The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

MISSION OPERATIONS—MISSION DATA PRODUCT DISTRIBUTION SERVICES

DRAFT RECOMMENDED STANDARD

CCSDS 522.2-R-1

RED BOOK

November 2018

The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

MISSION OPERATIONS— MISSION DATA PRODUCT DISTRIBUTION SERVICES

DRAFT RECOMMENDED STANDARD

CCSDS 522.2-R-1

RED BOOK

November 2018

AUTHORITY

Issue:	Red Book, Issue 1
Date:	November 2018
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the e-mail address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
E-mail: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

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

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

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

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

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

FOREWORD

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 *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

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

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 522.2-R-1	Mission Operations—Mission Data Product Distribution Services, Draft Recommended Standard, Issue 1	November 2018	Current draft

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 GENERAL

This Recommended Standard defines the Mission Operations (MO) Mission Data Product Distribution Services (MDPDS) in conformance with the service framework specified in reference [C1], *Mission Operations Services Concept.*

The MDPDS are a set of services of which the main aim is to provide controlled access to mission data for the community of users who do not have access to mission-control-system monitoring and control facilities. It uses a general concept of ‘product’ and the methods of ‘batch’ and ‘stream’ product distribution to allow delivery of any type and format of data to authorized users.

The MDPDS are defined in terms of the Message Abstraction Layer (MAL) (reference [1]) and utilize Common Object Model (COM) (reference [2]) Archive and Event services.

1.2 PURPOSE AND SCOPE

The Recommended Standard defines the MDPDS in terms of

a) the key requirements of the services;

b) the operations necessary to provide the services;

c) the data types required by the service;

d) the expected behavior of each operation;

e) the errors, which may occur during the operation execution;

f) the structure of basic products; and

g) the use of COM Archive, Events, and Objects.

It does not specify

a) the design of the implementation of the services;

b) the physical location of system components;

c) the programming languages or technologies needed for the implementation of services;

d) the communication technologies;

e) the means of cooperation with external systems to obtain mission data products for the distribution; or

f) the configuration steps needed for setting up the MDPDS.

1.3 APPLICABILITY

This specification is applicable to any system component that exchanges mission data products at the application layer. Nominally, but not exclusively, this applies to ground control systems to external entities and between external entities.

1.4 RATIONALE

The goals of this Recommended Standard are to increase the degree of portability and interoperability and to reduce the time and costs required to access space mission data. To this end, the document provides a standard service specification for online, controlled access to space mission data for the user community (e.g., for the members of the scientific community).

1.5 DOCUMENT STRUCTURE

This document is structured as follows:

a) Section 1 contains purpose and scope, applicability, rationale of the document, nomenclature, and lists definitions and referenced documents.

b) Section 2 provides an overview of the concepts.

c) Section 3 contains a specification of the MDPDS.

d) Section 4 is a formal specification of MDPDS data structures.

e) Section 5 is a formal specification of MDPDS errors.

f) Annex A contains an Implementation Conformance Statement proforma.

g) Annex B contains security, SANA, and patent considerations.

1.6 DEFINITIONS

batch mode: The provisioning mode used for the retrieval of the archived mission data.

catalogue of products: A registry containing the information related to the products supported by the MDPDS, by which the consumer can request products.

chunk: A portion of the mission data product.

chunk size: A size of the mission data product portion; in the case of product requests, a maximum data size, upon exceeding which the response is split in parts not bigger than this value.

delivery mode: One of the three methods of data delivery (a direct mode, a pull mode, or a push mode).

destination: A local or remote location where the product is to be delivered.

direct delivery: A delivery mode in which the distribution of the response data is guaranteed by the MDPDS. All response data are encapsulated into MAL messages and sent directly to the user, without any brokers or third-party distribution services such as the File Transfer Protocol (FTP).

mission data product (also referred as: **product):** A set of the space mission data available for the user community.

product attribute: One of the features of the product, along with its data type and allowed values. It can be used to filter for or sort by the mission data product.

product specification: A set of metadata attributes of a product of a given type. It is independent of the format used to encode the product. The specification has the form of an XML file, which holds all the attributes in a portable way, by the means of the MAL data type description language, separate from any encoding used by the product formats. For example, Mission Operations Monitor & Control Services (reference [4]) define a ParameterDefinition composite, and that definition in the form of a MAL XML file is considered to be a product specification.

product detail: A MAL composite (reference [1]) that contains information about the availability of a product in the specific data format, the provision mode, compression formats, and encryption formats.

product format: The data format used to represent an instance of the product. The format describes how to encode product attributes enumerated in the product specification.

product type: A distinct category of mission data, for example, parameter time series, action history, CCSDS space packets, navigation orbit files, and image files.

provision mode: One of the two methods of data retrieval (batch mode or stream mode).

pull delivery: A passive delivery mode in which the product response is stored locally by the MDPDS and is accessible for a direct download by a standard third-party transfer protocol (such as FTP) to authorized users.

push delivery: An active delivery mode in which the product response is sent to a remote destination by the user-specified transfer protocol.

schedule: An optional field of the product request indicating that the request has been timetabled for the execution within the specific time

stream mode: The provisioning mode in which the mission data is provided in small chunks as it becomes available to the consumers.

1.7 NOMENCLATURE

1.7.1 NORMATIVE TEXT

The following conventions apply for the normative specifications in this document:

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

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

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

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

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

1.7.2 INFORMATIVE TEXT

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

– Overview;

– Background;

– Rationale;

– Discussion.

1.8 CONVENTIONS

1.8.1 FIGURES

Unified Modelling Language (UML) modelling diagrams are used in this document. Reference [3] provides further information regarding diagrams types and their meaning.

1.8.2 TABLES

The format of tables presented throughout the document to illustrate MDPDS operations is described in reference [1]. Below is an excerpt from that document with a table format description.

