted by the implied destination modules.

3.1.10 SEND.REQUEST

3.1.10.1 Function

The **Send.request** primitive shall be used by the module to send a message privately to a single module.

3.1.10.2 Semantics

Send.request shall provide parameters as follows:

```
Send.request (SAP, continuum ID of destination,
              unit ID of destination,
              module number of destination,
              subject ID,
              [priority],
              [flow label],
              application data length,
              [application data],
              [context])
```

3.1.10.3 When Generated

Send.request is generated by the module at any time while the module is participating in its message space.

3.1.10.4 Effect on Reception

Reception of **Send.request** shall, if approved and if the destination module currently invites messages on the indicated subject from the operative module, cause AMS to construct a message as indicated (marshaling the application data as necessary) and send it to the specified destination module.

3.1.10.5 Additional Comments

Context, if specified, identifies context information that is meaningful to the sending module. Priority and flow label, where specified, override any priority and flow label preferences expressed in the applicable invitation as asserted by the destination module.

3.1.11 QUERY.REQUEST

3.1.11.1 Function

The **Query.request** primitive shall be used by the module to send a message privately to a single module, requesting a reply.

3.1.11.2 Semantics

Query.request shall provide parameters as follows:

Query.request (SAP, continuum ID of destination,
unit ID of destination,
module number of destination,
subject ID,
[priority],
[flow label],
application data length,
[application data],
[term],
context)

3.1.11.3 When Generated

Query.request is generated by the module at any time while the module is participating in its message space.

3.1.11.4 Effect on Reception

Reception of **Query.request** shall, if approved and if the destination module currently invites messages on the indicated subject from the operative module, cause AMS to construct a message as indicated (marshaling the application data as necessary) and send it to the specified destination module. If term is specified, the operation of the querying module may be suspended until either a reply to this message is received or—if *term* is greater than zero—the indicated term has expired.

3.1.11.5 Additional Comments

Context identifies context information that is meaningful to the querying module. Priority and flow label, where specified, override any priority and flow label preferences expressed in the applicable invitation as asserted by the destination module.

3.1.12 REPLY.REQUEST

3.1.12.1 Function

The **Reply.request** primitive shall be used by the module to reply to a message sent by some module.

3.1.12.2 Semantics

Reply.request shall provide parameters as follows:

Reply.request (SAP, continuum ID of destination,
unit ID of destination,
module number of destination,
subject ID,
[priority],
[flow label],
application data length,
[application data],
context)

3.1.12.3 When Generated

Reply.request is generated by the module at any time while the module is participating in its message space.

3.1.12.4 Effect on Reception

Reception of **Reply.request** shall, if approved and if the destination module currently invites messages on the indicated subject from the operative module, cause AMS to construct a message as indicated (marshaling the application data as necessary) and send it to the specified destination module.

3.1.12.5 Additional Comments

Continuum ID, unit ID, and module number must identify the module that sent some previously received message, and context must be the context provided with that message (which will be meaningful only to that module). Priority and flow label, where specified, override any priority and flow label preferences expressed in the applicable invitation as asserted by the destination module.

3.1.13 ANNOUNCE.REQUEST

3.1.13.1 Function

The **Announce.request** primitive shall be used by the module to send a message privately to a set of modules.

3.1.13.2 Semantics

Announce.request shall provide parameters as follows:

Announce.request (SAP, continuum ID,
unit ID,
role ID,
subject ID,
[priority],
[flow label],
application data length,
[application data],
[context])

3.1.13.3 When Generated

Announce.request is generated by the module at any time while the module is participating in its message space.

3.1.13.4 Effect on Reception

Reception of **Announce.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the application data as necessary) and send it to every module in the domain of the service request that currently invites messages on the indicated subject from the operative module.

3.1.13.5 Additional Comments

Priority and flow label, where specified, override any priority and flow label preferences expressed in the applicable invitations as asserted by the implied destination modules.

3.1.14 MESSAGE.INDICATION

3.1.14.1 Function

The **Message.indication** primitive shall be used to deliver AMS unary message information to the module.

3.1.14.2 Semantics

Message.indication shall provide parameters as follows:

Message.indication (SAP, continuum ID of source,
unit ID of source,
module number of source,
subject ID,
priority,
flow label,
application data length,
[application data],
[context])

3.1.14.3 When Generated

Message.indication is generated upon reception of an original (non-reply) message from a module.

3.1.14.4 Effect on Reception

The effect on reception of **Message.indication** by a module is undefined.

3.1.14.5 Additional Comments

None.

3.1.15 QUERY.INDICATION

3.1.15.1 Function

The **Query.indication** primitive shall be used to deliver AMS query message information to the module.

3.1.15.2 Semantics

Query.indication shall provide parameters as follows:

Query.indication (SAP, continuum ID of source,
unit ID of source,
module number of source,
subject ID,
priority,
flow label,
application data length,
[application data],
context)

3.1.15.3 When Generated

Query.indication is generated upon reception of a query message from a module.

3.1.15.4 Effect on Reception

The effect on reception of **Query.indication** by a module is undefined.

3.1.15.5 Additional Comments

Reception of a **Query.indication** indicates that the module that sent the message requests that a reply message be sent back to it in response. The reply message must be destined for the indicated source and must echo the indicated context.

3.1.16 REPLY.INDICATION

3.1.16.1 Function

The **Reply.indication** primitive shall be used to deliver AMS reply message information to the module.

3.1.16.2 Semantics

Reply.indication shall provide parameters as follows:

Reply.indication (SAP, continuum ID of source,
unit ID of source,
module number of source,
subject ID,
priority,
flow label,
application data length,
[application data],
context)

3.1.16.3 When Generated

Reply.indication is generated upon reception of a reply message from a module.

3.1.16.4 Effect on Reception

The effect on reception of **Reply.indication** by a module is undefined.

3.1.16.5 Additional Comments

Context is the context in which the antecedent message (the message to which the reply message is a response) was sent.

3.1.17 FAULT.INDICATION

3.1.17.1 Function

The **Fault.indication** primitive shall be used to indicate an AMS fault condition to the module.

3.1.17.2 Semantics

Fault.indication shall provide parameters as follows:

Fault.indication (SAP, fault expression)

3.1.17.3 When Generated

Fault.indication is generated when AMS encounters a fault condition.

3.1.17.4 Effect on Reception

The effect on reception of **Fault.indication** by a module is undefined.

3.1.17.5 Additional Comments

None.

3.1.18 REGISTER.INDICATION

3.1.18.1 Function

The **Register.indication** primitive shall be used to notify the module of the insertion of some module (including itself) into the message space.

3.1.18.2 Semantics

Register.indication shall provide parameters as follows:

```
Register.indication (SAP, module number of new module,
                     unit ID of new module,
                     role ID of new module)
```

3.1.18.3 When Generated

Register.indication is always generated upon insertion of the module into its own message space. **Register.indication** may optionally also be generated upon insertion of any other module into the message space.

3.1.18.4 Effect on Reception

The effect on reception of **Register.indication** by a module is undefined.

3.1.18.5 Additional Comments

The module number in the **Register.indication** primitive that notifies the module of its own successful registration is the one that the module must provide in the **Unregister.request** primitive.

3.1.19 UNREGISTER.INDICATION

3.1.19.1 Function

The **Unregister.indication** primitive shall be used to notify the module of the removal of some module (including itself) from the message space.

3.1.19.2 Semantics

Unregister.indication shall provide parameters as follows:

Unregister.indication (SAP, module number of removed module,
unit ID of removed module)

3.1.19.3 When Generated

Unregister.indication is always generated upon a module's removal of itself from the message space and is optionally generated upon removal of any other module from the message space.

3.1.19.4 Effect on Reception

The effect on reception of **Unregister.indication** by a module is undefined.

3.1.19.5 Additional Comments

None.

3.1.20 ASSERT_INVITATION.INDICATION

3.1.20.1 Function

The **Assert_invitation.indication** primitive shall be used to notify the module of a newly asserted message invitation in the message space.

3.1.20.2 Semantics

Assert_invitation.indication shall provide parameters as follows:

Assert_invitation.indication (SAP, module number of inviting module,
unit ID of inviting module,
subject ID,
service mode,
priority,
flow label,
continuum ID of invitee(s),
unit ID of invitee(s),
role ID of invitee(s))

3.1.20.3 When Generated

Assert_invitation.indication is optionally generated upon assertion of an invitation by some module in the message space.

3.1.20.4 Effect on Reception

The effect on reception of **Assert_invitation.indication** by a module is undefined.

3.1.20.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the module's understanding of the functional 'connections' between itself and other modules. It should be noted that 'invitees' are modules that are invited to send messages on the indicated subject to the inviting module.

3.1.21 CANCEL_INVITATION.INDICATION

3.1.21.1 Function

The **Cancel_invitation.indication** primitive shall be used to notify the module of a newly canceled message invitation in the message space.

3.1.21.2 Semantics

Cancel_invitation.indication shall provide parameters as follows:

Cancel_invitation.indication (SAP, module number of disinviting module,
unit ID of disinviting module,
subject ID,
continuum ID of invitee(s),
unit ID of invitee(s),
role ID of invitee(s))

3.1.21.3 When Generated

Cancel_invitation.indication is optionally generated upon cancellation of a message invitation by some module in the message space.

3.1.21.4 Effect on Reception

The effect on reception of **Cancel_invitation.indication** by a module is undefined.

3.1.21.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the module's understanding of the functional 'connections' between itself and other modules.

3.1.22 ASSERT_SUBSCRIPTION.INDICATION

3.1.22.1 Function

The **Assert_subscription.indication** primitive shall be used to notify the module of a newly asserted subscription in the message space.

3.1.22.2 Semantics

Assert_subscription.indication shall provide parameters as follows:

```
    Assert_subscription.indication (SAP, module number of subscribing module,
                                     unit ID of subscribing module,
                                     subject ID,
                                     service mode,
                                     priority,
                                     flow label,
                                     continuum ID of publisher(s),
                                     unit ID of publisher(s),
                                     role ID of publisher(s))
```

3.1.22.3 When Generated

Assert_subscription.indication is optionally generated upon assertion of a subscription by some module in the message space.

3.1.22.4 Effect on Reception

The effect on reception of **Assert_subscription.indication** by a module is undefined.

3.1.22.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the module's understanding of the functional 'connections' between itself and other modules.

The specification of a subscription has two elements: subject number and *publisher profile*. The publisher profile specifies the continuum, unit, and role characterizing the publishers from which messages are exclusively solicited by this subscription; it implicitly defines the subscription's *publisher set*, the set of all modules in the venture conforming to the publisher profile. A subscription whose publisher set is a superset of some other subscription's publisher set *subsumes* that other subscription.

3.1.23 CANCEL_SUBSCRIPTION.INDICATION

3.1.23.1 Function

The **Cancel_subscription.indication** primitive shall be used to notify the module of a newly canceled subscription in the message space.

3.1.23.2 Semantics

Cancel_subscription.indication shall provide parameters as follows:

Cancel_subscription.indication (SAP, module number of unsubscribing module,
unit ID of unsubscribing module,
subject ID,
continuum ID of publisher(s),
unit ID of publisher(s),
role ID of publisher(s))

3.1.23.3 When Generated

Cancel_subscription.indication is optionally generated upon cancellation of a subscription by some module in the message space.

3.1.23.4 Effect on Reception

The effect on reception of **Cancel_subscription.indication** by a module is undefined.

3.1.23.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the module's understanding of the functional 'connections' between itself and other modules.

3.1.24 MODULE_IS_DEAD.INDICATION

3.1.24.1 Function

The **Module_is_dead.indication** primitive shall be used to notify the module that it has been declared dead by its registrar.

3.1.24.2 Semantics

Module_is_dead.indication shall provide parameters as follows:

Module_is_dead.indication (SAP)

3.1.24.3 When Generated

Module_is_dead.indication is generated upon reception of a you_are_dead Meta-AMS message, indicating that the registrar has determined that the module is no longer an authentic participant in the message space.

3.1.24.4 Effect on Reception

The effect on reception of **Module_is_dead.indication** by a module is undefined.

3.1.24.5 Additional Comments

None.

