CCSDS Concept Paper: Aligning Messaging Efforts

Scott Burleigh
Stuart Fowell
Amalaye Oyake
June 6, 2005

Introduction

This document is a concept paper for the Consultative Committee for Space Data Systems (CCSDS). CCSDS concept papers are working documents of the Consultative Committee for Space Data Systems (CCSDS), its areas, and its working groups. CCSDS concept papers are valid for a maximum of nine months and may be updated, replaced, or obsoleted by other documents at any time.

Abstract

This concept paper proposes redirection of several current CCSDS standardization efforts on application message exchange, with the aim of eliminating duplication of effort. Specifically, it proposes:

- That those key elements of the SOIS Message Transfer Service (MTS) service specification that are not already addressed by the SIS Asynchronous Message Service (AMS) service specification be added to the latter.
- That development of MTS as a distinct product be discontinued.
- That the AMS specification be further augmented, as necessary, to assure that it satisfies all requirements imposed on the MOIMS System Monitor and Control Protocol (SMCP) that pertain to general-purpose messaging.
- That SMCP adopt AMS as its standard underlying end-to-end messaging service, and that the SMCP effort focus on the standardization of SMC application message syntax and semantics.

Background

The Message Transfer Service (MTS) effort within the CCSDS Spacecraft On-board Interface Services (SOIS) area is aimed at defining "a standard service for mediating the transfer of discrete data between on-board software applications in a (potentially) distributed, on-board system." It is documented in the "Spacecraft On-board Interface Services – Message Transfer Service" Red Book, dated April 2005.

The Spacecraft Monitor and Control Protocol (SMCP) effort within the CCSDS Mission Operations and Information Management Services (MOIMS) area is "an application level protocol designed to meet the need for standardized spacecraft message syntax and semantics." It is documented in the "Spacecraft Monitor and Control Protocol" Red Book, draft 0.4, dated January 2005. Additional remarks on the design intent of SMCP

are noted in a detailed email from Roger Thompson posted to the [sis-ams] mailing list on 28 January 2005.

The Asynchronous Message Service (AMS) effort within the CCSDS Space Internetworking Systems (SIS) area defines "a standard system of communication – *messaging* – among mission software modules." It is documented in the "Asynchronous Message Service (AMS)" concept paper dated August 2004.

It was observed at a CCSDS cross-area meeting in Athens (11 April 2005) that these three projects all seem to be directed at standardizing application message exchange in one way or another and that CCSDS resources might more effectively be allocated if they were aligned so as to eliminate any duplication of effort. A study of this problem was commissioned, and the present concept paper is the final report from that study.

Analysis and Discussion

Aligning MTS and AMS

At present, the MTS specification comprises only a detailed service interface definition, with no definition of a supporting wire protocol. The AMS specification defines both a service interface and the corresponding wire protocol. The two service interface definitions overlap significantly, with little apparent conceptual conflict, but each includes useful elements that the other omits. Nearly all of the MTS service elements that are not already included in AMS constructively complement the current AMS service elements.

The best way to align the two specifications, then, appears to be simply to (a) expand the AMS service interface to encompass the union of the current MTS and AMS service interfaces and (b) enhance the AMS wire protocol definition accordingly. Specific elements of this expansion are discussed below; for details, please see Annex 2.

"Connection" awareness

The currently specified AMS *Register*, *Unregister*, *Assert_subscription*, and *Cancel_subscription* indications notify the AMS user application of relevant changes in the configuration of the message exchange continuum. The user application can use this information to manage whatever concept of functional "connection" between itself and other user applications it finds valuable.

However, no such indication is currently provided with regard to subjects on which user applications are willing to accept messages but to which they don't subscribe. We propose to add service requests that assert and rescind "invitations" in addition to subscriptions, and the corresponding *Assert_invitation* and *Cancel_invitation* indications will remedy this omission.

As noted below, both subscriptions and invitations will be characterized by Quality of Service (QoS) specifications indicating the priority and diligence with which messages on the indicated subjects are to be transmitted. This enables association of QoS with connections, somewhat as in MTS Red Book section 3.3.2.3.2, but at message subject granularity.

Abstract Quality of Service

We propose to provide a **QoS specification** parameter for the service primitives that assert subscriptions and invitations. QoS specification indicates:

- The priority at which messages on the indicated subject are to be transmitted to this user application: Express, Standard, Bulk.
- The "diligence" with which these messages are to be transmitted: Assured (implying the need for some highly reliable transmission mechanism, such as ARQ or erasure coding) or Best-effort (implying that no such diligence is required).

AMS will use locale-specific QoS analysis rules in the Management Information Base (MIB) to infer from the QoS specification the specific transmission mechanisms to invoke. For example, in some contexts QoS = (Assured, Express) might imply the use of the TCP "urgent" feature across a local area network.

Message delivery deadlines, notifications of message delivery success and failure

We propose to provide a **handling instructions** parameter for the service primitives that send original (non-reply) messages. Handling instructions indicate:

- Whether or not the message sender is to be notified when delivery failure is confirmed.
- Whether or not the message sender is to be notified when delivery success is confirmed.
- Optionally, the *deadline* by which delivery must be accomplished in order to be considered successful.

A new Handling.indication primitive will report on confirmed delivery failure or success as requested in handling instructions.

Security

Although there is no detailed definition of security services in the MTS Red Book, the need for security measures in messaging is identified in the Requirements section of that document. We therefore propose to add the following optional security elements to the AMS specification:

- Node (user application) authentication
- Node service authorization
- Message confidentiality and integrity

For an overview of the specific mechanisms proposed, see section 2.3.8 in Annex 2.

Aligning SMCP with MTS and AMS

One suggestion as to what may be the right way to align SMCP with MTS (and with AMS as well, if MTS and AMS are aligned as proposed above) appears in the second

sentence of the SMCP Red Book's section 6.2, in a discussion of the underlying link protocols to use in the on-board system environment: "…candidates for an onboard messaging service (MTS, CORBA, VxWorks Messages, etc)."

That is, SMCP is designed to be an application-level protocol that facilitates interoperable end-to-end monitor and control of spacecraft subsystem elements – not a general-purpose messaging service but rather an application capability which requires the support of those general-purpose messaging services that are appropriate to the various environments that an SMC message must traverse in its end-to-end path.

AMS as augmented with MTS services as described above can be one such service, and potentially the only one that is needed. AMS as currently defined supports both the request/response communication pattern required by SMCP (timeouts would need to be added – a new *Query.request* primitive – but this is not difficult) and the publish/subscribe pattern. SMCP messages would be carried as the payloads of AMS messages. Since AMS handles the application-level message addressing and dispatching, and therefore can report on the identity of the sender of each message, the sending TARGET_NAME and CONTROLLER_NAME could be omitted from SMCP messages. The AMS publish/subscribe functionality would make the SCMP REGISTER and DEREGISTER messages unnecessary; the remaining SCMP messages implement core SCMP functionality and do not overlap with AMS capabilities.