Operation Identifier	<<Operation name>>	
Interaction Pattern	<<Interaction pattern name>>	
Pattern Sequence	Message	Body Type
<<Message direction>>	<<Message name>>	<<Message types*>>
...	...	...

The message direction denotes the direction of the message relative to the provider of the pattern and is either IN or OUT. That is, all messages directed towards the provider are IN messages, and all messages directed away from the provider are OUT messages. It is expected that message names match those defined in the Primitives section.

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

The Body Type column contains the list of types that make up the Body of a particular message. Zero to many types may be listed here and together define the body of the indicated message.

1.8.3 COM OBJECTS

The MDPDS use the COM Archive to store objects. All objects stored in the COM Archive must follow certain rules described in reference [2]. Each object stored in the COM Archive must have *object name*, *object number*, *object body type*, and *related* and *source object* fields. This document defines custom COM Objects that store either standard MAL types or MDPDS-defined composites. In the latter case, to allow easy distinguishing between both, MDPDS-defined composites have in their names suffix *Details*, whereas custom COM objects do not.

1.9 REFERENCES

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

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

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

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

[4] *Mission Operations Monitor & Control Services*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 522.1-B-1. Washington, D.C.: CCSDS, October 2017.

[5] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. STD 66. Reston, Virginia: ISOC, January 2005.

2 OVERVIEW

2.1 GENERAL

This document contains the specification of the MDPDS. The MDPDS provide access to space mission data. Figure 2-1 illustrates the set of standards in support of the Mission Operations Services Concepts. The MDPDS belong to the Service Specifications.

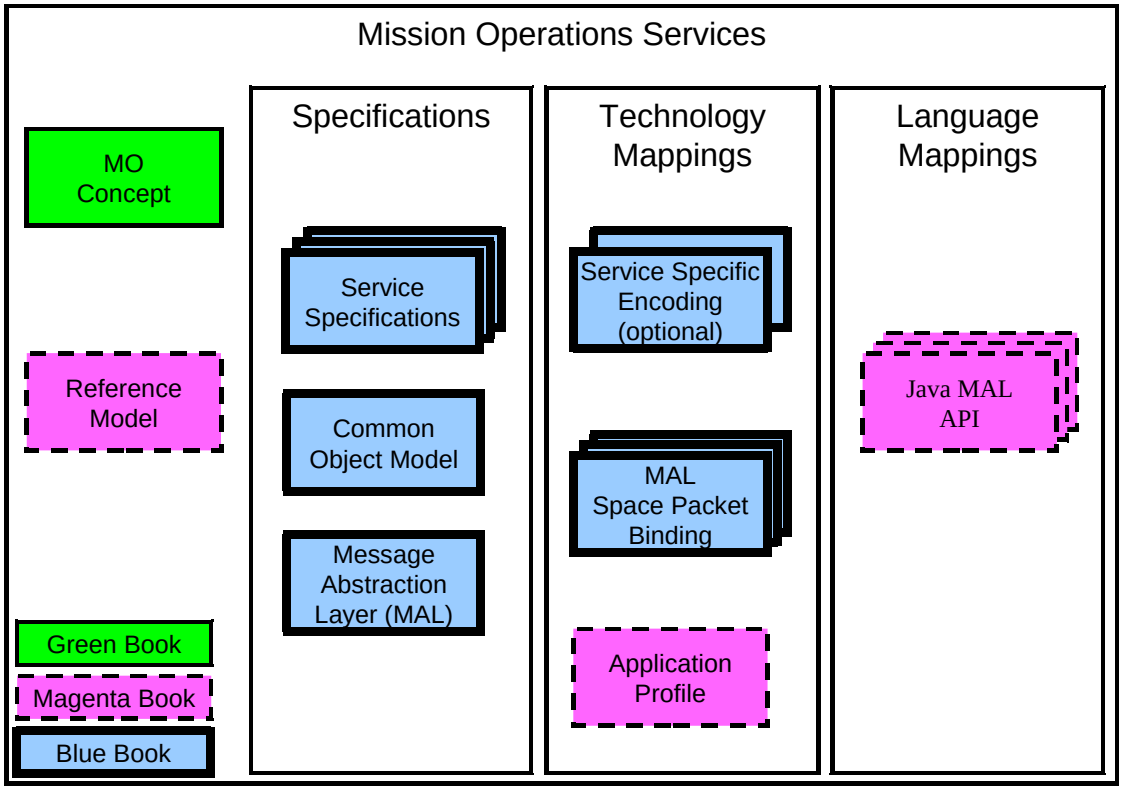

Figure 2-1: Mission Operations Services Concept Document Set

The MDPDS are in compliance with MO Service Framework layers (see reference [C1]) and are defined in terms of the MAL (reference [1]). Therefore it is possible to deploy them over any supported communication protocol and message transport technology.

2.2 PRODUCT CONCEPT

The ‘product’ concept has been introduced to abstract the structure, content, and format in which diverse space mission data products can be persisted, requested, and provisioned. This abstraction allows the specification of a generic set of services for managing, requesting, and provisioning space mission data products, without making assumptions about the implementations of the underlying mission data product distribution systems.

The MDPDS specify three elements, which in combination uniquely define each product. Those three elements consist of the product type, the product source, and the product format, as shown below:

– The product type represents a category of mission data to which the product belongs (e.g., parameter value evolution in a given time period, actions history, CCSDS space packet, navigation orbit files, image files). Each product type has a list of attributes associated with it. Attributes define the fields (by providing their names, data types, and usage) the product consists of. Some attributes can be selected to filter or sort products.

– The product source is a unique identifier, expressed in the form of a URI, which points unambiguously to the originator of the product. It indicates precisely what generated the product (e.g., the particular sensor of the particular satellite of the particular mission).