3.2 SERVICES REQUIRED OF THE TRANSPORT SERVICE

The service primitives and parameters required to access the services of the Transport service are:

`TRANSPORT.request`	(TSDU, Destination TS Address, Flow label)
`TRANSPORT.indication`	(TSDU, Source TS Address, Flow label)

NOTES

1 This Transport service definition applies to the Transport Services utilized by the MAMS and AMS protocols as described in 2.3.1. The services used to implement Remote AMS communication channels as described in 4.4.1 are not subject to this definition.

2 The service assumed of the underlying layer is as simple as possible to allow AMS to be as generally applicable as possible. For the purposes of this specification the service is referred to as the Transport Service and the delimited data unit it transfers is the Transport Service Data Unit (TSDU). Although the Transport service is conceptually a single service, in practice AMS will likely use several different underlying communication systems concurrently.

3 The format and contents of the TS Address parameter depend on the addressing capabilities and conventions of the underlying service. Information in the MIB must enable translation between the identifiers of AMS communicating entities and the corresponding TS addresses. The above list of mandatory Transport service primitives does not imply that the Transport service is prohibited from supporting additional services—or involving additional parameters (e.g., priority) in its support of the indicated services—for the convenience of the AMS implementation.

4 The quality of service provided by the transport service may vary, so long as it can be characterized within AMS in terms of the quality of service parameters discussed in 3.1.2.10 through 3.1.2.12 above. Flow label as specified in the AMS service request shall be presented to the Transport Service as an additional quality of service discriminator; consumption and interpretation of flow label values is a Transport Service implementation matter.

5 The Transport Service is required to detect all corrupted TSDUs and either discard them or else correct them before presenting them to AMS in **Transport.indication** primitives.

6 The Transport Service is required to deliver a **Fault.indication** primitive in the event that **Transport.request** fails.

7 Annex A contains the list of currently defined endpoint specification syntaxes.

(Blank page)

4 PROTOCOL SPECIFICATION

4.1 GENERAL

4.1.1 All Meta-AMS PDUs (or ‘MPDUs’) shall be transmitted using the AMS continuum’s primary transport service.

4.1.2 Reception of an MPDU which is determined to be inauthentic, ill-formed, or inappropriate shall cause that PDU to be discarded immediately and processed no further.

4.1.3 An ‘ill-formed’ AAMS, MAMS, or RAMS PDU shall be one whose structure and/or content do not conform to this Recommended Standard. An ‘inappropriate’ MPDU shall be one that is received at a moment when the receiving entity is in a state from which there is no prescribed transition that would be initiated by the arrival of this type of MPDU.

4.1.4 An MPDU shall be deemed inauthentic if the sender of the PDU is an AMS communicating entity for which a public key is present in the MIB and either:

– the PDU does not contain a digital signature (see 5.1.4), or

– when the encrypted string in the digital signature is decrypted using the applicable public key, the result is not identical to the concatenated text string from which that encrypted string was nominally generated.

4.1.5 Digital signatures contained in MPDUs from entities for which no public key is present in the MIB shall be ignored.

4.1.6 A procedure shall be ‘authorized’ if the acting entity is identified in the Management Information Base as an entity that is authorized to perform that procedure. Authorization can serve to prevent activity by a module that has not yet registered, because authorization is role-based and the role of an unregistered module is unknown.

4.1.7 A checksum may optionally be provided as the last 16 bits of any AAMS or MAMS PDU. The presence of a checksum in an AAMS or MAMS PDU shall be indicated by a value of ‘1’ in the checksum flag field of the PDU’s header. The checksum for an AAMS or MAMS PDU shall be computed as follows:

a) The array of octets over which the checksum is calculated shall be the non-checksum portion of the PDU, padded with a single octet of value zero if and only if the length of the non-checksum portion of the PDU is not an integral multiple of two.

b) All 16-bit words in this array of octets are summed together.

c) The computed checksum is the value of the low-order 16 bits of this sum.

4.1.8 Any received PDU that contains a checksum whose value is not equal to the checksum value computed for that PDU by the receiving communicating entity shall be discarded immediately and processed no further.

4.2 META-AMS CONFIGURATION PROCEDURES

4.2.1 CONFIGURATION SERVER INITIALIZATION

4.2.1.1 Upon commencing operations and once every N5 seconds thereafter, the configuration server shall send an *I_am_running* MPDU to every 'lower-ranking' network location at which the configuration server is authorized to operate, as indicated in the configuration server's Management Information Base.

NOTE – Since these network locations are listed in descending order of preference, a lower-ranking network location is defined as one that appears at some point in the list after the configuration server's own network location.

4.2.1.2 On reception of an *I_am_running* MPDU, the configuration server shall cease operations.

4.2.2 CONFIGURATION SERVER INTERROGATION

4.2.2.1 Interrogation of the continuum's configuration server shall be an iterative procedure which cycles repeatedly (as necessary) through all of the network locations at which the configuration server is authorized to operate, as listed (in descending order of preference) in the module's Management Information Base. If the configuration server's network location was known at some time in the past, then the first network location to which the interrogation procedure is applied shall be most recent previously noted network location; otherwise, the first network location to which the interrogation procedure is applied shall be the most preferred location listed in the MIB. The interrogation procedure shall iterate until either contact with the configuration server is deemed to have succeeded (as described below) or else AMS activity is terminated. If contact with the last (least preferred) network location in the list is deemed not to have succeeded, iteration of the configuration server interrogation procedure shall continue with the most preferred location listed in the MIB.

NOTE – Inter-iteration latency in excess of the latency implied by the definition of the configuration server interrogation procedure may be introduced by the implementation. This is an implementation matter.

4.2.2.2 Each cycle of the configuration server interrogation procedure shall comprise the following steps:

4.2.2.2.1 A procedure-specific query MPDU shall be sent to the configuration server at the indicated network location.

4.2.2.2.2 On reception of a procedure-specific query MPDU, the configuration server shall return the appropriate corresponding procedure-specific response MPDU.

4.2.2.2.3 On reception of a procedure-specific response MPDU in response to a procedure-specific query MPDU, contact with the configuration server shall be deemed to have succeeded and the network location of the originating configuration server shall be noted as the current configuration server network location.

4.2.2.2.4 If no responding procedure-specific response MPDU is received within N1 seconds, contact shall be deemed not to have succeeded.

4.2.3 REGISTRAR INITIALIZATION

4.2.3.1 Upon commencing operations and at any other time when a registrar is required to initialize communications with the configuration server, the registrar shall interrogate the configuration server. For this purpose, the applicable procedure-specific query MPDU shall be *announce_registrar* and the applicable procedure-specific response MPDU shall be either *rejection* or *registrar_noted*.

4.2.3.2 On reception of an *announce_registrar* MPDU, the configuration server shall:

4.2.3.2.1 Determine whether or not (a) the configuration server already knows the network location of the registrar for the indicated cell of the indicated message space (i.e., a duplicate or inauthentic registration) or (b) the indicated cell is unknown to the configuration server (i.e., not identified in the Management Information Base).

4.2.3.2.2 If so, return a *rejection* MPDU indicating the reason for refusing the announcement.

4.2.3.2.3 Otherwise:

a) note the network location of the registrar for this cell;

b) return a registrar_noted MPDU;

c) return one *cell_spec* MPDU for every other cell in the message space, indicating the network location of each cell's registrar—except that if the message space comprises only a single cell then the configuration server shall return a single *cell_spec* MPDU with unit number set to the new registrar's own unit number indicating 'no other cells in the message space besides you';

d) send to every other registrar in the message space a *cell_spec* MPDU indicating the network location of the new registrar;

e) commence an N3-second heartbeat cycle for heartbeat exchange with the registrar (see discussion of heartbeats below).

4.2.3.3 On reception of a *rejection* MPDU the registrar shall cease operations.

4.2.3.4 On reception of a *registrar_noted* MPDU, the registrar shall commence an N3-second heartbeat cycle for heartbeat exchange with the configuration server; see discussion of heartbeats below.

4.2.3.5 On reception of a *cell_spec* MPDU, the registrar shall note the network location of the indicated cell's registrar.

4.2.4 REGISTRAR LOCATION

4.2.4.1 When the network location of a cell's registrar is known, the registrar location procedure shall be deemed to have succeeded.

4.2.4.2 When the network location of a cell's registrar is unknown:

4.2.4.2.1 The module shall interrogate the configuration server. For this purpose, the applicable procedure-specific query MPDU shall be *registrar_query* and the applicable procedure-specific response MPDU shall be either *cell_spec* or *registrar_unknown*.

4.2.4.2.2 On reception of a *registrar_query* MPDU by the configuration server:

4.2.4.2.2.1 If the configuration server knows the network location for the registrar for the indicated cell, it shall return a *cell_spec* MPDU indicating that network location.

4.2.4.2.2.2 Otherwise, the configuration server shall return a *registrar_unknown* MPDU indicating its inability to return a *cell_spec*.

4.2.4.2.3 On reception of a *cell_spec* MPDU in response to a *registrar_query*, the network location of the registrar shall be noted and registrar location shall be deemed to have succeeded.

4.2.4.2.4 On reception of a *registrar_unknown* MPDU in response to a *registrar_query*, registrar location shall be deemed to have failed.

4.2.5 MODULE REGISTRATION

4.2.5.1 The following procedure shall be initiated upon reception of a **Register.request** primitive.

4.2.5.2 If the role ID indicates 'all roles' or is otherwise invalid, a **Fault.indication** primitive shall be delivered and the request shall be processed no further.

4.2.5.3 The network location of the registrar for the module's cell shall be determined. If the registrar location procedure is deemed to have failed, a **Fault.indication** primitive shall be delivered and the registration attempt may be either abandoned or repeated (an implementation decision).

4.2.5.4 The module shall send a *module_registration* MPDU to the registrar for the module's cell.

4.2.5.5 On reception of a *module_registration* MPDU:

4.2.5.5.1 If the number of modules in the cell's registered membership is already equal to the maximum permitted (an implementation matter), the registrar shall return a *rejection* MPDU indicating the reason for refusing the registration.

4.2.5.5.2 If the registrar has been operating for less than N5 seconds (N6 times the module heartbeat period) then it may not yet have an authoritative list of all modules in the cell's registered membership and therefore may not yet be able to assign module numbers without risk of duplication. In this case the registrar may return a *rejection* MPDU indicating the reason for refusing the registration.

4.2.5.5.3 Otherwise:

4.2.5.5.3.1 The registrar shall assign an unused module number to the module and return a *you_are_in* MPDU, indicating the module's own module number. At this point the module has become a member of the registered membership of the registrar's cell.

4.2.5.5.3.2 Optionally, the registrar may construct a set of one or more *I_am_here* MPDUs containing the MAMS state (i.e., MAMS endpoint name, all non-empty delivery vectors, and all current invitations and subscriptions) for all other modules in the message space and send these MPDUs to the newly registered module.

NOTE – This optional procedure is possible only if the registrar extracts this information from the MAMS message traffic it conveys and retains this information in local state (an implementation matter). Doing so will minimize the number of messages exchanged during module registration, at the cost of increased processing at the registrar.

4.2.5.5.3.3 The registrar shall construct either an *I_am_starting* MPDU or a *module_has_started* MPDU, containing the module's MAMS endpoint name and all non-empty delivery vectors, on behalf of the module and shall send this MPDU to every other module in its cell's registered membership and to every other registrar in the message space. If the registrar sent an *I_am_here* MPDU to the newly registered module as described above, the type of MPDU sent at this point shall be *module_has_started*; otherwise, the MPDU that is sent shall be *I_am_starting*.

4.2.5.5.3.4 The registrar shall commence an N4-second heartbeat cycle for heartbeat exchange with the new module (see discussion of heartbeats below).

4.2.5.5.4 On reception of a *rejection* MPDU a **Fault.indication** primitive shall be delivered and the registration attempt may be either abandoned or repeated (an implementation decision).

4.2.5.5.5 On reception of a *you_are_in* MPDU:

a) The module shall commence an N4-second heartbeat cycle for heartbeat exchange with the registrar (see discussion of heartbeats below).

b) An invitation for messages on the subject that is the additive inverse of the continuum number identifying the local continuum, from modules characterized by role '1' (the RAMS gateway) in the root cell of the local continuum, shall be asserted as described in 3.1.5 above. NOTE – The application data of each such message will itself be an AAMS message that was issued in a remote continuum and forwarded via RAMS procedures.

c) At this time the module may commence application-specific message subscription, invitation, transmission, and reception activity.