At the same time, there are aspects of the current SMCP specification that would likely be useful to a wide range of messaging-based application-level services, not just SMCP. These design elements could be abstracted from SMCP and realized instead in AMS, yielding a simpler SMCP design and a more powerful and useful general-purpose messaging service. Some discussion is offered below; for details, please see Annex 2.

Domain-independent transmission

As noted in the second paragraph of the SMCP Red Book section 1.3: "The SMCP will operate over heterogeneous links using data-link protocols native to those environments." This ability to operate over a space network of heterogeneous links is critical to the success of SMCP as a "common end-to-end monitor and control protocol"[1].

However, that ability is potentially useful to protocols beyond SMCP as well. [We note that CFDP's extended procedures and store-and-forward overlay service are in part intended to provide such a capability in the context of end-to-end file transfer.]

The AMS Concept Paper already provides this domain-independent transmission feature, implicitly, by offering the Delay-Tolerant Networking architecture's "Bundling" protocol as on of its own underlying transport services[2]. Bundling is expressly designed to convey

[1] SMCP Red Book section 1.1.

[2] See AMS Concept Paper section 2.3.1, second paragraph, and section 6.

data over an arbitrary series of heterogeneous links, and moreover to do so in a manner that is unaffected by the long signal propagation delays and frequent lapses in connectivity that characterize a space network.

Although the design of the Bundling protocol is still somewhat mutable at this writing, a number of implementations exist and the reference implementation is in use by a rapidly growing research community. Bundling, and the AMS adaptation enabling its use of Bundling, will be stable in time to support SMCP development and testing some time in 2006. We therefore propose that SMCP rely on AMS for end-to-end transmission of messages across the heterogeneous space network.

Content coding

The SMCP Red Book contains a number of allusions to the problem of encoding spacecraft message data in a manner that is suitable for transmission over heterogeneous and potentially bandwidth-constrained links. In particular it is noted that "Considerations must be made for a standard encoding of spacecraft messages."[3]

But the problem is arguably even more profound than the efficient encoding of spacecraft message data: for true interoperability it might be desirable to provide each mission or program with a standard means of dynamically defining the syntax of its own application-specific messages – the data parameters of each message – and automatically assuring that message encoding conforms to that syntax. This capability is one of the advantages of building mission data systems around a centrally managed "mission information model" as in the Command, Control, Communications, and Information architecture that has been proposed for NASA's Constellation program.

We therefore propose that message content encoding and decoding be part of the message transmission service provided by AMS: on transmission, message content will be marshaled from a format that is locally suitable for processing into one that is suitable for transmission; on message reception, message content will be un-marshaled back into local format. Message syntax and marshaling procedures will be encoded in the AMS Management Information Base (MIB), which would supplant (and be much more capable than) the "subject service" currently included in the AMS design. By adopting this model we open the door for the definition of a separate mission information model management and propagation service that, notionally, would populate the MIBs of AMS nodes. That service would be independent of both AMS and SMCP and would support not only these protocols but potentially a wide range of others.

---

[3] SMCP Red Book section 2.4. It is also proposed in section 7 of the Red Book that the encoding of a message might be different on different links of the end-to-end path, depending on the resource constraints to which each link is subject. However, changing the manner in which message content is encoded imposes some computational overhead and can complicate network and application troubleshooting. A better approach might be to select a single efficient standard encoding and use it on all links, even including those in which bandwidth constraints are not so severe as to make it strictly necessary.

Recommendation

Summing up the analysis presented above, we propose:

- That those key elements of the SOIS Message Transfer Service (MTS) service specification that are not already addressed by the SIS Asynchronous Message Service (AMS) service specification be added to the latter: abstract quality of service, message delivery deadlines, notification of message delivery success and/or failure, "connection" awareness, and security.
- That development of MTS as a distinct product be discontinued.
- That the AMS specification be further augmented, as necessary, to assure that it satisfies all requirements imposed on the MOIMS System Monitor and Control Protocol (SMCP) that pertain to general-purpose messaging: content coding (with reference to a mission information model that, for now, remains notional) and domain-independent transmission.
- That SMCP adopt AMS as its standard underlying end-to-end messaging service, and that the SMCP effort focus on the standardization of SMC application message syntax and semantics.

References

Asynchronous Message Service (AMS) Concept Paper, August 2004.

Bundle Protocol Specification, draft-irtf-dtnrg-bundle-spec-02.txt, September 2004.

Spacecraft On-board Interface Services – Message Transfer Service Red Book, April 2005.

System Monitor and Control Protocol Red Book, January 2005.

Annex 1: Email from Roger Thompson

From: <Roger.Thompson@scisys.co.uk>
To: "Scott Burleigh" <Scott.Burleigh@jpl.nasa.gov>
Cc: <Sam.Cooper@scisys.co.uk>
Subject: Re: [Sis-ams] AMS deliberations
Date: Thursday, January 27, 2005 8:56 AM

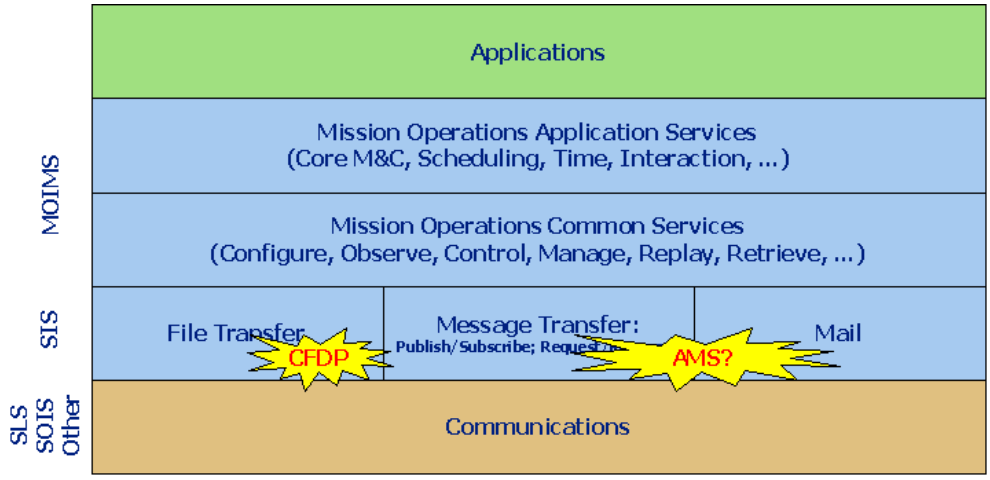
Hello Scott,

Happy New Year!

November does seem a long time ago, so I'm afraid I am having a little difficulty remembering what the specific issue was that I brought up that was not addressed by your initial AMS concept. My main recollection is how well the concept fitted with the MOIMS Mission Operations [MO]service concept.