– The product format is a standardized description detailing how mission data products are encoded (e.g., a raw binary value, XML). The product format is used to specify how a product is encoded so that it may be decoded by a consumer; it does not provide the actual encoding, only that it is encoded in a specific format. For each predefined product type, a list of possible formats is assigned.

Elements of the triad are related to each other. Every product type has an assigned list of product sources, indicating origins of the product. A product format is one of the formats in which the MDPDS can distribute products. The product format is either the original format, in which the source created the product, or the format to which the MDPDS support a conversion.

2.3 CATALOGUES

In order to browse available products, the MDPDS have introduced the concept of the catalogue of products.

Catalogue entries contain information describing all available products and methods of their distribution. A catalogue entry always includes the product type, product source, and product format, as those elements uniquely identify each product. A catalogue entry also contains information regarding the formats in which MDPDS can provide a product of a particular type and source.

Catalogue entries are created in the following order:

a) The available product types are defined.

b) For each defined product type, the sources from which this product can originate are defined (the source is tightly bound to the type and cannot exist on its own).

c) Independently, a system-wide list of all formats supported by the MDPDS is prepared.

d) For each combination of product type and source, the set of formats in which that combination can be obtained is created.

Such a solution allows for multiple products of the same type, but they come from different sources and appear in different formats (e.g., the telemetry from one agency's spacecraft can have a different format from those of another's).

After the relationships between product type, source, and formats have been established, a new catalogue entry, containing that relation, is automatically created.

The organization of the catalogue means that the removal of a product source will result in the removal of the related product catalogue entries, and removal of a product type will result in the removal of not only the related product catalogue entries but also any sources assigned to that type.

The catalogue entry can also store additional information, which includes a list of product attributes that can be used for filtering and sorting purposes. Furthermore, the catalogue entry contains a provisioning mode, compression methods, encryption methods, and a description of each product.

Optionally, it is also possible to define an internal structure of the product, called the product specification. The product specification represents product attributes in a portable way, by means of the MAL service description language. Without adding the product specification, filtering or sorting response data of product requests is not possible. The MDPDS provide a set of standardized product specifications for a limited number of product types, which are typically involved in interoperable mission operation scenarios. For other product types, it is possible to add customized product specifications.

2.3.1 EXAMPLE

A space agency launches a mission METEO for monitoring weather on Earth. That mission consists of two spacecraft named OBSERVER1 and OBSERVER2. Both gather data needed to forecast the weather. Among their sensors there is a camera for taking high-definition images of Earth's atmosphere.

Possible product types for those spacecraft are

- actions that command spacecraft,
- housekeeping parameters, and
- mission-specific data, such as images.

Sources of those product types could be

- *agency.mcs.meteo.observer1* and *agency.mcs.meteo.observer2* for actions, so that actions for particular spacecraft for a given mission, operated by given mission control center and run by particular space agency, could be unambiguously distinguished;

- *agency.mcs.meteo.observer1.heating.status* for heating housekeeping parameters of OBSERVER1 and *agency.mcs.meteo.observer2.cmd.obq* for a list of commands in the on board queue of OBSERVER2; and
- *agency.mcs.meteo.observer1.sensors.camera* for images taken by OBSERVER1 and *agency.mcs.meteo.observer2.sensors.camera* for images taken by OBSERVER2.

In turn, formats of data could be

- CCSDS MISSION OPERATIONS MONITOR & CONTROL SERVICES Action format type for actions from both spacecraft;
- CCSDS MISSION OPERATIONS MONITOR & CONTROL SERVICES Parameter format for raw representation of all parameters from both spacecraft;
- Excel file with a report containing reads from all heating components of the OBSERVER1;
- PDF file with human-readable parameter representations of a list of commands in the onboard queue of OBSERVER2;
- RAW format for images taken by OBSERVER1; and
- RAW and JPEG formats for images taken by OBSERVER2.

2.4 MISSION DATA PRODUCT DISTRIBUTION SERVICES COMPOSITION

The MDPDS consist of two services: Mission Data Product Management and Mission Data Product Distribution. The purpose of each service is different, but they are related by a common concept. The relation is established by the catalogue of products.

The Mission Data Product Management service allows manipulation of the catalogue by providing means for defining available products. It can also be seen as the service responsible for the configuration of the mission data product distribution solution. The MDPDS allow obtaining a list of products that the user is interested in and retrieving the selected product. From another perspective, it can be perceived as a specification of the interface for retrieval of the information about the products and the products themselves.