4.2.5.5.6 If no responsive *rejection* or *you_are_in* MPDU is received within N2 seconds, a **Fault.indication** primitive shall be delivered and the registration attempt may be either abandoned or repeated (an implementation decision).

4.2.5.5.7 On reception of an *I_am_starting* or *module_has_started* MPDU from another registrar, the registrar shall forward the MPDU without modification to every module in its cell's registered membership.

4.2.5.5.8 On a module's reception of an *I_am_starting* MPDU from a registrar:

a) The new module's MAMS endpoint name shall be noted.

b) Optionally, a **Register.indication** primitive shall be delivered.

c) For each of the new module's delivery vectors (i.e., supported service modes), the most preferred delivery point to which the receiving module is able to send messages—the 'best fit' delivery point—shall be noted. The 'best fit' delivery point is determined by examining the delivery points in the delivery vector, in the order in which they appear, until one is encountered whose transport service name is equal to that of one of the transport services by which the receiving module can transmit AAMS messages, as indicated in the receiving module's Management Information Base. If there is no 'best fit' delivery point for this service mode, then this fact shall be noted.

d) An *I_am_here* MPDU containing the receiving module's MAMS state (i.e., MAMS endpoint name, all of its non-empty delivery vectors, and all of its current invitations and subscriptions) shall be sent to the new module.

4.2.5.5.9 Alternatively, on a module's reception of a *module_has_started* MPDU from a registrar:

a) The new module's MAMS endpoint name shall be noted.

b) Optionally, a **Register.indication** primitive shall be delivered.

c) For each of the originating module's delivery vectors (i.e., supported service modes), the most preferred delivery point to which the receiving module is able to send messages—the 'best fit' delivery point—shall be noted as described above.

4.2.5.5.10 On a module's reception of a *I_am_here* MPDU, for each module whose MAMS state is contained in the MPDU:

a) The module's MAMS endpoint name shall be noted.

b) Optionally, a **Register.indication** primitive shall be delivered.

c) For each of the module's delivery vectors (i.e., supported service modes), the most preferred delivery point to which the receiving module is able to send messages—the 'best fit' delivery point—shall be noted as described above.

d) For each of the module's subscriptions:

   1) The subscription shall be noted.

   2) Optionally, an **Assert_subscription.indication** primitive shall be delivered.

e) For each of the module's invitations:

   1) The invitation shall be noted.

   2) Optionally, an **Assert_invitation.indication** primitive shall be delivered.

4.2.6 UNREGISTRATION

4.2.6.1 On reception of an **Unregister.request** primitive:

a) The network location of the registrar shall be determined.

b) An *I_am_stopping* MPDU shall be sent to the registrar.

c) The module's AMS activity shall cease immediately.

4.2.6.2 On reception of an *I_am_stopping* MPDU from a module in its cell's registered membership, the registrar shall note the termination of the module and shall forward the MPDU without modification to every other module in its cell's registered membership and to every other registrar in the message space.

4.2.6.3 On reception of an *I_am_stopping* MPDU from another registrar, the registrar shall forward the MPDU without modification to every module in its cell's registered membership.

4.2.6.4 On a module's reception of an *I_am_stopping* MPDU from a registrar:

a) The module's removal shall be noted.

b) Optionally, an **Unregister.indication** primitive shall be delivered.

c) For each of the terminated module's invitations:

   1) The invitation shall be forgotten.

2) Optionally, a **Cancel_invitation.indication** primitive shall be delivered.

d) For each of the terminated module's subscriptions:

1) The subscription shall be forgotten.

2) Optionally, a **Cancel_subscription.indication** primitive shall be delivered.

e) If the receiving module is the terminated module itself (i.e., the module's termination was imputed from a failure to deliver heartbeats to the registrar on a timely basis, as discussed below), the module's AMS activity shall cease immediately.

4.2.7 HEARTBEATS

4.2.7.1 Upon expiration of any heartbeat period, a *heartbeat* MPDU shall be sent to the relevant communicating entity and a reciprocating *heartbeat* MPDU shall be expected from that entity, subject to the following constraints.

4.2.7.1.1 At any time when the configuration is required to send a heartbeat MPDU to the registrar for a cell and the current network location of that registrar is unknown, no heartbeat MPDU shall be sent.

4.2.7.1.2 At any time when a registrar is required to send a heartbeat MPDU to the configuration server and the current network location of the configuration server is unknown, the registrar initialization procedure described in 4.2.3 above shall be performed.

4.2.7.1.3 At any time when a registrar is required to send a heartbeat MPDU to a module that has unregistered, no heartbeat MPDU shall be sent.

4.2.7.1.4 At any time when a module is required to send a heartbeat MPDU to the registrar for the module's cell and the current network location of that registrar is unknown, the reconnection procedure described in 4.2.8 below shall be performed.

4.2.7.2 Reception of a heartbeat MPDU shall have no effect on the state of the receiving entity; only non-reception of expected heartbeat MPDUs within bounded time is operationally significant. Whenever N6 successive heartbeat periods lapse after reception of an expected *heartbeat* MPDU before reception of another, an unplanned termination of the relevant communicating entity shall be imputed.

4.2.7.3 Imputed termination of the configuration server shall simply cause the current network location of the configuration server to be unknown for future communication purposes.

NOTE – Registrars will eventually be forced to re-locate the configuration server and re-register with it in order to continue transmission of heartbeat MPDUs to the configuration server.

4.2.7.4 Upon imputed termination of a registrar, the configuration server shall note that it no longer knows the network location of the registrar for this cell and shall take some action intended to cause the registrar to be restarted; the nature of the action to be taken is an implementation matter.

4.2.7.5 Upon imputed termination of a registrar, each module in the registrar's cell shall note that the current network location of that registrar is now unknown.

4.2.7.6 Upon imputed termination of a module, the registrar shall send a *you_are_dead* MPDU to the terminated module (in case it does not realize that its termination has been imputed) and shall send an *I_am_stopping* MPDU on behalf of this module to every other module in its cell and to every other registrar in the message space. This will have the effect of unregistering the module in the manner discussed above.

4.2.8 RECONNECTION

4.2.8.1 When a module is required to perform the reconnection procedure, because of inability to send a heartbeat MPDU to its registrar as described above:

a) The new network location of the registrar for the module's cell shall be determined. If the registrar location procedure is deemed to have failed, the reconnection procedure shall be terminated.

NOTE – The reconnection procedure will be performed again the next time the module is required to send a heartbeat MPDU to its registrar.

b) A *reconnect* MPDU indicating the numbers of all modules in the cell's registered membership shall be sent to the restarted registrar.

4.2.8.2 The exact procedure performed by a registrar upon reception of a *reconnect* MPDU from a module is largely an implementation matter; one way or another, however, either a *you_are_dead* MPDU or a *reconnected* MPDU shall be returned to that module and, in the latter case, an N4-second heartbeat cycle for heartbeat exchange with that module shall commence. For example, the registrar might:

a) return a *you_are_dead* MPDU if more than N5 seconds have elapsed since the registrar commenced heartbeat exchange with the current configuration server, during which time all modules in the cell should already have contacted it (this would serve to defend the registrar against spurious reconnections);

b) return a *you_are_dead* MPDU if a *reconnect* MPDU had previously been received from some other module, and the module census in that *reconnect* MPDU did not include the module that issued this new *reconnect* MPDU (this would serve to defend the registrar against inconsistent cell censuses);

c) return a *you_are_dead* MPDU if a *reconnect* MPDU had previously been received from the same module, i.e., another module identifying itself by the same module number;

d) otherwise:

   1) commence an N4-second heartbeat cycle for heartbeat exchange with this module,

   2) return a *reconnected* MPDU.

4.2.8.3 On reception of a *you_are_dead* MPDU, the receiving module shall cease AMS activity immediately and shall deliver a **Module_is_dead.indication** primitive.

4.2.8.4 On reception of a *reconnected* MPDU in response to a *reconnect* MPDU, the receiving module shall continue its N4-second heartbeat cycle for heartbeat exchange with the registrar.

4.2.9 RESYNCHRONIZATION

4.2.9.1 Resynchronization shall not be performed in any cell whose registrar's MIB states that the resynchronization interval is zero. Each registrar whose MIB states that the resynchronization interval is non-zero shall, upon expiration of that interval, send a *cell_status* MPDU to every module in its cell's registered membership and to every other registrar in the message space.

4.2.9.2 On reception of a *cell_status* MPDU from another registrar, the receiving registrar shall forward the MPDU without modification to every module in its cell's registered membership if and only if its own MIB states that the resynchronization interval is non-zero.

NOTE – That is, any given registrar's participation in all aspects of resynchronization depends on the value of its MIB's resynchronization interval and is therefore optional.

4.2.9.3 On reception of a *cell_status* MPDU at a module, the receiving module shall:

a) for each module in the referenced cell whose registration is currently noted but whose module number does not appear in the MPDU's module list, simulate reception of an *I_am_stopping* MPDU from this module as described in 4.2.6 above;

b) if the referenced cell is the module's own cell, return a *module_status* MPDU—containing only the module's own status—to that cell's registrar.

4.2.9.4 On reception of a *module_status* MPDU from a module in its own cell's registered membership:

4.2.9.4.1 The registrar may immediately forward the MPDU without modification to every other module in its cell's registered membership and to every other registrar in the message space.

4.2.9.4.2 Alternatively, the registrar may optionally concatenate the module status structure in the MPDU with one or more other module status structures in an aggregate *module_status* MPDU that has not yet been forwarded; after doing so, it may immediately forward that MPDU as described above or it may defer the forwarding of that MPDU until a future time. In the latter case, determination of the appropriate time to forward the deferred, possibly aggregated MPDU is an implementation matter, **but in no case shall this time be later than N5 seconds after the time at which the *cell_status* MPDU was transmitted**. Module status aggregation may reduce the number of messages exchanged during resynchronization, at the cost of increased processing at the registrar and somewhat increased resynchronization latency at the modules.

4.2.9.5 On reception of a *module_status* MPDU from another registrar, the receiving registrar shall forward the MPDU without modification to every module in its cell's registered membership if and only if its own MIB states that the resynchronization interval is non-zero.

NOTE – Again, any given registrar's participation in all aspects of resynchronization depends on the value of its MIB's resynchronization interval and is therefore optional.

4.2.9.6 On reception of a *module_status* MPDU at a module, the receiving module shall:

a) if the declared MAMS endpoint name and/or role of the MPDU conflict with those currently noted for the indicated module, simulate reception of an *I_am_stopping* MPDU from this module as described in 4.2.6 above;

b) if, at this point, the indicated module is unknown, note the module;

c) for each currently noted subscription for the indicated module that does not appear in the MPDU's subscription list, simulate reception of an *unsubscribe* MPDU for this subscription for this module as described in 4.2.11 below;

d) for each currently noted invitation for the indicated module that does not appear in the MPDU's invitation list, simulate reception of a *disinvite* MPDU for this invitation for this module as described in 4.2.13 below;

e) for each subscription in the MPDU's subscription list that is not currently noted by the operative module, simulate reception of a *subscribe* MPDU for this subscription for this module as described in 4.2.10 below;

f) for each invitation in the MPDU's invitation list that is not currently noted by the operative module, simulate reception of an *invite* MPDU for this invitation for this module as described in 4.2.12 below.

4.2.10 SUBSCRIPTION ASSERTION

4.2.10.1 On reception of an **Assert_subscription.request** primitive:

4.2.10.1.1 If the operative module is not an authorized receiver of messages on this subject, or if the subject ID of the request indicates 'all subjects' and the continuum ID of the request indicates either 'all continua' or any continuum other than the local continuum, then a **Fault.indication** primitive shall be delivered.

4.2.10.1.2 Otherwise:

a) the network location of the registrar shall be determined;

b) a *subscribe* MPDU shall be sent to the registrar;

c) optionally, an **Assert_subscription.indication** primitive shall be delivered, notifying the operative module of its own subscription assertion.

NOTE – A module might issue subscriptions whose domains are not disjoint sets. Selection of delivery vector, priority, and flow label when sending a message that satisfies multiple, possibly conflicting subscriptions from the same subscriber is an implementation matter.

4.2.10.2 On a registrar's reception of a *subscribe* MPDU from a module in its cell's registered membership, the registrar shall forward the MPDU without modification to every other module in its cell's registered membership and to every other registrar in the message space.