I will try and summarise our messaging requirements.
The application level MO services would be overlaid on a Common service layer. The MO Common layer provides support for a [hopefully small] set of interaction patterns. These in turn would be implemented through [potentially multiple] underlying communications technologies/protocols. Obviously, it would unify the CCSDS world if there is a good match between these and the messaging services provided by SIS.

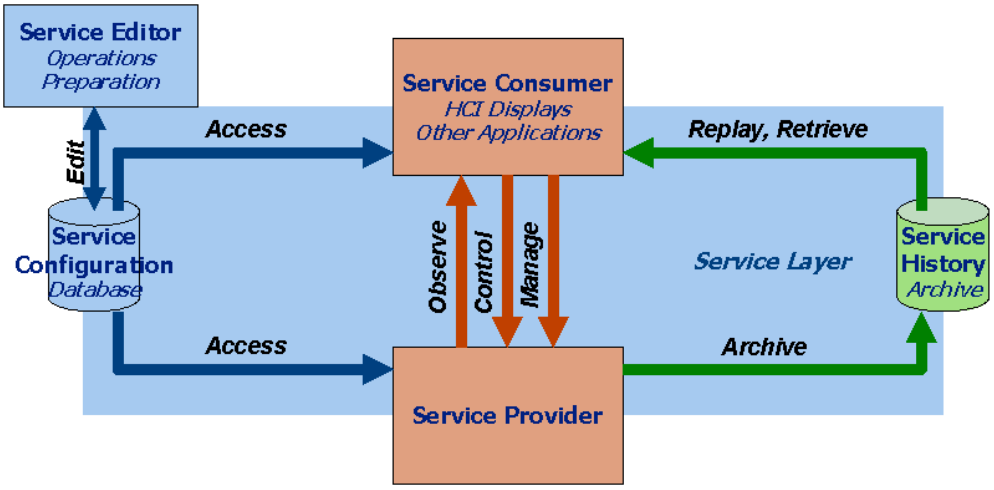
I attach I diagram I put together during the November meeting after your presentation ...



We are in the process of, but have not completed, definition of the MO Common Services. My preliminary view is that we only need a small set of fundamental interaction patterns:

- Publish Subscribe
- Request/Response
- Mail?
- File Transfer

The following shows a summary view of our common service pattern for MO services:



It is the red services between service consumer and provider that we see as being the principle subject for AMS, although the Replay service should have a high degree of commonality with Observe.

Publish/Subscribe would support the MO Common "observe" service, in which the service Consumer would register interest in a subset of available objects and the Provider would respond with asynchronous status update messages. The published objects are either pre-defined [in the configuration data] and known in advance to both consumer and provider, or dynamically created by the provider, in which case the consumer can only subscribe to a specific object if its ID has been obtained in advance [e.g. in a response to a Control message], otherwise subscription is to a scoped sub-set.

Request/Response would be used for transactional message exchanges, and could be used to support the MO Common "control" and "manage" services. Two cases have been identified: 1) a control message from service Consumer to the Provider, with optional response(s) from the Provider; 2) a request message from service Consumer, with an associated response message containing the requested data.

By "Mail" I mean an e-mail like service, involving delivery of messages to a mailbox, that the recipient can retrieve messages from at some later point in time. Useful for unsolicited messages from Provider to Consumer.

I'm not sure whether or not an e-Mail type interaction is necessary, or even whether it is significantly different to Request/Response. I suppose we were starting from the position that the Request/Response interface is based on an open end-to-end connection/session between consumer and provider, while Mail implied some form of connectionless store-and-forward. However, this distinction may not be necessary if requests and responses do not require an open connection, in order to handle discontinuous contact.

File/Transfer would allow for bulk data transfer between service consumer and provider, in either direction. In a SIS context, we assumed this would be provided by CFDP, not EMS.

All services would need to exist within a partitioned context. In this sense the world is partitioned in two dimensions:

"Domain":

This is used to decompose the world into a scoping hierarchy, to ensure uniqueness of identifiers, etc.: i.e. Mission.Spacecraft.Subsystem.Component.Object

The same service may be instantiated for multiple domains. Its a good idea to make sure you're connecting to the right spacecraft ...

"Session" or Timeframe or, I believe the term you used was "Continuum":

This is used to separate data that would otherwise be potentially identically keyed - live operational data, multiple simulated or test data series, and maybe historical replay data series.

The timeframe can be different for each session: the live session is presumably synchronised with the present; simulated sessions can be in the past, present or future; replay sessions can be past or future. Timeframe for simulated and replay is externally controllable - it may be paused and stepped; or evolve at a real-time rate, faster or slower than real-time, or at a random rate [e.g. as fast as possible]. Not sure if this has any impact at all, other than the potential for "continua" to be dynamically created ...Not all services would be available for replay sessions - you can observe history, but not change it.

Other issues to be considered:
- User [Consumer] Authentication: access rights may be limited to certain services for certain domains and certain classes of session
- Security
- Quality of Service

Not sure if that's got close enough to the issue to identify the issue raised in November.
Sorry its a bit of a ramble ...

Roger Thompson
SciSys Ltd.

Annex 2: Proposed AMS Service Interface

1 Introduction

1.1 Purpose and Scope

1.2 Applicability

1.3 Conventions and Definitions

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 (nodes) rather than through any central message dispatching nexus.

- Messages are automatically conveyed using the "best" (typically – though not necessarily – the fastest) 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 zones enables an application instance to comprise hundreds or thousands of cooperating modules without significant impact on application performance.

2.1.2 Message exchange models

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

Deleted: All

Although message exchange among nodes 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, AMS enables a node to include a *context number* in any original (non-reply) message; AMS procedures can be used to reply to any original message, whether published or sent privately, 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.

Deleted: all

For some purposes true message synchrony may be necessary as well: that is, it may be desirable for a node that has issued a message to suspend operations altogether until a reply is received – to *query* some other node. AMS procedures additionally support this communication model when it is required.

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

- A node uses AMS procedures to announce that it is *subscribing* to messages on a specified subject.
- From that time on (until the subscription is canceled), whenever any node in the message space uses AMS procedures to *publish* a message on that subject, a copy of the message is automatically delivered to that subscribing node and to all others that have announced similar subscriptions.

This model can greatly simplify application development and integration. In effect, each node 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 node 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 node is largely decoupled from the designs of all other nodes 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 node to send a message privately and explicitly to some specific node, e.g., in reply to a published message. Again, AMS procedures support this communication model as well when it is required.

2.2 Architectural elements

2.2.1 General

The architectural elements involved in the asynchronous message service protocol are depicted in Figure 1 and described below.