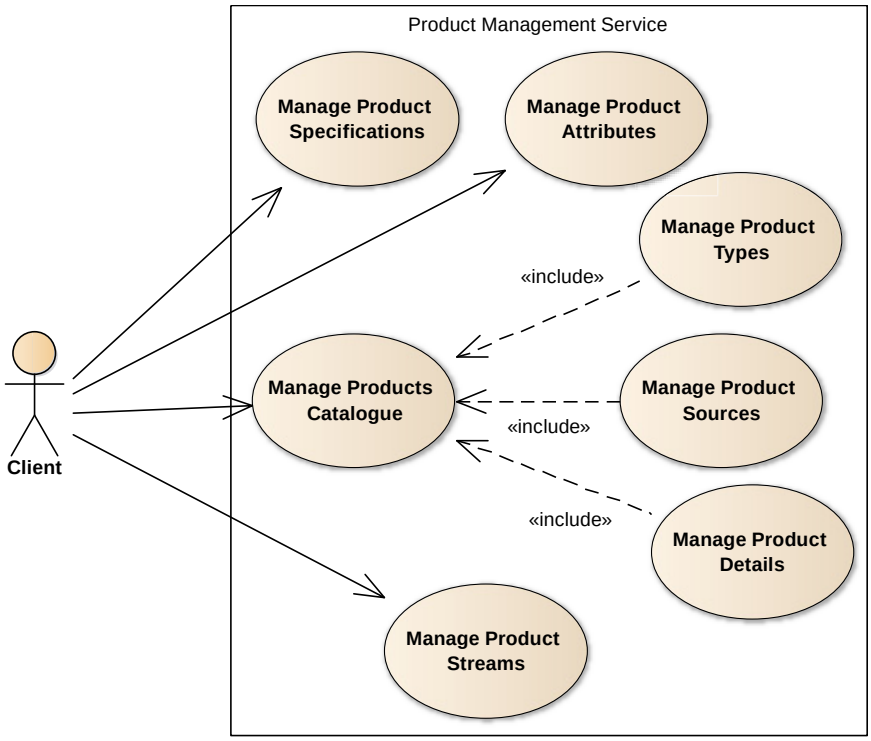

Figure 2-2: The Use Case Diagram for the Mission Data Product Management Service

The use case diagram in figure 2-2 presents an overview of the functionality provided by the Mission Data Product Management service. The main role of the service is to manage the catalogue of products. Each product, before it can be accessed, need to have been created in the catalogue. As mentioned in 2.2, the product catalogue entry contains the triad of the product type, source, and format along with additional details. Hence the management of the catalogue relies on defining catalogue entries containing all the information for new products and removing entries for unavailable products.

The product becomes available when it has been defined in the catalogue. Providing product specification is recommended, but not required. To satisfy this recommendation, the Mission Data Product Management service allows setting the specification of each product. Moreover, some basic products, like parameter timeline or actions history, are predefined. Attributes listed in the product specification can be configured to be used to filter or to sort the requested product. Furthermore, filtering or sorting of the data forming the response on the product request is only possible when the product specification is provided.

Another aspect of the service is the possibility of defining product streams. The MDPDS provide products in batch- and stream-provisioning modes. To allow provisioning of products in the latter mode, the stream needs to have been created in the Mission Data Product Management service first. The service allows for the creation of streams that transmit only instances of products that have the specified type, come from the selected source, are in the chosen format, and, optionally, match certain user-specified criteria.

The model in figure 2-3 illustrates the capabilities offered by the Mission Data Product Distribution service. The service allows the retrieval of the product catalogue, which has been defined in the Product Management service. It provides the functionality to query the catalogue by a given set of criteria to limit the number of catalogue entries to those in which the client may be interested.

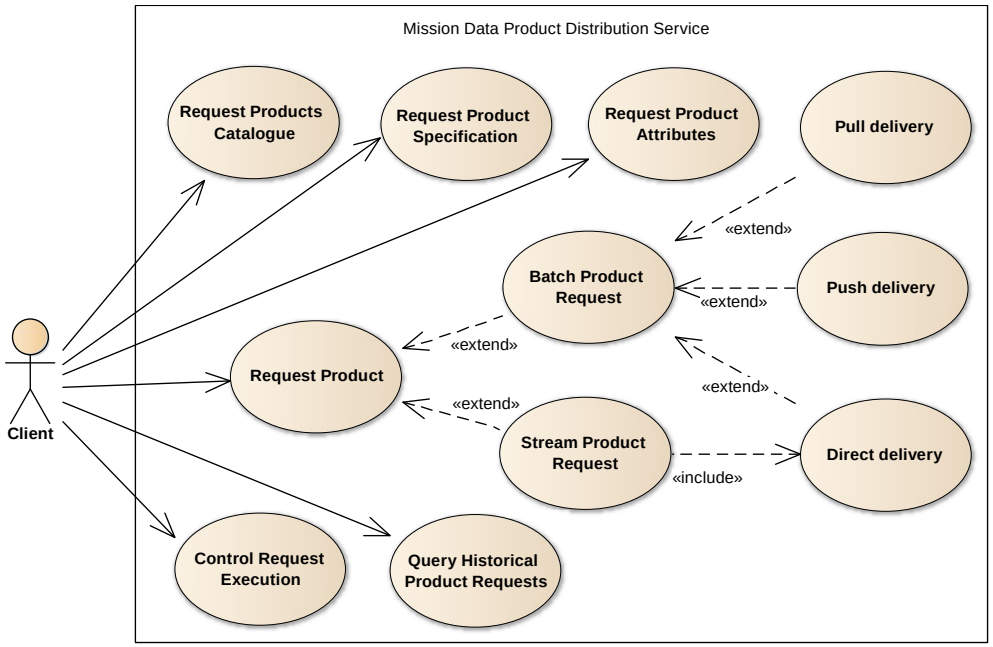

Figure 2-3: The Use Case Diagram for the Mission Data Product Distribution Service

The main goal of the service is to deliver the mission data products to the user that requested them. There are two provisioning modes, a batch mode and a stream mode. They differ in how they work and in the functionality they offer. The batch mode enables retrieval of the historical data in a collection, as well as the future data by scheduled periodic polling to check for the availability of new data. On the other hand, the stream mode provides the data as it becomes available and does so for as long as the user is interested in receiving updates. Both provisioning modes enable limiting the number of the products to retrieve by specifying filter criteria.

In the batch mode, the mission data product can be provided in one of three ways: delivered directly by the MDPDS to the requester; delivered to the remote destination, using a third-party transfer protocol (a push delivery); or stored by the MDPDS locally and made accessible to download to the authenticated users (a pull delivery). In the case of the stream mode, the MDPDS always deliver the response directly.

The service allows the client to obtain a list of attributes that can be used to compose a filter. The specification of the product can also be downloaded. The execution of each request may be cancelled at any moment. Additionally, batch requests can be suspended and resumed on demand. The service maintains the list of historical product requests and provides the ability to browse them.

2.5 MISSION DATA PRODUCT MANAGEMENT SERVICE

The Mission Data Product Management service provides the capabilities to manage the catalogue of the products on the service provider side. It is meant to be used by administrators, who are the only consumers of the service and who are given a convenient interface to modify catalogue entries, product specifications, and product attributes (user authentication and authorization, if required, are assumed to be in place but outside of the scope of this document). It can also be used to create a stream of product updates. Storage of any data generated in the service is delegated to the COM archive.