4.2.10.3 On reception of a *subscribe* MPDU from another registrar, the registrar shall forward the MPDU without modification to every module in its cell's registered membership.

4.2.10.4 On a module's reception of a *subscribe* MPDU from a registrar:

a) The subscription shall be noted. A subscription to the subject whose number is 0 shall signify a subscription to messages on all subjects.

b) Optionally, an **Assert_subscription.indication** primitive shall be delivered.

4.2.11 SUBSCRIPTION CANCELLATION

4.2.11.1 On reception of a **Cancel_subscription.request** primitive:

4.2.11.1.1 If no subscription for messages on the indicated subject is currently asserted—that is, asserted and not subsequently canceled—for the indicated continuum, unit, and role, then the cancellation shall be judged redundant and a **Fault.indication** primitive shall be delivered. The prior assertion of a subscription for messages on subject 0 ('all subjects') does **not** enable subsequent cancellation of a subscription for messages on any subject other than 0 in the manner of an 'exception' to the global subscription.

4.2.11.1.2 Otherwise:

a) the network location of the registrar shall be determined;

b) an *unsubscribe* MPDU shall be sent to the registrar;

c) optionally, a **Cancel_subscription.indication** primitive shall be delivered, notifying the operative module of its own subscription cancellation.

4.2.11.2 On reception of an *unsubscribe* MPDU from a module in its cell's registered membership, the registrar shall forward the MPDU without modification to every other module in its cell's registered membership and to every other registrar in the message space.

4.2.11.3 On reception of an *unsubscribe* MPDU from another registrar, the registrar shall forward the MPDU without modification to every module in its cell's registered membership.

4.2.11.4 On a module's reception of an *unsubscribe* MPDU from a registrar:

a) the indicated subscription shall be forgotten;

b) optionally, a **Cancel_subscription.indication** primitive shall be delivered.

4.2.12 INVITATION ASSERTION

4.2.12.1 On reception of an **Assert_invitation.request** primitive:

4.2.12.1.1 If the operative module is not an authorized receiver of messages on this subject, or if the subject ID of the request indicates 'all subjects' and the continuum ID of the request indicates either 'all continua' or any continuum other than the local continuum, a **Fault.indication** primitive shall be delivered.

4.2.12.1.2 Otherwise:

a) the network location of the registrar shall be determined;

b) an *invite* MPDU shall be sent to the registrar;

c) optionally, an **Assert_invitation.indication** primitive shall be delivered, notifying the operative module of its own invitation assertion.

NOTE – A module might issue invitations whose domains are not disjoint sets. Selection of delivery vector, priority, and flow label when sending a message that satisfies multiple, possibly conflicting invitations from the same inviter is an implementation matter.

4.2.12.2 On a registrar's reception of an *invite* MPDU from a module in its cell's registered membership, the registrar shall forward the MPDU without modification to every other module in its cell's registered membership and to every other registrar in the message space.

4.2.12.3 On reception of an *invite* MPDU from another registrar, the registrar shall forward the MPDU without modification to every module in its cell's registered membership.

4.2.12.4 On a module's reception of an *invite* MPDU from a registrar:

a) The invitation shall be noted. An invitation on the subject whose number is 0 shall signify an invitation for messages on all subjects.

b) Optionally, an **Assert_invitation.indication** primitive shall be delivered.

4.2.13 INVITATION CANCELLATION

4.2.13.1 On reception of a **Cancel_invitation.request** primitive:

4.2.13.1.1 If no invitation for messages on the indicated subject is currently asserted—that is, asserted and not subsequently canceled—for the indicated unit and role, then the cancellation shall be judged redundant and a **Fault.indication** primitive shall be delivered. The prior assertion of an invitation for messages on subject 0 ('all subjects') does **not** enable subsequent cancellation of an invitation for messages on any subject other than 0 in the manner of an 'exception' to the global invitation.

4.2.13.1.2 Otherwise:

a) the network location of the registrar shall be determined;

b) a *disinvite* MPDU shall be sent to the registrar;

c) optionally, a **Cancel_invitation.indication** primitive shall be delivered, notifying the operative module of its own invitation cancellation.

4.2.13.2 On reception of a *disinvite* MPDU from a module in its cell's registered membership, the registrar shall forward the MPDU without modification to every other module in its cell and to every other registrar in the message space.

4.2.13.3 On reception of a *disinvite* MPDU from another registrar, the registrar shall forward the MPDU without modification to every module in its cell's registered membership.

4.2.13.4 On a module's reception of a *disinvite* MPDU from a registrar:

a) the indicated invitation shall be forgotten;

b) optionally, a **Cancel_invitation.indication** primitive shall be delivered.

4.3 APPLICATION AMS COMMUNICATION PROCEDURES

4.3.1 MESSAGE TRANSMISSION

4.3.1.1 Whenever an AAMS message is to be transmitted from one module to another, the transport service to use for this transmission shall be selected as follows:

4.3.1.1.1 If the delivery vector (i.e., service mode) specified for this message transmission activity by the destination module is one for which no 'best fit' delivery point could be determined, then transmission of the message is impossible; a **Fault.indication** primitive shall be delivered.

4.3.1.1.2 Otherwise, the 'best fit' delivery point shall be used as the destination endpoint for the message and the indicated transport service shall be used to accomplish the transmission. If the transmission is determined to have failed for any reason, a **Fault.indication** primitive shall be delivered.

NOTE – A module might issue subscriptions and invitations whose domains are not disjoint sets. Selection of delivery vector, priority, and flow label when sending a message that satisfies multiple, possibly conflicting subscriptions and/or invitations from the same module is an implementation matter.

4.3.1.2 Authorization to send or receive messages on a given subject shall be determined by reference to security information in the MIB where present.

4.3.1.3 Direction to marshal and un-marshal messages on a given subject, and the manner in which such marshaling and un-marshaling is to be accomplished, shall be determined by reference to subject catalog information in the MIB where present.

4.3.1.4 Direction to encrypt and decrypt messages on a given subject, and the manner in which such encryption and decryption is to be accomplished, shall be determined by reference to security information in the MIB where present.

4.3.2 PUBLISH

4.3.2.1 On reception of a **Publish.request** primitive:

4.3.2.1.1 If the module is not an authorized sender of messages on this subject, a **Fault.indication** primitive shall be delivered.

4.3.2.1.2 Otherwise, an AAMS message shall be constructed using the provided parameters, with application data marshaled and/or encrypted as necessary, and one copy of this message shall be transmitted to every module in the message space that is an authorized receiver of messages on the indicated subject for which the operative module has noted at least one currently asserted subscription (a) that is for messages on this subject or on all subjects and (b) whose domain includes the operative module.

4.3.3 REMOTE AMS PRIVATE MESSAGE TRANSMISSION

4.3.3.1 Whenever a service primitive is received that requests that a message be sent privately (as described in subsections 4.3.4, 4.3.5, 4.3.6, and 4.3.7 below) to a module or modules residing in some continuum other than the local continuum:

4.3.3.1.1 An AAMS message shall be constructed from the parameters of the service primitive, in the same way that such a message would be constructed if the destination module resided in the local continuum (i.e., with application data marshaled and/or encrypted as necessary). NOTE – The newly constructed AAMS message is not transmitted at this time.

4.3.3.1.2 An *envelope* data structure shall be constructed whose content is the newly constructed AAMS message (see 5.3.2 below). The continuum number, unit number, and destination ID number of the envelope shall be those identifying the destination as specified by the service primitive; the source ID number of the envelope shall identify the source module's own role, and the subject number of the envelope shall identify the subject specified in the original service primitive. The control code of the envelope shall be '5' in the event that the original service primitive specified destination module number (send, query, or reply); it shall be '6' in the event that the original service primitive specified destination role number (announce).

4.3.3.1.3 A message shall be published whose application data is the envelope constructed above. If the destination continuum number is 0, signifying 'all continua', then the subject of the published message shall be the additive inverse of the number of the local continuum. Otherwise, the subject of the published message shall be the additive inverse of the destination continuum number.

4.3.4 SEND

4.3.4.1 On reception of a **Send.request** primitive:

4.3.4.1.1 If the module is not an authorized sender of messages on this subject or the indicated destination module is not an authorized receiver of messages on this subject, a **Fault.indication** primitive shall be delivered.

4.3.4.1.2 Otherwise, if the indicated destination continuum is not the local continuum, then the Remote AMS private message transmission procedure described in 4.3.3 above shall be performed.

4.3.4.1.3 Otherwise, if the operative module has noted no current invitation from the indicated destination module (a) that is for messages on the indicated subject or on all subjects and (b) whose domain includes the operative module, a **Fault.indication** primitive shall be delivered.

4.3.4.1.4 Otherwise, an AAMS message shall be constructed using the provided parameters, with application data marshaled and/or encrypted as necessary, and the message shall be transmitted to the indicated destination module.

4.3.5 QUERY

4.3.5.1 On reception of a **Query.request** primitive:

4.3.5.1.1 If the module is not an authorized sender of messages on this subject or the indicated destination module is not an authorized receiver of messages on this subject, or if context number is 0 or is equal to the context associated with some other pending query for this module, or if the indicated destination continuum is the local continuum and the operative module has noted no current invitation from the indicated destination module (a) that is for messages on the indicated subject and (b) whose domain includes the operative module, a **Fault.indication** primitive shall be delivered.

4.3.5.1.2 Otherwise:

4.3.5.1.2.1 If the indicated destination continuum is not the local continuum, then the Remote AMS private message transmission procedure described in 4.3.3 above shall be performed, transmitting a Query message.

4.3.5.1.2.2 Otherwise, if the operative module has noted no current invitation from the indicated destination module (a) that is for messages on the indicated subject or on all subjects and (b) whose domain includes the operative module, a **Fault.indication** primitive shall be delivered.

4.3.5.1.2.3 Otherwise an AMS Query message shall be constructed using the provided parameters, with application data marshaled and/or encrypted as necessary, and the message shall be transmitted to the indicated destination module.

4.3.5.1.2.4 If a **non-zero** term was specified in the **Query.request** primitive:

4.3.5.1.2.4.1 Handling of inbound AAMS messages by the operative module **may** be suspended pending arrival of a reply message—i.e., a message of type Reply whose context number is the same as that of the query message—in response to this query message.

4.3.5.1.2.4.2 If the term specified in the **Query.request** primitive was greater than zero:

a) An implementation-defined method shall be initiated to determine if the specified interval has passed before the arrival of a reply message in response to this query message (e.g., a query countdown timer for the indicated interval shall be started).

b) On passing of the specified interval without the arrival of a reply message in response to this query message:

   1) A **Fault.indication** primitive shall be delivered indicating the timeout.

   2) Handling of inbound AAMS messages by the operative module**, if suspended,** shall be resumed.

4.3.6 REPLY

4.3.6.1 On reception of a **Reply.request** primitive:

4.3.6.1.1 If the module is not an authorized sender of messages on this subject or the indicated destination module is not an authorized receiver of messages on this subject, or if the value of the context parameter is zero, a **Fault.indication** primitive shall be delivered.

4.3.6.1.2 Otherwise, if the indicated destination continuum is not the local continuum, then the Remote AMS private message transmission procedure described in 4.3.3 above shall be performed, transmitting a Reply message.

4.3.6.1.3 Otherwise, if the operative module has noted no current invitation from the indicated destination module (a) that is for messages on the indicated subject or on all subjects and (b) whose domain includes the operative module, a **Fault.indication** primitive shall be delivered.

4.3.6.1.4 Otherwise, an AMS Reply message shall be constructed using the provided parameters, with application data marshaled and/or encrypted as necessary, and the message shall be transmitted to the indicated destination module.

4.3.7 ANNOUNCE

4.3.7.1 On reception of an **Announce.request** primitive:

4.3.7.1.1 If the module is not an authorized sender of messages on this subject, a **Fault.indication** primitive shall be delivered.

4.3.7.1.2 Otherwise:

4.3.7.1.2.1 If the indicated continuum is either the local continuum or 'all continua' then:

a) An AAMS message shall be constructed using the provided parameters, with application data marshaled and/or encrypted as necessary.

b) One copy of this message shall be transmitted to every module in the message space that is in the domain of the service request, that is an authorized receiver of messages on the indicated subject, and that is a module for which the operative module has noted at least one current invitation (a) that is for messages on the indicated subject or on all subjects and (b) whose domain includes the operative module.