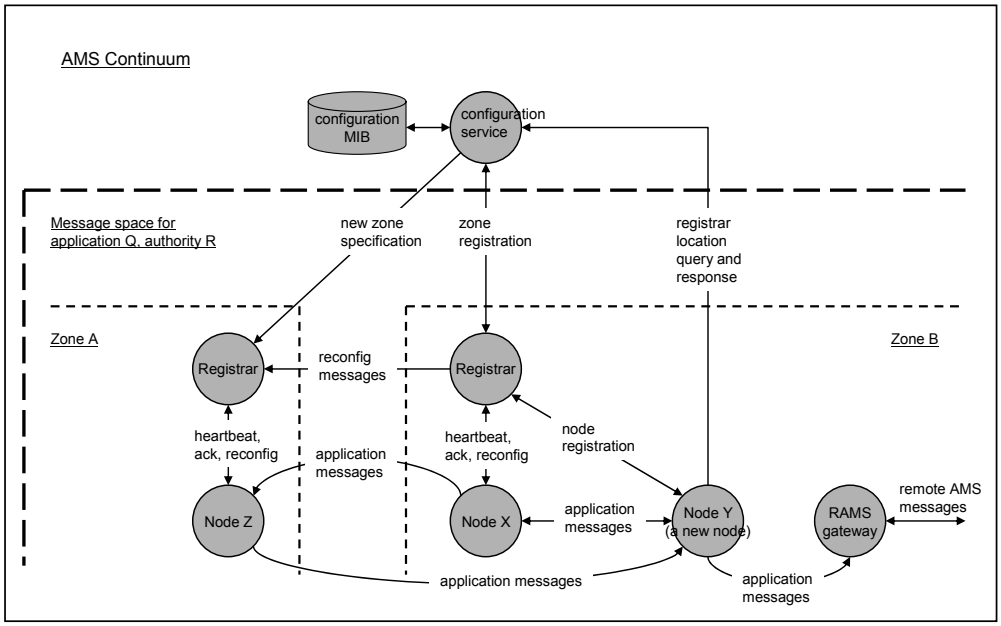


Figure 1: Architectural Elements of AMS

2.2.2 Communicating entities

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

Deleted: four

Deleted: subject servers,

A *registrar* is a communicating entity that catalogues information regarding the nodes that populate a single zone 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.

Deleted: A *subject server* is a communicating entity that manages a database of subject names and corresponding subject numbers for a single message space. It responds to queries for this information, and it updates this information as changes are announced.¶

Deleted: subject servers and all

2.3 Overview of interactions

2.3.1 Transport services for application messages

AMS occupies a position in the OSI protocol stack model somewhere between level 4 (Transport) and level 7 (Application); AMS might best be characterized as a messaging "middleware" protocol. 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 node to the memory of the receiving node. It additionally relies on those capabilities to accomplish the transmission of *meta-AMS* (or *MAMS*) messages to and from registrars and configuration servers that enables the dynamic self-configuration of AMS message spaces.

Deleted: , subject servers,

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* or *PTS*. Selection of the PTS for a continuum is an implementation matter; the "Bundling" protocol of the Delay-Tolerant Networking architecture is a plausible choice, but for some continua an implementation based on UDP/IP may be more appropriate.

The PTS clearly can also be used for application message exchange among all nodes 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 nodes 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* or *STSs*.

2.3.2 Registrar registration

Every message space always comprises at least one zone, and each node resides within (is registered in) some zone; in the simplest case all nodes of the message space reside in the same zone. Each zone is served by a single registrar, which is responsible for monitoring the health of all nodes in the zone and for propagating six kinds of message space configuration changes: node registrations and terminations, subscription and invitation assertions, and subscription and invitation cancellations. On receipt of one of

Deleted: four

these reconfiguration messages from a node in its own zone, the registrar immediately propagates the message to every other node in the same zone and then to the registrars of all other zones in the message space; on receiving such a message from a remote zone's registrar, the registrar propagates it to all nodes in its own zone.

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, in order of descending preference, must be "well known" – that is, included in the AMS management information bases (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 nodes to register. (The manner in which this latter requirement is satisfied is an implementation matter. One advantage of selecting the DTN Bundling protocol as the PTS for a continuum is that the "resilient delivery" features of Bundling may provide a simple solution for this problem: when a configuration server registers with its bundle agent, it can supply a script that will reanimate that server automatically in the event that a bundle carrying a MAMS message arrives following a configuration server crash.)

Deleted: to the

Deleted: ,

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

2.3.3 Node registration

Each node has a *name*, an application-specific ASCII string containing no whitespace, which generally connotes its function within the application but need not uniquely identify it within its message space. Each node also has a single associated *meta-AMS access point (MAAP)* at which the node is prepared to receive MAMS messages.

Deleted: one or more "network identities", one for each underlying transport service on

A new node joins a message space by registering itself within some zone of the message space, i.e., by announcing its name and its MAAP to the zone's registrar. However, knowledge of how to communicate with that registrar can't be hard-coded into the node because the relevant registrar might be running at different network locations at different times.

Deleted: network identities

For this reason, the first step in registering a new node 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 nodes. The configuration server tells the new node how to contact its registrar. The node obtains a unique numeric *node ID* from the registrar and thereby registers. The registrar ensures that all other nodes in the message space learn the new node's name, node ID, and MAAP. Those nodes in turn announce their own names, node IDs, and MAAPs to the new node.

Deleted: its well-known network location

Deleted: network identities

Deleted: network identities

2.3.4 Monitoring node 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 nodes or per concluded subscriptions), it is important for each registrar always to detect the terminations of nodes in its zone. When a node terminates under application control it automatically notifies its registrar that it is stopping. However, if a node crashes – or the host on which a node resides is simply powered off or rebooted – no such notification is transmitted to the registrar. For this reason, every node automatically sends a "heartbeat" message to its registrar every twenty seconds. The registrar interprets three successive missing heartbeats as an indication that the node has terminated.

Whenever it detects the termination of a node (either an overt termination or a termination imputed from heartbeat failure), the registrar informs all other nodes in the zone – and, via other registrars, all other nodes in the message space – of the node's demise.

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

2.3.5 Monitoring registrar health

In addition to monitoring the heartbeats of all nodes in its zone, each registrar issues its own heartbeats to those nodes on the same cycle. Each node interprets three successive missing registrar heartbeats as an indication that the registrar itself has crashed. On detecting a registrar crash, the node 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 and resumes exchanging heartbeats.

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

Deleted: (The same is true of subject servers, as discussed later.)

Since the node heartbeat interval is twenty seconds, within the first sixty seconds after restart the registrar will have received heartbeat messages from every node that is still running in the zone and will therefore know accurately the configuration of the zone. This accurate configuration information must be delivered to new nodes at the time they start up (so that they in turn are qualified to orient a newly-restarted registrar to the zone's

configuration in the event that the registrar crashes). For this reason, during the first sixty seconds after the registrar starts it accepts only MAMS messages from nodes that are already registered in the zone (i.e., have been assigned node IDs); if it accepted and processed a registration message from a new node before being certain of the status of all old ones, it would run the risk of delivering incorrect information to the new node.