The process of creating a product needs to be conducted in the proper order, which is illustrated in figure 2-4 below.

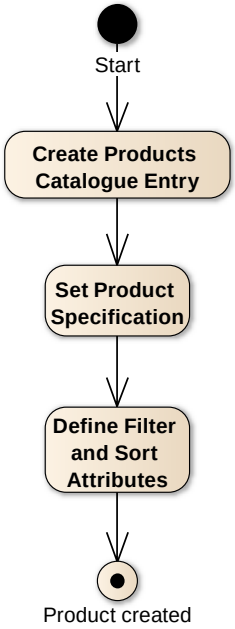

Figure 2-4: Actions to Create a New Product

The first step of the process is to create a catalogue entry. It is a complex activity that can be decomposed into a group of smaller steps, as shown in figure 2-5 below.

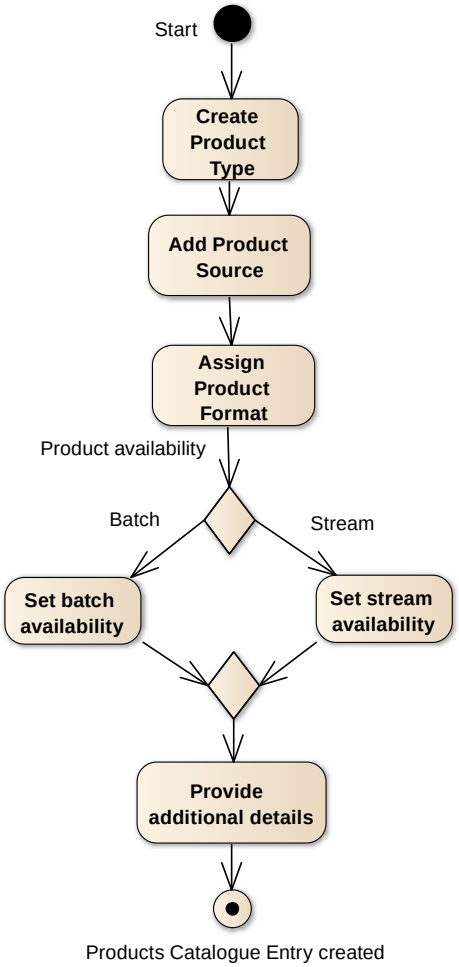

Figure 2-5: Actions to Create a Catalogue Entry

The creation of the catalogue entry begins with providing a type of a new product. Because the name of the product type may be not sufficient to explain what category of mission data the product belongs to, the consumer may additionally add a description. As type alone does not provide all information that describes the product, the next step is to specify all the sources from which the product is available. Then the only piece of information left to distinguish different products unambiguously is their data representation, which is expressed by a product format. Having defined the product type, the source, and the format, the availability in the batch or the stream provision mode has to be set. Optionally, in the last step, there could be specified compression and encryption methods applicable for the product. Finally, the catalogue entry is created and stored in the COM archive.

The creation of the catalogue entry is sufficient to make the product available. However, it is recommended that a formal description of the product be provided in the product specification. This is illustrated in step two in figure 2-4. Eventually, the selected attributes of the product specification may be configured to be used in product requests as ‘filter’ or ‘sort’ parameters. The COM archive is used to store provided information in this case as well.

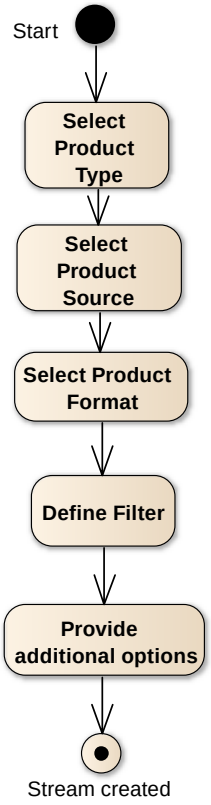

Figure 2-6: The Creation of the Product Stream

The catalogue keeps information indicating whether a particular product can be streamed. For products applicable for streaming, the Product Management service may be used to create their streams. Streams are created only when needed and need to be closed afterwards to save system resources. The flow of actions presented in figure 2-6 above illustrates how the stream is created. At first, the product type, the source, and the format are selected. As the client may be interested in obtaining only products that match certain criteria, a filter with a desired combination of chosen product attributes has to be provided. If required, additional stream options may be passed: for securing the stream, an encryption algorithm has to be set; for limiting the size of the data, a compression algorithm has to be identified. An option to close the stream automatically after a specified date has passed is also provided.

2.6 MISSION DATA PRODUCT DISTRIBUTION SERVICE

A Mission Data Product Distribution service provides capabilities for retrieving information about mission data products supported by the particular instance of the service, and for distribution of selected products. It presents the consumer with an interface to retrieve a list of available products, their specifications, and their attributes; to compose batch or stream product requests; to manage their state; and to check their execution status. The COM archive is responsible for storing the data that compose the product catalogue, product specifications, and historical requests.

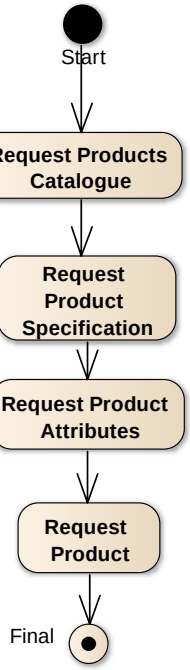

Figure 2-7: A Basic Flow of Actions Required to Request the Product

The diagram in figure 2-7 presents a typical flow of actions performed to obtain the product. The service allows the consumer to retrieve the information about the products, which are supported by a particular instance of the service provider, through requesting the catalogue of products of that provider instance. The returned list may contain a large number of catalogue entries. Therefore a request for catalogue entries can contain a set of filters to limit the response to contain only entries with desired product types, descriptions, product formats, product sources, compression and encryption methods, and provision modes. Once the product is decided upon, the user can also request its specification, a list of sort attributes, and a list of filter attributes.