4.3.7.1.2.2 If the indicated continuum is either 'all continua' or any continuum other than the local continuum, then the Remote AMS private message transmission procedure described in 4.3.3 above shall also be performed.

4.3.8 RECEIVE

4.3.8.1 On reception of an AAMS message:

4.3.8.1.1 If the indicated source module is not an authorized sender of messages on this subject, a **Fault.indication** primitive shall be delivered. The message shall be discarded without further processing.

4.3.8.1.2 If and only if the sender of the message is a RAMS gateway (that is, a module whose role number is '1'), indicating that the application data of the message is itself an AAMS message, then that encapsulated AAMS message shall be the message to which the rest of this subsection pertains.

NOTE – In effect, the 'outer' header of the message is stripped off and discarded before processing continues.

4.3.8.1.3 The message's application data shall be decrypted and/or unmarshaled as necessary.

4.3.8.1.4 If the message is of type Reply:

a) If handling of inbound AAMS messages by the operative module is suspended because of a query request, and the received message's context number is equal to the context number of the query, then handling of inbound AAMS messages by the operative module shall be resumed.

b) A **Reply.indication** primitive shall be delivered.

4.3.8.1.5 If the message is of type Query, a **Query.indication** primitive shall be delivered.

4.3.8.1.6 Otherwise, a **Message.indication** primitive shall be delivered.

4.4 REMOTE AMS GATEWAY PROCEDURES

4.4.1 OVERVIEW

As noted earlier, RAMS procedures are executed by special-purpose application modules called *RAMS gateways*.

No single message space may contain more than one RAMS gateway. RAMS gateway behavior is undefined when two or more RAMS gateways are registered within any single message space.

The protocol data units used to effect interoperation among RAMS gateways are termed *RAMS PDUs* (*RPDUs*). RPDUs shall be exchanged among RAMS gateways by means of RAMS communication channels (or, in this document, simply *channels*). A channel is the configuration of the communications hardware and software at two RAMS gateways such that RPDUs may be reliably conveyed from one gateway to the other without being relayed by any other RAMS gateway.

The set of RAMS gateways for all of the message spaces for some venture and no message space of any other venture is termed the *RAMS network* for that venture. Each member of a RAMS network must have a channel to at least one other member of the set, and no member of a RAMS network may have channels to any RAMS gateways that are not members of the set. Every member of the RAMS network to which a given RAMS gateway has a channel is termed a *neighbor* of that gateway.

Each RAMS gateway has interfaces to two communication environments: its RAMS network and the AMS message space it serves. These interfaces enable the RAMS gateway to receive AAMS messages produced by the modules in its message space and forward them to modules in other message spaces of the same venture—directly or indirectly—by encapsulating those messages in RPDUs and sending the RPDUs to its neighbors.

The RAMS network for a given venture must be configured either as a mesh (all RAMS gateways in the network are neighbors of one another) or as a tree (some pairs of RAMS gateways can only intercommunicate through one or more other forwarding gateways, but there is always exactly one 'route' from any gateway in the venture to any other).

As explained below, configuring a RAMS network as a mesh assures that no RPDU will ever traverse more than one channel, but (absent an underlying multicast system) the transmission burden placed on each gateway in the network increases as the number of interconnected message spaces increases. Configuring a RAMS network as a tree enables the transmission burden placed on each gateway in the network to be kept small no matter how many message spaces are included in the venture, so the venture can scale up indefinitely; however, introducing the possibility that messages may traverse multiple channels increases the venture's maximum possible message delivery latency.

In concept, the state of a RAMS gateway is a set of *petitions*. A petition for a given subscription specification (see 3.1.22.5 above) is a summary of the inter-continuum interest, as known to this RAMS gateway, in all AAMS messages satisfying that subscription specification. A petition 'for' a given subject is a petition for a subscription specification that cites that subject.

Each petition has three components:

– *SGS* (source gateway set), a list of all neighbors to which the local gateway has asserted (and not yet canceled) this petition;

– *DGS* (destination gateway set), a list of all neighbors that have asserted (and not yet canceled) this petition;

– *DMS* (destination module set), a list of all AMS modules in the local continuum that have asserted (and not yet canceled) subscriptions that (a) are characterized by this petition's subscription specification and (b) cite in their domains either 'all continua' or some continuum other than the local continuum.

A petition is considered to be in *asserted* state while (and only while) its SGS is non-empty. If the RAMS network is configured as a mesh, the petition is *assertable* while (and only while) its DMS is non-empty; if the RAMS network is configured as a tree, the petition is *assertable* while (and only while) either its DGS or its DMS is non-empty.

The *pseudo-subject* for a continuum is the AMS message subject whose number is the additive inverse of the number of that continuum.

4.4.2 DECLARATION AND RETRACTION

4.4.2.1 Initial Declaration

4.4.2.1.1 Upon instantiation, the RAMS gateway for a message space identified within its continuum by a given application name and authority name shall obtain from its Management Information Base (MIB) the RAMS network locations of all currently known neighbors.

4.4.2.1.2 The new RAMS gateway shall subscribe to messages on the pseudo-subject for the local continuum, from all modules in all message spaces of the venture.

NOTE – Messages published locally on this subject will contain announcements destined for modules in all continua.

4.4.2.1.3 The RAMS gateway shall add itself to the DMS of the *declaration petition*. The declaration petition is a petition for the subscription specification whose subject is the pseudo-subject for the gateway's local continuum and whose domain is all modules in all continua.

4.4.2.1.4 Optionally, the RAMS gateway may *retract* itself from each of its currently known neighbors. A RAMS gateway shall retract itself from a neighbor by sending that neighbor a cancellation (see 5.3.3) of the declaration petition and removing that neighbor from the SGS of that petition. This proactive retraction may enable neighbors that have previously declared themselves (as described below) to re-declare themselves to the newly instantiated RAMS gateway upon receiving its declaration.

4.4.2.1.5 The RAMS gateway shall *declare* itself to each of its currently known neighbors. A RAMS gateway shall declare itself to a neighbor by sending that neighbor an assertion (see 5.3.3) of the declaration petition and adding that neighbor to the SGS of that petition.

4.4.2.2 Subsequent Declaration Activity

4.4.2.2.1 Upon insertion of a new neighbor into the MIB, the RAMS gateway shall declare itself to that neighbor.

4.4.2.2.2 Upon removal of a neighbor from the MIB, the RAMS gateway shall *retract* itself from that neighbor.

4.4.2.2.3 Upon termination, the RAMS gateway shall remove itself from the DMS of the declaration petition and retract itself from each of its neighbors. Since a RAMS gateway is an AMS module, the 'termination' of a RAMS gateway entails the termination of that module, conforming to 4.2.6.

4.4.2.2.4 Under no conditions other than those detailed in 4.4.2 shall a RAMS gateway retract itself from any neighbor.

4.4.2.3 Derived Definitions

4.4.2.3.1 A *declared* neighbor is a neighbor whose declaration has been asserted and not yet canceled.

4.4.2.3.2 The *conduit* for a continuum is the sole member of the DGS of the petition for the pseudo-subject for that continuum.

NOTE – I.e., it is the declared neighbor to which RAMS PDUs destined (ultimately) for that continuum are sent.

4.4.3 PETITION ASSERTION AND CANCELLATION

4.4.3.1 Computing the Assertion Set

The Assertion Set (AS) for a petition shall be computed as follows:

a) The AS is initially the empty set.

b) If the continuum cited in the domain of the petition's subscription specification is 'all continua', all declared neighbors shall be added to the AS. Otherwise only the conduit for the continuum cited in the domain shall be added to the AS.

c) All members of the SGS of this petition shall be removed from the AS.

NOTE – These gateways are already forwarding all messages that satisfy the subscription specification of the petition; there's no need to assert to them again.

d) If the petition's DMS is empty and its DGS has exactly one member, the sole member of the DGS shall be removed from the AS.

NOTE – If the only destination for these messages is this single gateway, then the gateway does not need to send these messages (only to be sent back again).

4.4.3.2 Asserting a Petition

When a petition is to be asserted:

a) The AS for this petition shall be computed as described above.

b) An assertion of the petition shall be sent to all members of the computed AS.

NOTE – The computed AS may be the empty set, in which case no RPDUs are sent.

c) The SGS of the petition shall be changed to the union of itself and the computed AS.

4.4.3.3 Computing the Cancellation Set

The Cancellation Set (CS) for a petition shall be computed as follows:

a) If the petition's DGS is empty, then the CS shall contain all members of the petition's SGS.

b) If, instead, the petition's DGS contains only a single neighbor and that neighbor is also a member of the petition's SGS, then the CS shall contain only the sole member of the DGS.

NOTE – Again, if the only destination for these messages is this single gateway, then the gateway does not need to send these messages (only to be sent back again).

c) Otherwise, the CS shall be the empty set.

4.4.3.4 Canceling a Petition

When a petition is to be cancelled:

a) The CS for this petition shall be computed as described above.

b) A cancellation of the petition shall be sent to all members of the computed CS.

c) All members of the computed CS shall be removed from the SGS of the petition.

4.4.4 ASSERTION REPORTING

Upon delivery to the RAMS gateway of an **Assert_subscription.indication** primitive that is **not** the result of one of the RAMS gateway's own subscription assertions, where the continuum cited in the domain of the subscription specification is not the local continuum:

a) The subscribing module shall be added to the DMS of the petition for this subscription specification.

b) The petition shall be asserted as described above.

NOTE – The petition is known to be assertable at this time, regardless of the configuration of the RAMS network, because its DMS is now non-empty.

4.4.5 CANCELLATION REPORTING

Upon delivery of a **Cancel_subscription.indication** primitive to the RAMS gateway that is **not** the result of one of the RAMS gateway's own subscription cancellations, where the continuum cited in the domain of the subscription specification is not the local continuum:

a) The canceling module shall be removed from the DMS of the petition for this subscription specification.

b) If the petition is no longer assertable then the petition shall be canceled as described above.

4.4.6 ASSERTION REPLICATION

4.4.6.1 Upon reception of a petition assertion RPDU from a neighbor, the following procedure shall be performed:

4.4.6.2 If the neighbor is already a member of the DGS of the asserted petition, then a prior cancellation by this neighbor of this petition—which was issued but not received—shall be assumed. The cancellation replication procedure defined in 4.4.7 below shall be performed, exactly as if this inferred prior petition cancellation had finally been received, before any further steps in the assertion replication procedure are performed.

4.4.6.3 The neighbor shall be added to the DGS of this petition.

4.4.6.4 If the DGS of this petition has one member (i.e., it was formerly empty but is now non-empty) and the domain continuum number cited in the petition's subscription specification is either the local continuum number or 0 (indicating 'all continua'), the RAMS gateway shall submit a Subscribe request whose parameters are obtained from the subscription specification for this petition.

4.4.6.5 If the subject cited in the subscription specification for this petition is the pseudo-subject for some continuum—i.e., the neighbor is declaring itself to be the conduit for that continuum (possibly its own local continuum)—then all relevant petition relationships with this neighbor must be established. For each assertable petition:

a) The AS for this assertable petition shall be computed as described above.

b) If the neighbor is a member of the computed AS, then an assertion of this assertable petition shall be sent to the neighbor and the neighbor shall be added to the SGS of this assertable petition.

4.4.6.6 Finally, if the petition asserted by the neighbor is itself assertable, then the petition shall be asserted as described above.

NOTE – This has the effect of propagating the petition assertion throughout the RAMS network if and only if the RAMS network is configured as a tree.

4.4.7 CANCELLATION REPLICATION

4.4.7.1 Upon reception of a petition cancellation RPDU from a neighbor, the following procedure shall be performed:

4.4.7.2 The neighbor shall be removed from the DGS of this petition.

4.4.7.3 If the DGS of this petition has zero members (i.e., it was formerly non-empty but is now empty) and the domain continuum number cited in the petition's subscription specification is either the local continuum number or 0 (indicating 'all continua'), the RAMS gateway shall submit an Unsubscribe request whose parameters are obtained from the subscription specification for this petition.

4.4.7.4 If the subject cited in the subscription specification for this petition is the pseudo-subject for the neighbor's local continuum—i.e., the neighbor is retracting itself—then the canceling neighbor shall be removed from the DGS and SGS of all petitions.