2.3.6 Configuration service fail-over

It's of course also possible for a configuration server to be killed (or for its host to be rebooted, etc.). Each registrar interprets three 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. While it is doing so it is not issuing heartbeats or responding to other messages, so eventually all nodes may infer that their registrars have crashed and therefore begin re-querying the now-dead configuration server to re-establish communication with their registrars; when the configuration server fails to respond, the nodes too will begin cycling through network locations seeking a restarted configuration server. While they are doing so, they too will not be responding to messages, so all message space activity will eventually come to a halt.

When the configuration server is restarted at one of its well-known possible network locations, however, all registrars will eventually find it and re-announce themselves to it, so that when application nodes finally find it they can re-establish communication with their registrars; all application processing 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 nodes 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 resync

Finally, every registrar can optionally be configured to re-advertise to the entire message space the detailed configuration of its zone (all active nodes, all subscriptions and invitations) at some user-specified frequency, e.g., once per minute. This capability is referred to as *configuration resync*. Configuration resync 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 application message space configuration will eventually be propagated to every node 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 relatively fault tolerant:

- When a node crashes, its registrar detects the loss of heartbeat within three heartbeat intervals and notifies the rest of the message space. Message transmission everywhere is unaffected.

- When a registrar crashes, its configuration server detects the loss of heartbeat within three heartbeat intervals and restarts the registrar. During the time that the zone has no registrar, transmission of application messages among nodes of the message space is unaffected, but the heartbeat failures of crashed nodes are not detected and reconfiguration messages originating in the zone (registrations, terminations, subscription and invitation assertions, and subscription and invitation cancellations) are not propagated to any nodes. However, after the registrar is restarted it will eventually detect the losses of heartbeat from all crashed nodes and will issue obituaries to the message space, and if configuration resync is enabled it will eventually re-propagate the lost reconfiguration messages.

- When a configuration server crashes, all activity may eventually come to a standstill. But no application nodes fail (at least, not because of communication failure), and on restart of the configuration server all activity eventually resumes.

2.3.8 Security

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 within a given message space as follows:

- The AMS MIB exposed to the configuration server contains a list of all zones in which registration service may be offered within the message space, identified by *registrar name* – the concatenation of application name, authority name, and zone name. Associated with each registrar name is the AMS public encryption key for that zone's registrar.

- The AMS MIB exposed to every registrar in the message space contains a list of all node names under which modules may register in that message space. Associated with each node name is the AMS public encryption key for the application node(s) that may register under that name.

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

Formatted: Bullets and Numbering

- The AMS MIBs exposed to the configuration server and to all registrars and application nodes in the message space contain those entities' own AMS private encryption keys.

As described later, this information is used to authenticate registrar registration and exclude spurious registrars from the message space, to authenticate node registration attempts and deny registration to unauthorized application modules, and to assure the authenticity, confidentiality, and integrity of MAMS traffic exchanged between nodes 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 node of a given message space may contain, for any subset of the message subjects (identified by name and number) used in the message space:

- a list of the names of all nodes that are authorized senders of messages on this subject;
- a list of the names of all nodes 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

Deleted: service

The structure of the content of messages on a given subject is application-specific; message content structure is not defined by the AMS protocol. However, the AMS MIB exposed to all nodes of a given message space will contain, for each message subject (identified by name and number) used in the message space:

Formatted: Bullets and Numbering

- a description of this message subject, discussing the semantics of this type of message;
- a detailed specification of the structure of the content of messages on this subject;
- optionally, a specification of the manner in which a correctly assembled message is marshaled 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 AMS is requested to send a message on a given subject, the message content 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, AMS will marshal the message content as required before transmitting the message.

When AMS receives a message on a subject for which a marshaling specification is present in the MIB, AMS will un-marshal the message content 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 contents 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 dynamically allocated and possibly sparse arrays that are indexed by subject number.

This implementation choice, however, would require that message management control arrays be large enough to accommodate the largest subject numbers used in the application. The use of extremely large subject numbers would therefore cause these arrays to consume huge amounts of memory. In general, it is best for an AMS application to use the **smallest** subject numbers possible, starting with 1. AMS MIB developers should bear this in mind.

2.3.10 Remote AMS message exchange

Because issuance of an asynchronous message need not suspend the operation of the issuing node until a response is received, AMS'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 node must wait for responses from the configuration server, the registrar, and the other nodes in its message space before proceeding with application activity. For this reason, the core AMS 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.

Deleted: message handling

Deleted: One way to ensure this is to require that applications cite message subjects by symbolic name, rather than cite the subject numbers themselves, and let the AMS infrastructure automatically assign the smallest unused subject numbers to subjects as their names are declared. To support the mapping of subject names to numbers, and vice versa, subject definition services are provided by the message space's subject server.¶
The subject server manages a private database of subject definitions for the message space. Each subject definition pairs a subject name (an application-specific ASCII string containing no whitespace) with a subject number and, optionally, a message content format string. The subject server itself assigns numbers sequentially (starting at 1) to subject names, in the order in which the subject names are declared to it by application nodes. It responds to subject number queries by returning the corresponding subject names and to subject name queries by returning the previously assigned subject numbers.¶
Subject names are also the basis for Remote AMS communication, described below. Since subject names might be declared in different sequences within different message spaces – for the same application instance, but in different continua – they may be mapped to different numbers. But so long as subject naming consistency is maintained when applications are developed, the inter-continuum Remote AMS communication between subject servers will accurately replicate subscription and publication in all message spaces.¶
The AMS heartbeat discipline monitors the health of subject servers just as it monitors the health of registrars: the configuration server interprets three successive registrar heartbeat delivery failures as an indication that the subject server has crashed. On detecting such a crash it automatically restarts the subject server, possibly on a different host. Subject servers, again like registrars, also expect heartbeats from the configuration server and respond to a configuration server failure in the same way that registrars do.¶

Formatted: Bullets and Numbering

AMS application messages may readily be exchanged between nodes in different AMS continua, however, by means of the auxiliary Remote AMS (RAMS) protocol. RAMS procedures are executed by special-purpose application nodes called *RAMS gateways*:

- Each RAMS gateway opens persistent, private RAMS communication channels to the RAMS gateways of other message spaces for the same application instance, in other continua.
- The interconnected RAMS gateways use these channels to forward subscription 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 node uses RAMS to forward the message to every other RAMS gateway to which it's linked whose message space contains at least one other node that has subscribed to messages on that subject.
- On receiving a message via RAMS from some other RAMS gateway, the RAMS gateway node simply publishes the message in its own message space.