The service allows the user to create either a batch or a stream request. Both require details to be provided, among which the most important are the product type, source, and format. Optionally, a list of filtering criteria can be specified, as well as encryption and compression algorithms to be applied to the response data.

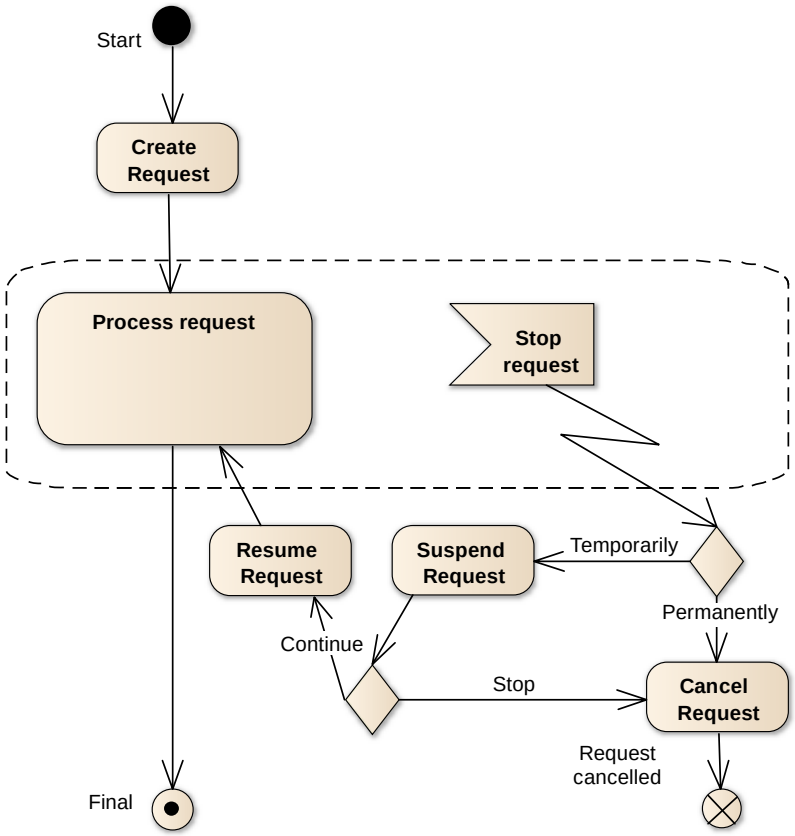

Figure 2-8: A Batch Product Request

Figure 2-8 presents a batch request, which can be handled immediately or scheduled for single or repeatable execution. The repeatable request is defined in terms of an execution date of the first request, an interval between subsequent executions, and the expiry date. When the user issues the repeatable request, the Mission Data Product Distribution service converts it to a list of sub-requests and assigns request identifiers for each of them, so they can be checked and controlled separately. To avoid scheduling an infinite number of sub-requests, the maximum number of sub-requests or the maximum period of time to schedule sub-requests needs to be defined during the implementation. The request can be cancelled or suspended at any time on demand and resumed from the same point within the data transmission. The current status of the request is maintained by the service and can be checked at any time.

In the batch mode, by default, the Mission Data Product Distribution service delivers products directly to the user. The service is self-standing and does not require third-party services or brokers to offer an end-to-end delivery of the products to ensure their integrity and consistency. The service guarantees that all products the user has requested will be delivered, or that alternatively, in case of failure, an appropriate error message will be returned. Moreover, the Mission Data Product Distribution service allows specifying different delivery modes. In a pull delivery the response data is stored locally by the service and made available for later download. Finally, there is also an option to deliver the response data to the remote destination (a push delivery). The response data is sent to the third-party system using the user-specified transfer protocol. In both of the last two cases the Mission Data Product Distribution service

returns to the user links that point to the response and that can be used to download the data directly. Those delivery methods are foreseen to provide the data to multiple users, with sufficient access privileges, interested in the same products. Those modes enable cooperation with legacy systems that expect to receive the data using a fixed transfer protocol like FTP. However, the Mission Data Product Distribution service guarantees neither that the user eventually downloads the response from the third-party system nor that the third-party system provides a reliable transfer.

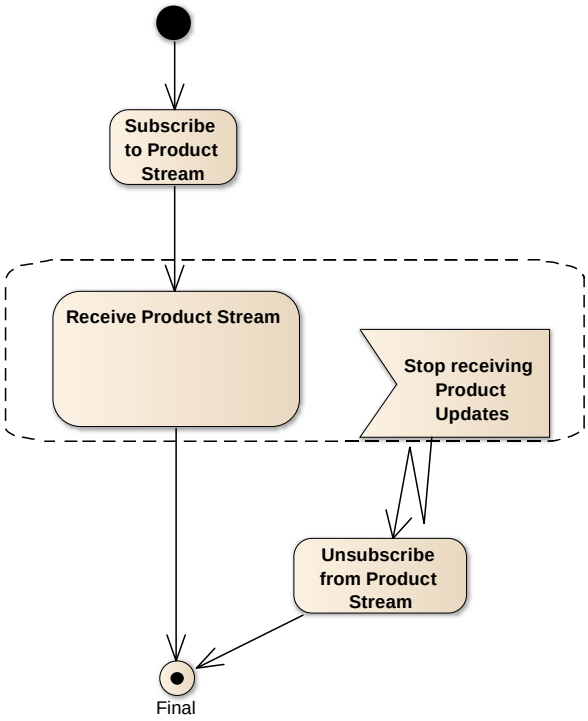

Figure 2-9: A Stream Product Request

Figure 2-9 illustrates a process of subscribing to the product stream. The stream containing instances of the requested product is already created in the Product Management service, so the only action left to execute is to subscribe to it to receive updates. Updates are sent until the stream is closed or expired, or until a signal with an unsubscribe message is sent. The stream can be closed at any time by invoking a proper operation in the Product Management service, upon which invocation all subscriptions become annulled and all subscribers stop receiving updates. The same situation happens when the stream expires; the expiry date is one of the parameters of the stream creation. Updates to the a given subscriber stop for that subscriber when an unsubscribe action is called.