4.4.7.5 Finally, every petition that is no longer assertable shall be canceled as described above.

NOTE – The effect of these cancellations will be to cause the local gateway to cease being the conduit to the canceling neighbor and to all continua for which the canceling neighbor was the local gateway's conduit.

4.4.8 FORWARDING OF PRIVATE AND ANNOUNCED MESSAGES VIA RAMS

4.4.8.1 On reception of a **Message.indication** primitive (indicating reception of an AAMS message from a module in the local message space) whose subject is the pseudo-subject for some continuum, the RAMS gateway shall proceed as follows:

4.4.8.2 The application data of the primitive must be an envelope data structure with control code indicating either 'send on reception' or 'announce on reception', whose content is an AAMS message that is intended for private delivery or announcement rather than re-publication (see 5.3.3 below).

4.4.8.3 If the source module of the received message is not an authorized sender of messages on the subject of the envelope data structure, then the primitive shall be discarded.

4.4.8.4 Otherwise:

4.4.8.4.1 If the continuum number in the envelope header is 0, indicating announcement to all continua, then (a) if the control code in the envelope indicates 'send on reception' then the message shall be discarded, otherwise (b) the RAMS gateway shall forward the envelope data structure—an RPDU—to all declared neighbors.

4.4.8.4.2 Otherwise the RAMS gateway shall forward the envelope data structure—an RPDU—to the conduit for the indicated continuum.

4.4.9 FORWARDING OF PUBLISHED MESSAGES VIA RAMS

4.4.9.1 On reception of a **Message.indication** primitive (indicating reception of an AAMS message from a module in the local message space) whose subject is greater than zero (i.e., a message on some application-defined subject), the RAMS gateway shall proceed as follows:

4.4.9.2 An AAMS message shall be constructed from the parameters of the primitive, in the same way that an AAMS message would be constructed from those parameters if the message's destination were a module in the local continuum (i.e., with application data marshaled and/or encrypted as necessary). NOTE – The newly constructed AAMS message is not transmitted at this time.

4.4.9.3 For each neighbor that is a member of the DGS of at least one petition whose subscription specification is satisfied by the parameters of the primitive (that is, whose continuum number is either 0 or the source continuum number, whose unit number identifies a unit that is a superset of the source unit number, and whose role number is either 0 or the role number for the role in which the module identified by the source module number is registered):

4.4.9.3.1 The RAMS gateway shall construct an envelope data structure with control code '4' ('publish on reception') whose content is the newly constructed AAMS message; see 5.3.2 below.

4.4.9.3.2 The RAMS gateway shall send the newly constructed envelope data structure—an RPDU—to that neighbor.

4.4.10 FORWARDING FROM NEIGHBORS

4.4.10.1 Upon reception of an RPDU with control code greater than '3' from a neighbor, the RAMS gateway shall proceed as follows:

4.4.10.2 If the control code of the RPDU is '4' ('publish on reception'), then:

4.4.10.2.1 The content of the RPDU must be an AAMS message replicating a message that was published in a remote continuum.

4.4.10.2.2 If the RAMS network is configured as a tree, then for each neighbor—other than the one from which the RPDU was received—that is a member of the DGS of at least one petition whose subscription specification is satisfied by the headers of this encapsulated AAMS message (source continuum number and unit number) and the RPDU (source ID number, indicating the source role number), the RAMS gateway shall forward the received RPDU to that neighbor.

4.4.10.2.3 For each module that is a member of the DMS of at least one petition whose subscription specification is satisfied by the headers of this encapsulated AAMS message (source continuum number and unit number) and the RPDU (source ID number, indicating the source role number), the RAMS gateway shall submit a Send request causing the encapsulated AAMS message to be sent as the application data of a message destined for this module; the subject number of this message shall be the additive inverse of the continuum number identifying the local continuum.

NOTE – The encapsulated AAMS message cannot simply be locally re-published by the RAMS gateway: the RAMS gateway itself might not be in the domain of one or more of the subscriptions asserted by members of the applicable destination set, and in any event this re-publication would conceal from the receiving modules the identity of the original publisher. Using the pseudo-subject for the local continuum as the subject of the private transmission assures that the message can be successfully transmitted to all indicated recipients (see 4.2.5 above, reception of the *you_are_in* MPDU).

4.4.10.3 If the control code of the RPDU is '5' ('send on reception'), then the intent is private message delivery rather than re-publication.

4.4.10.3.1 If the continuum number in the envelope header is not equal to the local continuum number (possible only when the RAMS network is configured as a tree), then the RAMS gateway shall forward the received RPDU to the conduit for the indicated continuum.

4.4.10.3.2 Otherwise, if the RAMS gateway has noted no current invitation from the destination module identified by the RPDU header (a) that is for messages on the subject of the encapsulated AAMS message that is the content of the RPDU or on all subjects and (b) whose domain includes the module identified as the source of the encapsulated AAMS message, then the RPDU shall simply be discarded.

4.4.10.3.3 Otherwise, the RAMS gateway shall submit a Send request; the destination unit and module parameters of the request shall be as indicated by the header of the RPDU, the destination continuum shall be the local continuum, the subject shall be the additive inverse of the continuum number identifying the local continuum, and the application data to be sent shall be the encapsulated AAMS message that is the content of the RPDU.

4.4.10.4 If the control code of the RPDU is '6' ('announce on reception'), then the intent is message announcement rather than re-publication.

4.4.10.4.1 If the RAMS network is configured as a tree and the continuum number in the envelope header is 0 (indicating 'all continua'), then the RAMS gateway shall forward the received RPDU to all of its declared neighbors except the one from which the RPDU was received.

4.4.10.4.2 If the continuum number in the envelope header is not 0 and is also not the number of the local continuum (possible only when the RAMS network is configured as a tree), then the RAMS gateway shall forward the received RPDU to the conduit for the indicated continuum. Otherwise (i.e., the continuum number in the envelope header is either 0 or the number of the local continuum), for each local module in the domain of the announcement (as indicated by the destination unit and role of the header of the RPDU) from which the RAMS gateway has noted a current invitation (a) that is for messages on the subject of the encapsulated AAMS message that is the content of the RPDU or on all subjects and (b) whose domain includes the module identified as the source of the encapsulated AAMS message (that is, whose continuum number is either 0 or the source continuum number of the encapsulated AAMS message, whose unit number identifies a unit that is a superset of the source unit number of the encapsulated AAMS message, and whose source role number is either 0 or the role number for the role in which the module identified by the encapsulated AAMS message's source module number is registered), the RAMS gateway shall submit a Send request. The destination unit and module parameters of this request shall be those which identify this local module, the destination continuum shall be the local continuum, the subject shall be the additive inverse of the continuum number identifying the local continuum, and the application data to be announced shall be the encapsulated AAMS message that is the content of the RPDU.

5 PROTOCOL DATA UNITS

5.1 META-AMS PROTOCOL DATA UNITS

5.1.1 META-AMS PROTOCOL DATA UNIT FORMAT

5.1.1.1 The protocol data units used to effect configuration and discovery procedures are termed *Meta-AMS (MAMS)* PDUs, or *MPDUs*. Each MPDU shall consist of a header in fixed format followed by a variable-length digital signature and/or variable-length supplementary data, possibly followed by a 16-bit checksum; the length of the digital signature in an MPDU may be zero in the event that AMS is not configured for MAMS traffic security, and otherwise may be up to 255 octets; the length of the supplementary data in an MPDU shall vary from zero to 4095 octets.

5.1.1.2 The MPDU header shall consist of the fields shown in table 5-1. The MPDU header fields shall be transmitted in the order of presentation in table 5-1. The 'sender' identified in an MPDU shall always be the MPDU's original source entity; 'sender' identification fields are not altered when a registrar forwards an MPDU.

Table 5-1: MAMS PDU Header Fields

Field	Length (bits)	Values	Comment
Version number	2	'00' for this version of MAMS.	
Checksum flag	1		Set to '1' if the digital signature and supplementary data (where present) of this message are followed by a 16-bit checksum; otherwise 'zero'.
MPDU type	5	See table 5-2 below.	
Sender's venture number	8		Insert 0 here if sender is configuration service.
Sender's unit number	16		Insert 0 here if sender is configuration service.
Sender's role number	8		Insert 0 here if sender is not a module.
Length of digital signature	8		
Length of supplementary data	16	0 through 4095.	
Reference	32	See 5-2 below.	Semantics vary by MPDU type.
Time tag	variable	Current time in CCSDS Unsegmented Time Code (CUC) format with preamble (P-field)	See CCSDS Recommended Standard 301.0-B-3 'Time Code Formats' (reference [3]).

5.1.2 META-AMS PROTOCOL DATA UNIT TYPES

The MAMS protocol data unit type shall indicate the nature of the MPDU as noted in table 5-2.

Table 5-2: MAMS PDU Types

MPDU Type (decimal)	Function
00	(reserved)
01	heartbeat
02	rejection
03	you_are_dead
04	registrar_noted
05	registrar_unknown
06	reconnected
07	announce_registrar
08	invite
09	disinvite
10	cell_spec
11	(reserved)
12	(reserved)
13	(reserved)
14	(reserved)
15	(reserved)
16	(reserved)
17	(reserved)
18	registrar_query
19	module_registration
20	you_are_in
21	I_am_starting
22	I_am_here
23	(reserved)
24	subscribe
25	unsubscribe
26	I_am_stopping
27	reconnect
28	cell_status
29	module_has_started
30	I_am_running
31	module_status

5.1.3 META-AMS REFERENCE VALUES

5.1.3.1 Query Number. A query number sequentially assigned by AMS for the purposes of a synchronous MAMS message exchange. Query numbers are 32-bit integers. Each AMS communicating entity has its own distinct series of query numbers.

5.1.3.2 Echo. The reference number of the received MPDU that caused this MPDU to be sent.

5.1.3.3 Heartbeat Source. If heartbeat is from a module, then the module's number; otherwise 0 (i.e., the source of the heartbeat is a registrar or configuration server).

5.1.3.4 Module ID. A numeric value identifying the module to which the MPDU pertains: the sum of module number plus (256 * unit number) plus (16777216 * role number). Module IDs are 32-bit integers.

5.1.4 META-AMS DIGITAL SIGNATURES

5.1.4.1 General

A digital signature may be included in a MAMS message to assure the receiver of the message that the sender is authentic. The nature of the digital signature shall depend on the source and destination of the message. Each digital signature shall be generated by:

a) selecting four ASCII characters—the 'nonce' ('number used once') for the digital signature—at random;

b) encrypting in the private key of the sender the concatenation of the nonce and some well-known string;

c) concatenating the nonce, in clear text, with the encrypted string to form the digital signature.

5.1.4.2 Module Signature

For any MAMS message sent by an AMS module, the applicable digital signature shall be generated from the module's role name using the private key of the role.

5.1.4.3 Registrar Signature

For any MAMS message sent by a registrar, the applicable digital signature shall be generated from the message space's application name using the private key of the application.

5.1.4.4 Configuration Server Signature

For any MAMS message sent by a configuration server, the digital signature shall be generated from the continuum's name using the private key of the continuum.

5.1.5 META-AMS SUPPLEMENTARY DATA VALUES

5.1.5.1 General

The term *string* is used here to signify an array of text characters formed by the concatenation of one or more lexical tokens and/or other strings delimited by single spaces. All string text is in ASCII representation. The last character of the last text character in a supplementary data value is immediately followed by a NULL character, terminating the string.

All binary integer values are in network byte order, as implied by the CCSDS standard bit numbering conventions described in 1.3.1.

5.1.5.2 MAMS Endpoint Name

A MAMS endpoint name is a string comprising the name, within the namespace identified by the primary transport service name, of some endpoint at which MAMS messages may be received.

5.1.5.3 Cell Descriptor

A cell descriptor is the concatenation of unit number (a sixteen-bit integer) and the MAMS endpoint name of the cell's registrar.

5.1.5.4 Module List

A module list is the concatenation of an eight-bit integer indicating the number of modules in the list, followed by that number of eight-bit integer module numbers.

5.1.5.5 Assigned Module Number

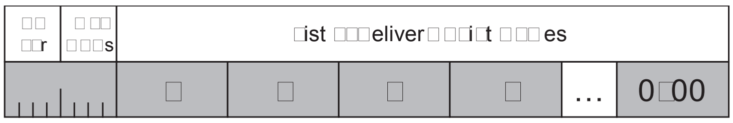
Assigned module number is an eight-bit integer containing the module number newly assigned to the recipient module.