Deleted: the subject servers of message spaces
Deleted: subject server
Deleted: subject servers
Deleted: subject servers
Deleted: subject server, acting as a node,
Deleted: subject server
Deleted: subject server
Deleted: subject server
Deleted: subject server

In this way the RAMS protocol enables the free flow of 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 channel, no matter how many subscribers will receive copies when the message reaches its destination continuum.

Again, this extension of the publish/subscribe model to interplanetary communications is invisible to application nodes. 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.

3 Service descriptions

3.1 Services provided to the application

3.1.1 Summary of primitives

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**;

Formatted: Bullets and Numbering

f) **Cancel_subscription.request**;

g) **Publish.request**;

h) **Send.request**;

i) **Query.request**;

j) **Reply.request**;

The AMS service shall deliver the following indication primitives:

a) **Message.indication**;

b) **Handling.indication**;

c) **Reply.indication**;

d) **Fault.indication**;

e) **Register.indication**;

f) **Unregister.indication**;

g) **Assert_invitation.indication**;

h) **Cancel_invitation.indication**;

i) **Assert_subscription.indication**;

j) **Cancel_subscription.indication**

Deleted: <#>**Declare_subject.request**;¶
<#>**Look_up_subject.request**.¶

Formatted: Bullets and Numbering

Deleted: ;

Deleted: <#>**Subject.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 *application name* parameter shall identify the application served by a message space's application instance.

3.1.2.2 The *authority name* parameter shall identify the organizational unit that is responsible for a message space's application instance. The combination of application name and authority name shall uniquely identify a message space.

3.1.2.3 The *zone name* parameter shall identify, within a given message space, some administrative subset of nodes.

3.1.2.4 The *node name* parameter shall indicate the functional nature of a node.

3.1.2.5 The *Meta-AMS access point (MAAP) specification* parameter shall be a *transport service endpoint specification* characterizing the manner in which a node is prepared to receive MAMS messages. The MAAP 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 C.

3.1.2.6 The *node ID* parameter shall uniquely identify a node within the message space in which it is registered.

3.1.2.7 The *subject name* parameter shall indicate the general nature of the application data in a message.

3.1.2.8 The *delivery specification* parameter shall characterize the manner in which a node is prepared to receive AMS messages on a given subject. The delivery specification shall comprise a *Quality of Service (QoS) specification* and optionally a list of one or more explicit *delivery point specifications*. The QoS specification shall indicate (a) the *priority* at which messages on this subject must be issued (Express, Standard, Bulk) and (b) the *diligence* with which delivery of messages on this subject must be attempted (Assured – nominally implying acknowledgement and retransmission as necessary – or Best-effort). In the absence of an explicit delivery point specification list, an implied delivery point specification list shall be inferred, based on the QoS specification together with the standard delivery preferences noted in the node's AMS MIB. If noted explicitly, delivery point specifications shall be listed in declining order of preference; that is, the delivery point on which the node 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.

3.1.2.9 The *context* parameter shall be a number that identifies the application context in which a message was sent, if any. The significance and interpretation of this number are implementation matters. Conceptually, the context parameter functions within AMS as a message identifier. The context in which a reply is sent must be the additive inverse of the context in which the reply's antecedent message was sent; the context in which any non-reply message is sent must be non-negative.

3.1.2.10 The *content length* parameter shall indicate the length (in octets) of the *content* parameter.

Formatted: Bullets and Numbering

Deleted: <#>The *node specifiction* parameter shall be an ASCII text string that characterizes the manner in which a node is prepared to receive AMS messages. The node specification shall comprise a comma-separated list of one or more *port specifications* in declining order of preference; that is, the port on which the node most prefers to receive messages is specified first, followed by the next-most-preferred port, and so on. Each port specification shall be the concatenation of a *transport service name*, an "equals" (=) symbol, and a *transport service endpoint specification*, in that order. 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. The syntax in which the transport service endpoint specification is represented shall be specific to the indicated transport service. Definitions of valid endpoint specification syntax for all recognized transport services are given in Annex C.¶

Deleted: The *subject format* parameter shall provide information enabling an application to parse the application data in any message of a given subject

Deleted: an original (non-reply)

3.1.2.11 The *content* parameter shall be an array of zero or more octets comprising the application data in a message, in a format that is suitable for processing by the application.

3.1.2.12 The *handling instructions* parameter shall indicate the manner in which delivery of a message is to be regulated. Handling instructions shall indicate (a) whether or not the message sender is to be notified when delivery failure is confirmed, (b) whether or not the message sender is to be notified when delivery success is confirmed, and (c) optionally the *deadline* by which delivery must be accomplished in order to be considered successful.

3.1.2.13 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".

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

3.1.2.15 The *inferred QoS specification* parameter shall indicate the QoS (priority and diligence) with which the node has requested that messages on this subject be issued to it.

3.1.3 Register.request

3.1.3.1 Function

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

3.1.3.2 Semantics

Register.request shall provide parameters as follows:

Register.request (application name,
authority name,
zone name,
node name,
[MAAP specification])

3.1.3.3 When Generated

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

3.1.3.4 Effect on Receipt

Receipt of **Register.request** shall, if approved, cause AMS to add the node to the indicated message space zone.

3.1.3.5 Additional Comments

If MAAP specification is omitted, a default MAAP specification based on preferences noted in the AMS MIB shall be computed.

Deleted: None

3.1.4 Unregister.request

3.1.4.1 Function

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

3.1.4.2 Semantics

Unregister.request shall provide parameters as follows:

Unregister.request (node ID)

3.1.4.3 When Generated

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

3.1.4.4 Effect on Receipt

Receipt of **Unregister.request** shall cause AMS to remove the node from the indicated message space zone.

3.1.4.5 Additional Comments

The node ID provided in the **Unregister.request** primitive must be the one that was provided in the **Register.indication** primitive that notified the node of its own successful registration[4].

[4] The manner in which AMS service indications are linked to the AMS service requests to which they are directly responsive, where applicable, is an implementation matter.

3.1.5 Assert_invitation.request

Formatted: Bullets and Numbering

3.1.5.1 Function

The **Assert_invitation.request** primitive shall be used by the node to indicate the manner in which private messages on a specific subject may be delivered to the node.

Formatted: Bullets and Numbering

3.1.5.2 Semantics

Assert_invitation.request shall provide parameters as follows:

Assert_invitation.request (subject name,
delivery specification)

Formatted: Bullets and Numbering

3.1.5.3 When Generated

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

Formatted: Bullets and Numbering

3.1.5.4 Effect on Receipt

Receipt of **Assert_invitation.request** shall, if approved, cause AMS to notify all nodes in the message space of the node's willingness to accept private messages on the indicated subject.

Formatted: Bullets and Numbering

3.1.5.5 Additional Comments

Note that invitations are distinct from subscriptions: they enable private transmission of messages (*send*, *reply*, *query*) on the indicated subject and **do not** mandate delivery of a copy of each published message on this subject.