In the stream mode the Mission Data Product Distribution service always sends products directly to the user, ensuring their integrity and consistency.

2.7 MDPDS AND COM SERVICES

Common Object Model services (reference [2]) play an important role in the architecture of the MDPDS.

The COM Archive provides a functionality to store all artifacts generated by the Mission Data Product Management service and holds historical requests made to the Mission Data Product Distribution service. Custom data types defined by MDPDS are stored in the COM Archive using COM objects. Every COM object has a member field called *Object body type*, which is a container for holding the data meant for archiving. When the MDPDS need to store an artifact in the COM archive, they create a new COM object and fill the *Object body type* container with the data to preserve. To distinguish between objects used by the MDPDS and the ones kept in the COM archive, the former have a *Details* suffix in their names.

The function of the COM Event service is to notify interested parties that some event in the system has just happened. The COM Event service works in the publish-subscribe manner; hence to receive notifications, the user first needs to subscribe for particular kinds of events. When the given event occurs, a notification is automatically distributed to all subscribers. The Mission Data Product Management service uses the COM Event service to generate events when a new artifact has been created or an existing one has been removed. The body of the event contains COM identifiers of the objects being created or removed. The related fields point to structures affected by either of two operations. The Mission Data Product Distribution creates events when a new product becomes available or when an existing product becomes unavailable. It uses a ‘related’ field to point to Product Catalogue Entries affected by the operation.

3 SPECIFICATION: MISSION DATA PRODUCT DISTRIBUTION SERVICES

3.1 OVERVIEW

3.1.1 RATIONALE

The MDPDS specification defines the means for distribution of mission data products. Therefore this section is intended to detail the description of the interface exposed by MDPDS, designated to fulfil this need. It enumerates the service requirements, specifies all the operations and data types required to manage and transfer products, and describes the patterns of the interaction between objects.

3.1.2 FUNCTIONAL REQUIREMENTS

This Recommended Standard specifies an interface to distribute mission data products, which meets the following requirements:

Title	Description
Retrieval of mission data products for a given time period.	The consumer of the MDPDS shall be able to request historic mission data products for a given time period, specifying: – product type; – product source; – product format; – encryption format (optional); – compression format (optional); – schedule (optional); – filter (optional); – sort field name (optional); – message chunk size (optional).
Batch Request	The MDPDS provider shall generate a complete mission data product that fulfills the product request and provide it to the consumer in a finite number of chunks. The number of chunks shall depend on the maximum size of the chunk specified by the consumer. The MDPDS provider shall allow distribution of the mission data products over a transport protocol and delivery of the data to the remote destination; both the protocol and the destination shall be specified in the product request.
Request Identifiers	The MDPDS shall assign a unique ID to each request for a mission product. It shall report the status of the product delivery and the number of provisioned chunks after the last chunk of the mission data product has been provisioned.
Request scheduling	The MDPDS shall allow the consumer to schedule an execution of a request. The MDPDS shall support at least the following scheduling attributes: – begin time in relation to UTC time zone; – interval; – repeatability; – expiry time in relation to UTC time zone.

Title	Description
Stream Request	The MDPDS provider shall generate a stream of mission data product instances that fulfill the product request, and shall provide them to the consumer in a stream of messages. Once the new mission data product becomes available, the provider shall immediately send it to all subscribed consumers. The MDPDS shall assign a unique ID to each product stream.
Query Product Requests	The MDPDS provider shall allow generation of a list of historical requests of mission data products. The MDPDS shall allow the consumer to filter requests for mission data products by: – time range; – status; – account name.
Control Request Execution	The MDPDS shall allow the consumer to suspend a request that has not been completed. The MDPDS shall allow the consumer to resume a request. The MDPDS shall allow the consumer to terminate a request that has not been completed.
Retrieval of Catalogue of Products	The MDPDS shall be able to request a list of available products and the parameters of the request.
Modification of catalogue of products	The MDPDS shall allow modification of a catalogue of products.
Assignment and removal of the specification of the product	The MDPDS consumer shall be able to assign a product specification to every product. The MDPDS consumer shall be able to remove a product specification.
Retrieval of Product Attributes	The MDPDS shall be able to request a list of product attributes.
Modification of the list of product types	The MDPDS consumer shall be able to add a product type by specifying a self-descriptive name and (optionally) a description. The MDPDS consumer shall be able to remove a product type.
Modification of the list of sources	The MDPDS consumer shall be able to add a source, specifying a related product type and a URI. The MDPDS consumer shall be able to remove a source.
Modification of the list of product formats	The MDPDS consumer shall be able to add a product format, specifying – product type; – source; – provisioning mode; – compression format; and – encryption format. The MDPDS consumer shall be able to remove a product format.
Modification of the list of product attributes	The MDPDS consumer shall be able to define a product attribute, which can be used by filtering or product-sorting operations. The MDPDS consumer shall be able to remove a product attribute.
Definition of standard products	The MDPDS shall define the structures of typical product types.
Definition of standard formats	The MDPDS shall define the typical formats of products.

3.2 SERVICE: MISSION DATA PRODUCT DISTRIBUTION

3.2.1 OVERVIEW

The Mission Data Product Distribution service provides operations to retrieve a catalogue of available products and to request selected products in either the batch or stream provisioning mode from one of configured data sources. It also provides mechanisms to check the progress of the data delivery and to control it by ‘suspend’, ‘resume’, and ‘cancel’ operations.