5.1.5.6 Delivery Point Name

Delivery point name is a string constructed by concatenating transport service name, an equals (=) symbol, and the name (within the namespace identified by that transport service name) of some endpoint at which the module can receive AAMS PDUs, in that order. No transport service name shall comprise more than 15 characters. No transport service endpoint name shall comprise more than 63 characters.

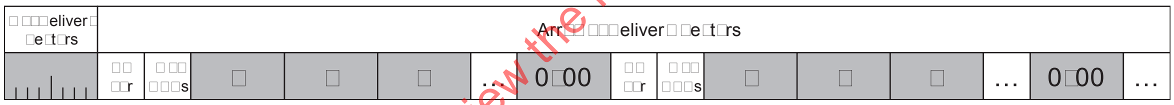
5.1.5.7 Delivery Vector

A delivery vector is the concatenation of a four-bit integer identifying the delivery vector (the delivery vector number) and a four-bit integer indicating the number of delivery point names in the vector, followed by that number of delivery point names; the last delivery point name in the vector is delimited by a NULL character, while all preceding delivery point names are delimited by commas. These delivery point names identify a set of endpoints at which AAMS messages can be delivered in the service mode that is common to all the transport services cited in the vector.



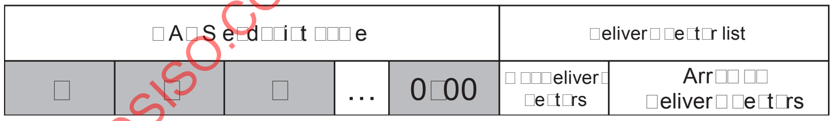
5.1.5.8 Delivery Vector List

A delivery vector list is the concatenation of an eight-bit integer indicating the number of delivery vectors in the list, followed by that number of delivery vectors. The list identifies all delivery vectors established for the module.



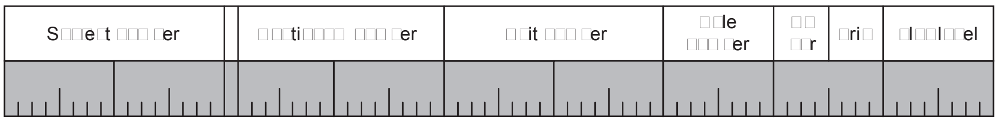
5.1.5.9 Contact Summary

A contact summary is the concatenation of a module's NULL-terminated MAMS endpoint name and its delivery vector list.



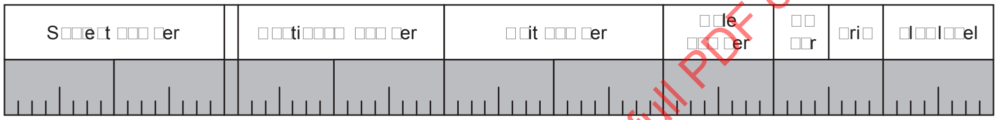
5.1.5.10 Subscription Assertion Structure

A subscription assertion structure is the concatenation of a 16-bit integer subject number; a single reserved bit, for alignment, followed by a 15-bit integer identifying the continuum containing all the modules to which the asserted subscription pertains; a 16-bit integer identifying the unit containing all the modules to which the asserted subscription pertains; an 8-bit integer identifying the role characterizing all the modules to which the asserted subscription pertains; the 4-bit delivery vector number of the delivery vector to use in transmitting messages on the subject; a 4-bit integer indicating the priority at which transmission of the messages is requested; and an 8-bit flow identifier integer with which it is requested that the messages be labeled.



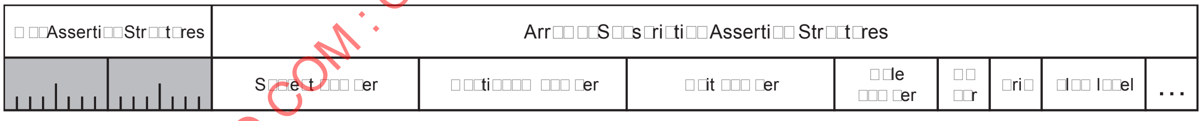
5.1.5.11 Invitation Assertion Structure

An invitation assertion structure is the concatenation of a 16-bit integer subject number; a single reserved bit for alignment, followed by a 15-bit integer identifying the continuum containing all the modules to which the asserted invitation pertains; a 16-bit integer identifying the unit containing all the modules to which the asserted invitation pertains; an 8-bit integer identifying the role characterizing all the modules to which the asserted invitation pertains; the 4-bit delivery vector number of the delivery vector to use in transmitting messages on the subject; a 4-bit integer indicating the priority at which transmission of the messages is requested; and an 8-bit flow identifier integer with which it is requested that the messages be labeled.



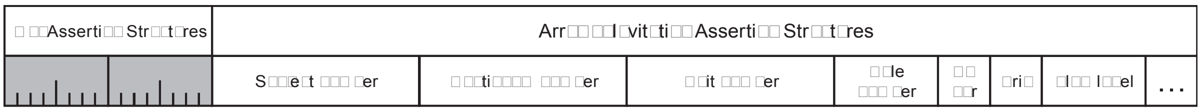
5.1.5.12 Subscription List

A subscription list is the concatenation of a 16-bit integer indicating the number of subscription assertion structures in the list, followed by that number of subscription assertion structures. The list identifies subjects to which the module subscribes.



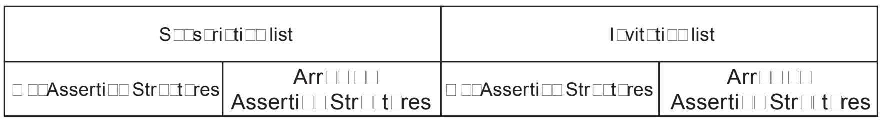
5.1.5.13 Invitation List

An invitation list is the concatenation of a 16-bit integer indicating the number of invitation assertion structures in the list, followed by that number of invitation assertion structures. The list identifies subjects to which the module does not subscribe but on which it is willing to accept messages.



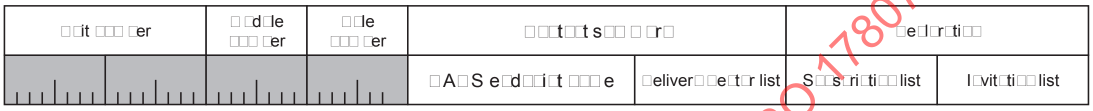
5.1.5.14 Declaration Structure

A declaration structure is the concatenation of a subscription list and an invitation list.



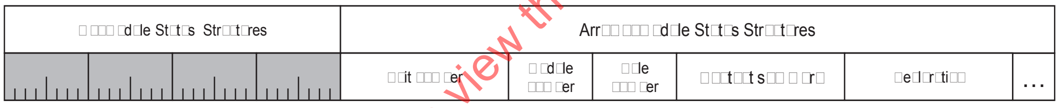
5.1.5.15 Module Status Structure

A module status structure is the concatenation of the 16-bit integer unit number and 8-bit integer module number that identify some module, the 8-bit integer identifying the role in which that module originally registered, the contact summary for that module, and the declaration structure for that module.



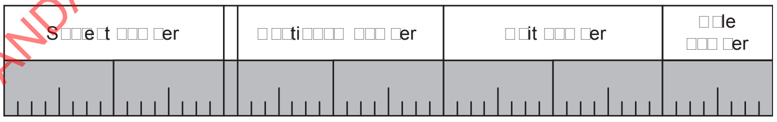
5.1.5.16 Module Status List

A module status list is the concatenation of a 32-bit integer indicating the number of module status structures in the list, followed by that number of module status structures.



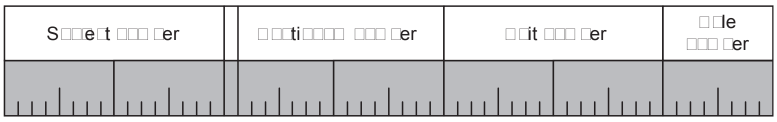
5.1.5.17 Subscription Cancellation Structure

A subscription cancellation structure is the concatenation of a 16-bit integer subject number; a single reserved bit, for alignment, followed by a 15-bit integer identifying the continuum containing all the modules to which the canceled subscription pertains; a 16-bit integer identifying the unit containing all the modules to which the canceled subscription pertains; and an 8-bit integer identifying the role characterizing all the modules to which the canceled subscription pertains.



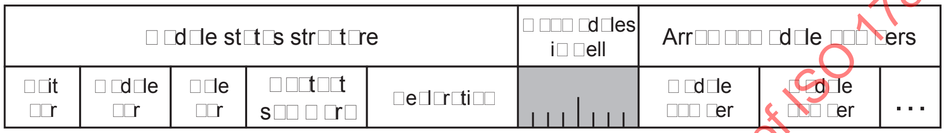
5.1.5.18 Invitation Cancellation Structure

An invitation cancellation structure is the concatenation of a 16-bit integer subject number; a single reserved bit, for alignment, followed by a 15-bit integer identifying the continuum containing all the modules to which the canceled invitation pertains; a 16-bit integer identifying the unit containing all the modules to which the canceled invitation pertains; and an 8-bit integer identifying the role characterizing all the modules to which the canceled invitation pertains.



5.1.5.19 Reconnect Structure

A reconnect structure is the concatenation of the module's own module status structure and the cell's module list.



5.1.5.20 Refusal Reason

Refusal reason is an eight-bit integer code explaining the rejection: '1' = duplicate registrar, '2' = cell census still in progress, '3' = cell is full, '4' = no such unit. Refusal reason codes '0' and '5' through '255' are reserved for future use.

5.1.6 META-AMS PROTOCOL DATA UNIT STRUCTURES

The contents of MAMS protocol data units shall be as specified in table 5-3.

Table 5-3a: MAMS PDU Structures (Sorted by Type)

Type		Reference	Supplementary Data
01	he□rt□e□t	Heartbeat source	None
02	re□e□ti□□	Echo	Refusal reason
03	□□□□□re□de□d	0	None
0□	registr□r□□□ted	Echo	None
0□	registr□r□□□□□□□□ □	Echo	None
0□	re□□□□e□ted	Echo	None
0□	□□□□□□□□e□registr□r	0	MAMS endpoint name
0□	i□vite	Module ID	Invitation assertion structure
0□	disi□vite	Module ID	Invitation cancellation structure
10	□ell□s□e□	Echo	Cell descriptor
1□	registr□r□□□er□	Query number	MAMS endpoint name
1□	□ □d□le□registr□ti□□	Query number	Contact summary
20	□□□□□re□i□	Echo	Assigned module number
21	l□□□ □st□rti□g	Module ID	Contact summary
22	l□□□ □here	0	Module status list
2□	s□□s□ri□e	Module ID	Subscription assertion structure
2□	□□s□□s□ri□e	Module ID	Subscription cancellation structure
2□	l□□□ □st□□□i□g	Module ID	None
2□	re□□□□e□t	Query number	Reconnect structure
2□	□ell□st□t□s	Module ID (but module and role numbers in ID are 0)	Module list
2□	□ □d□le□h□s□st□rted	Module ID	Contact summary
30	l□□□ □r□□□i□g	0	None
31	□ □d□le□st□t□s	Module ID (but module and role numbers in ID are 0 for any aggregate/deferred MPDU issued by a registrar)	Module status list

Table 5-3b: MAMS PDU Structures (Sorted Alphabetically)

Type		Reference	Supplementary Data
0□	□□□□□□□e□registr□r	0	MAMS endpoint name
10	□ell□s□e□	Echo	Cell descriptor
2□	□ell□st□t□s	Module ID (but module and role numbers in ID are 0)	Module list
0□	disi□vite	Module ID	Invitation cancellation structure
01	he□rt□e□t	Heartbeat source	None
22	I□□□ □here	0	Module status list
30	I□□□ □r□□□i□g	0	None
21	I□□□ □st□rti□g	Module ID	Contact summary
2□	I□□□ □st□□□i□g	Module ID	None
0□	i□vite	Module ID	Invitation assertion structure
2□	□ □d□le□h□s□st□rted	Module ID	Contact summary
1□	□ □d□le□registr□ti□□	Query number	Contact summary
31	□ □d□le□st□t□s	Module ID (but module and role numbers in ID are 0 for any aggregate/deferred MPDU issued by a registrar)	Module status list
2□	re□□□□e□t	Query number	Reconnect structure
0□	re□□□□e□ted	Echo	None
0□	registr□r□□□ted	Echo	None
1□	registr□r□□□er□	Query number	MAMS endpoint name
0□	registr□r□□□□□□□□ □	Echo	None
02	re□e□ti□□	Echo	Refusal reason
2□	s□□s□ri□e	Module ID	Subscription assertion structure
2□	□□s□□s□ri□e	Module ID	Subscription cancellation structure
03	□□□□□re□de□d	0	None
20	□□□□□re□i□	Echo	Assigned module number

5.2 AAMS PROTOCOL DATA UNITS

5.2.1 GENERAL

5.2.1.1 Each AAMS message shall consist of a header in fixed format followed by zero or more octets of application data, possibly followed by a 16-bit checksum. The length of the application data in an AAMS message shall vary as indicated by the application data length field of the header.