3.1.6 Cancel_invitation.request

Formatted: Bullets and Numbering

3.1.6.1 Function

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

Formatted: Bullets and Numbering

3.1.6.2 Semantics

Cancel_invitation.request shall provide parameters as follows:

Cancel_invitation.request (subject name)

Formatted: Bullets and Numbering

3.1.6.3 When Generated

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

3.1.6.4 Effect on Receipt

Receipt of **Cancel_invitation.request** shall cause AMS to notify all nodes in the message space that the node is no longer willing to accept private messages on the indicated subject.

3.1.6.5 Additional Comments

None.

3.1.7 Assert_subscription.request

3.1.7.1 Function

The **Assert_subscription.request** primitive shall be used by the node 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 (subject name, delivery specification)

3.1.7.3 When Generated

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

3.1.7.4 Effect on Receipt

Receipt of **Assert_subscription.request** shall, if approved, cause AMS to notify all nodes in the message space of the node's subscription to the indicated message subject.

3.1.7.5 Additional Comments

None.

3.1.8 Cancel_subscription.request

3.1.8.1 Function

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

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Deleted: Assert_subscription.request (subject name)¶

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

3.1.8.2 Semantics

Cancel_subscription.request shall provide parameters as follows:

Cancel_subscription.request (subject name)

3.1.8.3 When Generated

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

3.1.8.4 Effect on Receipt

Receipt of **Cancel_subscription.request** shall cause AMS to notify all nodes in the message space that the node is no longer subscribed to the indicated message subject.

3.1.8.5 Additional Comments

None.

3.1.9 Publish.request

3.1.9.1 Function

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

3.1.9.2 Semantics

Publish.request shall provide parameters as follows:

Publish.request (subject name,
content length,
[content],
[handling instructions],
[context])

3.1.9.3 When Generated

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

3.1.9.4 Effect on Receipt

Receipt of **Publish.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the content as necessary) and send one copy of that message to every node in the message space that is currently subscribed to the indicated message subject.

3.1.9.5 Additional Comments

Formatted: Bullets and Numbering

Context, if specified, identifies context information that is meaningful to the publishing node.

3.1.10 Send.request

Formatted: Bullets and Numbering

3.1.10.1 Function

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

Formatted: Bullets and Numbering

3.1.10.2 Semantics

Send.request shall provide parameters as follows:

Send.request (node ID,
subject name,
content length,
[content],
[handling instructions],
[context])

Formatted: Bullets and Numbering

3.1.10.3 When Generated

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

Formatted: Bullets and Numbering

3.1.10.4 Effect on Receipt

Receipt of **Send.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the content as necessary) and send it to the specified node.

Formatted: Bullets and Numbering

3.1.10.5 Additional Comments

Node ID identifies the node to which the message is to be sent. Context, if specified, identifies context information that is meaningful to the sending node.

3.1.11 Query.request

Formatted: Bullets and Numbering

3.1.11.1 Function

The **Query.request** primitive shall be used by the node to send a message privately to a single node in a synchronous fashion.

3.1.11.2 Semantics

Query.request shall provide parameters as follows:

Query.request (node ID,
subject name,
content length,
[content],
[handling instructions],
[term],
[context])

3.1.11.3 When Generated

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

3.1.11.4 Effect on Receipt

Receipt of **Query.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the content as necessary), send it to the specified node, and suspend the operation of the querying node until either a reply to this message is received, a notice of delivery failure for this message is received, or – if *term* is specified – the indicated term has expired.

3.1.11.5 Additional Comments

Node ID identifies the node to which the message is to be sent. Context, if specified, identifies context information that is meaningful to the querying node. If term is omitted, **only** reception of a reply or of a notice of delivery failure will cause the operation of the querying node to be resumed.

3.1.12 Reply.request

3.1.12.1 Function

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

3.1.12.2 Semantics

Reply.request shall provide parameters as follows:

Reply.request (node ID,
subject name,
content length,

[content],
context)

3.1.12.3 When Generated

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

3.1.12.4 Effect on Receipt

Receipt of **Reply.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the content as necessary) and send it to the specified node.

3.1.12.5 Additional Comments

Node ID must identify the node that sent some previously received message, and context must be the context provided with that message (which will be meaningful only to that node).

3.1.13 Message.indication

3.1.13.1 Function

The **Message.indication** primitive shall be used to deliver AMS original (non-reply) message information to the node.

3.1.13.2 Semantics

Message.indication shall provide parameters as follows:

Message.indication (node ID,
subject name,
content length,
[content],
[context])

3.1.13.3 When Generated

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

3.1.13.4 Effect on Receipt

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

3.1.13.5 Additional Comments

Node ID identifies the node that sent or published the message.

Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Deleted: <#>Declare_subject.request¶
<#>Function¶
The **Declare_subject.request** primitive shall be used by the node to declare the validity of a specified message subject.¶
<#>Semantics¶
Declare_subject.request shall provide parameters as follows:¶
Declare_subject.request (subject name, . [subject format])¶
<#>When Generated¶
Declare_subject.request is generated by the node at any time while the node is participating in its message space.¶
<#>Effect on Receipt¶
Receipt of **Declare_subject.request** shall cause AMS to recognize the validity of the indicated subject and to note the format in which the content of every message of this subject is represented (if specified). If the subject was previously declared, the new subject format (if specified) supersedes the subject's current format.¶
<#>Additional Comments¶
AMS message content formats are expected to be generally static. The AMS protocol does not include provisions for actively propagating revised formats to nodes that might previously have cached older formats, so cache coherency failures are possible. Future upgrades to AMS to redress this deficiency may eventually prove necessary.¶
<#>Look_up_subject.request¶
<#>Function¶
The **Look_up_subject.request** primitive shall be used by the node to determine the validity of a specified message subject and the format in which the content of every message of this subject is represented (if defined).¶
<#>Semantics¶
Look_up_subject.request shall provide parameters as follows:¶
Loop_up_subject.request (subject name)¶
<#>When Generated¶
Look_up_subject.request is generated by the node at any time while the node is participating in its mes ... [1]
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering

3.1.14 Handling.indication

Formatted: Bullets and Numbering

3.1.14.1 Function

The **Handling.indication** primitive shall be used to notify the node of message delivery success or failure per handling instructions specified when the message was issued.

Formatted: Bullets and Numbering

3.1.14.2 Semantics

Handling.indication shall provide parameters as follows:

Handling.indication (context,
[fault expression])

Formatted: Bullets and Numbering

3.1.14.3 When Generated

Handling.indication is generated upon confirmation that:

Formatted: Bullets and Numbering

- delivery has failed for a message for which the sending node's handling instructions stipulated that the sender is to be notified when delivery failure is confirmed, or
- delivery has succeeded for a message for which the sending node's handling instructions stipulated that the sender is to be notified when delivery success is confirmed.