5.2.1.2 The AAMS message header shall consist of the fields shown in table 5-4. The PDU header fields shall be transmitted in the order of presentation in table 5-4.

Table 5-4: AAMS Message Header Fields

Field	Length (bits)	Values	Comment
Version number	2	'00' for this version of AMS.	
Message type	2	'0' = Unary, '1' = Query, '2' = Reply. '3' is reserved.	A Unary message is one that does not contribute to a query/reply exchange. A Query message is one to which a responding Reply message is requested. A Reply message is the response to a Query, echoing the context number of the Query message.
Priority	4	1-15	Lower numeric values signify 'higher priority', i.e., greater urgency in delivery of the data. Priority value 0 is reserved for internal high-priority event processing, for use by application code. Any AAMS message that has priority set to 0 is an ill-formed message.
Flow Label	8		
Checksum flag	1		Set to 1 if the application data of this message is followed by a 16-bit checksum; otherwise zero.
Source continuum number	15		To support Remote AMS operations.
Source unit number	16		
Source module number	8		
Reserved	8	Always zero.	May be redefined in future versions.
Context number	32		
Message subject number	16		
Application data length	16		Limited to 65,000.

5.2.2 RESPONSIVE MESSAGES

5.2.2.1 For any AAMS message issued in response to a **Publish.request** primitive, a **Send.request** primitive, or an **Announce.request** primitive:

– the message type shall be set to Unary;

– context number may be either zero or non-zero; if non-zero, it serves merely as ancillary information.

5.2.2.2 For any AAMS message issued in response to a **Query.request** primitive:

– the message type shall be set to Query;

– context number shall be non-zero, identifying the context within which the Query should be answered.

5.2.2.3 For any AAMS message issued in response to a **Reply.request** primitive:

– the message type shall be set to Reply;

– context number shall be equal to the context number of the message to which this reply is a response.

5.3 REMOTE AMS PROTOCOL DATA UNITS

5.3.1 GENERAL

For the purposes of Remote AMS (RAMS) communication, AAMS message traffic must be encapsulated in other data structures that enable forwarding through RAMS gateways. This encapsulation is described here.

5.3.2 REMOTE AMS PDUS

Each RPDU shall comprise a single *envelope* structure as described below.

5.3.3 ENVELOPES

5.3.3.1 The data structures used to direct the forwarding of AAMS message traffic by source and destination RAMS gateways are termed *envelopes*.

5.3.3.2 Each envelope shall consist of a header in fixed format followed by zero or more octets of content; if the length of the content exceeds zero, then the content shall be an encapsulated AAMS message. Envelopes with control code ‘2’ or ‘3’ (petition assertions and cancellations) must have zero octets of content.

5.3.3.3 The envelope header shall consist of the fields shown in table 5-5. The envelope header fields shall be transmitted in the order of presentation in table 5-5.

Table 5-5: Envelope Header Fields

Field	Length (bits)	Values	Comment
Version number	2	'00' for this version of RAMS.	
Reserved	2	Always '00'	May be redefined in future versions.
Control code	4	'2' = petition assertion '3' = petition cancellation '4' = publish on reception '5' = send on reception '6' = announce on reception	All other values are reserved for future use.
Reserved	9	Always '000000000'	May be redefined in future versions.
Continuum number	15		
Unit number	16		
Source ID number	8		
Destination ID number	8		
Subject number	16		
Length of content	16		

5.3.3.4 The contents of envelope header fields shall be constrained as described in table 5-6.

Table 5-6: Envelope Header Field Contents

Control code	Continuum number	Unit number	Source ID number	Destination ID number
'2'	Identifies publisher(s)	Identifies publisher(s)	Role number of publisher(s)	0
'3'	Identifies publisher(s)	Identifies publisher(s)	Role number of publisher(s)	0
'4'	0	0	Source role number	0
'5'	Destination of message	Destination of message	Source role number	Destination module number
'6'	Destination of message	Destination of message	Source role number	Destination role number

5.3.4 ENCAPSULATED AAMS MESSAGES

5.3.4.1 Encapsulated AAMS messages may constitute the contents of envelopes (described above) received by source and destination RAMS gateways.

5.3.4.2 When destination RAMS gateways send, announce, or publish messages, the application data of those messages are encapsulated AAMS messages that were received as the contents of envelopes.

5.3.4.3 The encapsulated AAMS messages that are the application data of messages sent, announced, or published by destination RAMS gateways and received by application modules are, in practical effect, AAMS messages that were sent, announced, or published by application modules in remote continua.

5.4 IMPLEMENTATION NOTE

It is recognized that a single primary transport service endpoint could be used for reception of both AAMS and MAMS PDUs if a flag were added to the headers of both to disambiguate between them. However, it would only be possible to do so by adding at least one byte to each header, which would (a) somewhat increase protocol overhead and (b) break word alignment in both headers, possibly requiring the insertion of another byte (or perhaps even three bytes) to recover alignment, at even greater cost in overhead.

Since the disadvantage of providing this flag would be a recurring cost in every transmission, no such flag was added. Developers of AMS implementations need to be aware of its absence and use separate, dedicated transport protocol endpoints for AAMS versus MAMS traffic, so that—in effect—the transport protocol endpoint identifier disambiguates between the two traffic flows.

6 USER OPERATIONS

NOTE – This section describes standard interoperable procedures which are not included in AMS itself but which use the services provided by AMS.

6.1 AMS PROXY ARCHITECTURE OVERVIEW

The AMS Proxy (AMSP) architecture is designed to offer AMS-based information exchange services to a community of non-AMS application programs. The architecture is as follows:

- One or more AMS application programs function as AMSP servers.
- Each AMSP server subscribes to messages on one or more AMS subjects on behalf of one or more non-AMS application programs functioning as AMSP clients.
- The server forwards information contained in AAMS messages to its clients, subject to client-specified processing.
- In addition, the server may receive information from its clients and use AMS to publish that information on its clients' behalf.

This general architecture can serve several purposes:

- An AMSP server can function as a message filtering agent, receiving AAMS messages and forwarding different subsets of the data in those messages to its clients.
- An AMSP server can decouple message reception from message transmission. AAMS messages are delivered immediately, but for some purposes it may be important for the time of reception of an AAMS message to differ from its time of transmission by some arbitrary interval; e.g., the recipient of a message may not be running at the time the message is transmitted. An AMSP server can function as a reception agent, storing AAMS messages in durable queues until its clients request them.
- An AMSP server can perform 'impedance matching' between message systems. That is, it may exchange messages with its clients at data rates that are much lower—or much higher—than the rate at which it publishes and receives AAMS messages.
- An AMSP server can function as a bridge between an AMS environment and a non-AMS information exchange system. For example, an AMSP server could process Remote Procedure Calls (RPCs) or offer Web services via HTTPS.

Specific implementations of the AMSP architecture may be based on a variety of client/server interchange protocols. AMSP clients that utilize different client/server interchange protocols may interoperate, however, because their interoperation is entirely through the agency of AMSP servers which all utilize AMS for communication among themselves.

6.2 AMQP-BASED AMSP

6.2.1 Application software that utilizes an AMSP implementation based on the Advanced Message Queuing Protocol (AMQP) shall conform to the AMQP specification (reference [4]—see also reference [F1]) as follows:

6.2.1.1 AMSP servers shall function as AMQP messaging middleware servers.

6.2.1.2 AMSP clients shall function as AMQP clients.

6.2.1.3 All data communicated between AMSP clients and AMSP servers shall be encapsulated in AMQP frames.

6.2.1.4 All data communicated among AMSP servers shall be encapsulated in messages conforming to the AMS Recommended Standard.

7 CONFORMANCE CLASSES

7.1 OVERVIEW

Not all capabilities of AMS will necessarily be needed everywhere the standard is used. For example:

– An AMS deployment embedded within a robot might be a self-contained message space whose configuration remains static throughout the robot's mission. In this case, the automated meta-AMS procedures may be unnecessary.

– A real-time AMS deployment might utilize only private message exchange, choosing not to exercise message publication functions because their execution latency is non-deterministic (as it may vary with the number of subscribers).

In order to extend standard AMS interoperation to a broad range of operational contexts, eight distinct classes of conformance to the AMS standard are defined. The interoperability of any pair of AMS modules whose implementations satisfy defined AMS conformance classes can be determined from the intersection of their conformance classes' sets of capabilities.

7.2 CAPABILITY SETS

7.2.1 For the purposes of conformance class definition, AMS functionality is partitioned into the following five sets of capabilities:

a) **basic message exchange**: SEND and RECEIVE;

b) **synchronous message exchange**: QUERY and REPLY;

c) **multi-point message distribution**: PUBLISH and ANNOUNCE;

d) **the Meta-AMS protocol**;

NOTE – In the absence of the Meta-AMS capability it is the responsibility of the implementation to provide AAMS message sources with all information needed in order to direct messages to relevant destinations, i.e., the effects produced by the INVITE capability (in support of basic message exchange) and the SUBSCRIBE capability (in support of multi-point message distribution). The manner in which this may be accomplished is beyond the scope of this Recommended Standard.

e) **the Remote AMS protocol**, including support for RAMS networks configured both as meshes and as trees.

7.2.2 An AMS module whose implementation conforms to a given conformance class shall implement all of the procedures that are included in all capability sets encompassed by that class. The specific procedures included in each capability set shall be as defined below. The capability sets encompassed by each conformance class shall be as shown in table 7-1.

Table 7-1: Capability Sets Encompassed by Conformance Classes

Class number	Class Description	Basic	Synchronous	Multi-point	MAMS	RAMS
1	Static basic. For static configurations where no message bus is needed.	X				
2	Dynamic basic. For dynamic configurations where no message bus is needed.	X			X	
3	Static client/server. Supports client/server operations in a static configuration.	X	X			
4	Dynamic client/server. Supports client/server operations in a dynamic configuration.	X	X		X	
5	Static closed bus. Supports operations of a single message bus in a static configuration.	X	X	X		
6	Dynamic closed bus. Supports operations of a single message bus in a dynamic configuration.	X	X	X	X	
7	Static open bus. Supports operations over multiple interconnected message buses in a static configuration.	X	X	X		X
8	Dynamic open bus. Supports operations over multiple interconnected message buses in a dynamic configuration.	X	X	X	X	X

7.3 PROCEDURES INCLUDED IN CAPABILITY SETS

7.3.1 BASIC MESSAGE EXCHANGE PROCEDURES

The basic message exchange capability set shall include all procedures defined in 4.3.1, 4.3.4, and 4.3.8 of this Recommended Standard.

7.3.2 SYNCHRONOUS MESSAGE EXCHANGE PROCEDURES

The synchronous message exchange capability set shall include all procedures defined in 4.3.5 and 4.3.6 of this Recommended Standard.

7.3.3 MULTI-POINT MESSAGE DISTRIBUTION PROCEDURES

The multi-point message distribution capability set shall include all procedures defined in 4.3.2 and 4.3.7 of this Recommended Standard.

7.3.4 META-AMS PROCEDURES

The meta-AMS capability set shall include all procedures defined in 4.2 of this Recommended Standard.

7.3.5 REMOTE AMS PROCEDURES

The Remote AMS capability set shall include all procedures defined in 4.3.3 and 4.4 of this Recommended Standard.

(Blank page)