3.1.14.4 Effect on Receipt

The effect on reception of **Handling.indication** by a node is undefined.

Formatted: Bullets and Numbering

3.1.14.5 Additional Comments

The context parameter in the **Handling.indication** primitive implicitly identifies the affected message; if negative, the affected message was a reply. The absence of a fault expression in the primitive indicates delivery success; if a fault expression is present, that expression indicates the nature of the delivery failure (e.g., delivery deadline expired prior to delivery).

3.1.15 Reply.indication

Formatted: Bullets and Numbering

3.1.15.1 Function

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

3.1.15.2 Semantics

Reply.indication shall provide parameters as follows:

Reply.indication (node ID,
subject name,
content length,
[content],
context)

3.1.15.3 When Generated

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

3.1.15.4 Effect on Receipt

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

3.1.15.5 Additional Comments

Node ID identifies the node that sent the reply message. Context is the additive inverse of the context in which the antecedent message (the message to which the reply message is a response) was sent.

3.1.16 Fault.indication

3.1.16.1 Function

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

3.1.16.2 Semantics

Fault.indication shall provide parameters as follows:

Fault.indication (fault expression)

3.1.16.3 When Generated

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

3.1.16.4 Effect on Receipt

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

3.1.16.5 Additional Comments

None.

3.1.17 Register.indication

Formatted: Bullets and Numbering

3.1.17.1 Function

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

Formatted: Bullets and Numbering

3.1.17.2 Semantics

Register.indication shall provide parameters as follows:

Register.indication (node ID,
node name)

Formatted: Bullets and Numbering

3.1.17.3 When Generated

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

Formatted: Bullets and Numbering

3.1.17.4 Effect on Receipt

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

Formatted: Bullets and Numbering

3.1.17.5 Additional Comments

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

3.1.18 Unregister.indication

Formatted: Bullets and Numbering

3.1.18.1 Function

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

Formatted: Bullets and Numbering

3.1.18.2 Semantics

Unregister.indication shall provide parameters as follows:

Unregister.indication (node ID)

Formatted: Bullets and Numbering

3.1.18.3 When Generated

Unregister.indication is optionally generated upon removal of any node from the message space.

Deleted: always generated upon removal of the node from its message space. **Unregister.indication** may optionally also be

Deleted: other

3.1.18.4 Effect on Receipt

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

3.1.18.5 Additional Comments

None.

3.1.19 Assert_invitation.indication

3.1.19.1 Function

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

3.1.19.2 Semantics

Assert_invitation.indication shall provide parameters as follows:

Assert_invitation.indication (node ID,
node name,
subject name,
inferred QoS specification)

3.1.19.3 When Generated

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

3.1.19.4 Effect on Receipt

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

3.1.19.5 Additional Comments

This indication is provided solely to facilitate message space configuration monitoring. Compliance with QoS specifications upon message transmission is the responsibility of AMS itself, not the node, so generation of this indication is strictly optional. The information provided in the indication may be used to help manage the node's understanding of the functional "connections" between itself and other nodes.

3.1.20 Cancel_invitation.indication

3.1.20.1 Function

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

3.1.20.2 Semantics

Cancel_invitation.indication shall provide parameters as follows:

Cancel_invitation.indication (node ID,
node name,
subject name)

3.1.20.3 When Generated

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

3.1.20.4 Effect on Receipt

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

3.1.20.5 Additional Comments

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

3.1.21 Assert_subscription.indication

3.1.21.1 Function

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

3.1.21.2 Semantics

Assert_subscription.indication shall provide parameters as follows:

Assert_subscription.indication (node ID,
node name,
subject name,
inferred QoS specification)

3.1.21.3 When Generated

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

3.1.21.4 Effect on Receipt

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

3.1.21.5 Additional Comments

This indication is provided solely to facilitate message space configuration monitoring. Message publication is accomplished by AMS itself, not by the node, so generation of this indication is strictly optional. The information provided in the indication may be used to help manage the node's understanding of the functional "connections" between itself and other nodes.

3.1.22 Cancel_subscription.indication

3.1.22.1 Function

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

3.1.22.2 Semantics

Cancel_subscription.indication shall provide parameters as follows:

Cancel_subscription.indication (node ID,
node name,
subject name)

3.1.22.3 When Generated

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

3.1.22.4 Effect on Receipt

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

3.1.22.5 Additional Comments

This indication is provided solely to facilitate message space configuration monitoring. Message publication is accomplished by AMS itself, not by the node, so generation of this indication is strictly optional. The information provided in the indication may be used to help manage the node's understanding of the functional "connections" between itself and other nodes.

3.2 Services required of the transport system

TBD.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Deleted: <#>Subject.indication¶
<#>Function¶
The **Subject.indication** primitive shall be used to report on the validity and (if specified) defined message format of a subject.¶
<#>Semantics¶
Subject.indication shall provide parameters as follows:¶
Subject.indication (subject name, . [subject format])¶
<#>When Generated¶
Subject.indication is generated upon reception of a subject report message from the subject server of the message space, which in turn is produced in response to a **Declare_subject.request** or **Look_up_subject.request** primitive.¶
<#>Effect on Receipt¶
The effect on reception of **Subject.indication** by a node is undefined.¶
<#>Additional Comments¶
None.¶

3.1.10Declare_subject.request

3.1.10.1Function

The Declare_subject.request primitive shall be used by the node to declare the validity of a specified message subject.

3.1.10.2Semantics

Declare_subject.request shall provide parameters as follows:

Declare_subject.request (subject name,
[subject format])

3.1.10.3When Generated

Declare_subject.request is generated by the node at any time while the node is participating in its message space.

3.1.10.4Effect on Receipt

Receipt of Declare_subject.request shall cause AMS to recognize the validity of the indicated subject and to note the format in which the content of every message of this subject is represented (if specified). If the subject was previously declared, the new subject format (if specified) supersedes the subject's current format.

3.1.10.5Additional Comments

AMS message content formats are expected to be generally static. The AMS protocol does not include provisions for actively propagating revised formats to nodes that might previously have cached older formats, so cache coherency failures are possible. Future upgrades to AMS to redress this deficiency may eventually prove necessary.

3.1.11Look_up_subject.request

3.1.11.1Function

The Look_up_subject.request primitive shall be used by the node to determine the validity of a specified message subject and the format in which the content of every message of this subject is represented (if defined).

3.1.11.2Semantics

Look_up_subject.request shall provide parameters as follows:

Loop_up_subject.request (subject name)

3.1.11.3When Generated

Look_up_subject.request is generated by the node at any time while the node is participating in its message space.

3.1.11.4Effect on Receipt

Receipt of Look_up_subject.request shall cause AMS to validate the indicated subject and report on the format in which the content of every message of this subject is represented (if defined).

3.1.11.5Additional Comments

None.