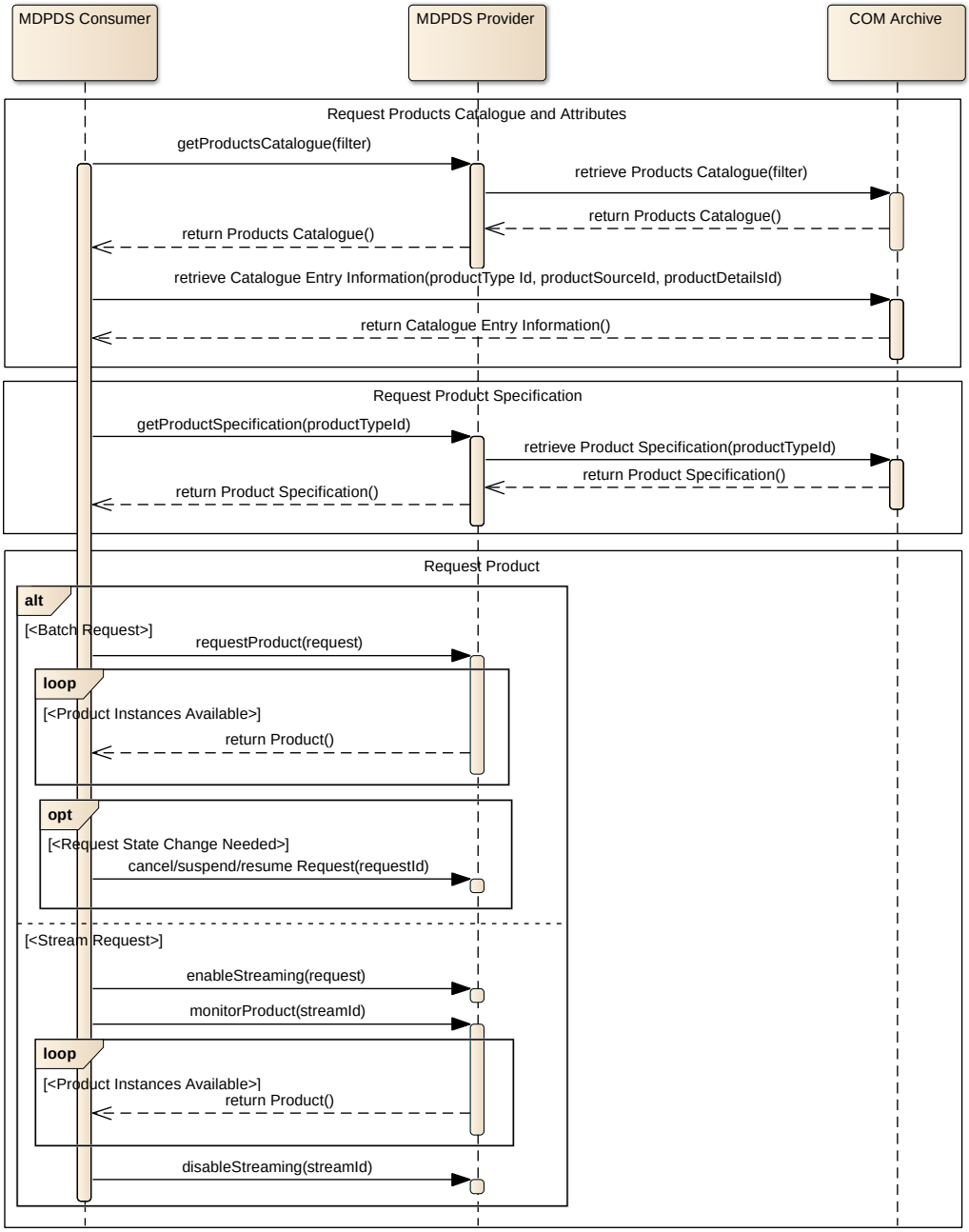

Figure 3-1: A Typical Use Case Scenario for Retrieval of the Mission Data Product

A typical use case scenario of the Mission Data Product Distribution service is shown in figure 3-1. By invoking the getProductsCatalogue() operation, the consumer retrieves from the provider a list of product catalogue entries in which the consumer may be interested. To limit the number of potential matches, criteria that the response must meet may be provided. This is annotated in the picture by a filter parameter of the getProductsCatalogue() operation. As the catalogue entries are actually stored in the COM archive, the provider passes the request to the archive and returns a filtered list of matching entries. Next, the consumer selects one of the products and retrieves, directly from the archive, its detailed information (the type, source, format, provisioning mode, and attributes). Through the provider, the consumer obtains the detailed specification of the product. For that purpose, the getProductSpecification() operation is executed. Its parameter is the type of the selected product. Finally, the consumer sends a customized product request to the provider in either batch or stream provisioning mode. The request specifies all the conditions the product to be retrieved must meet. In case of the batch mode, the product is obtained through the requestProduct() operation. All the matching product instances are received one by one in the loop. Optionally, the execution of the request can be cancelled, suspended, and resumed at any time. In turn, for the stream mode, the consumer first creates a stream containing instances of the specified product. The stream is created by the enableStreaming() operation of the Product Management service. Then the monitorProduct() operation is invoked by the consumer to subscribe to all messages from that stream and to receive upcoming updates. At the end, the stream is closed by the disableStreaming() operation of the Product Management service.

Table 3-1: Mission Data Product Distribution Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
distribution	MissionDataProductDistribution	9	1	1
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in Replay**	**Capability Set**
PROGRESS	requestProduct	1	No	1
SUBMIT	suspendRequest	2	No	
SUBMIT	resumeRequest	3	No	
SUBMIT	cancelRequest	4	No	
REQUEST	getRequestStatus	5	No	2
REQUEST	queryProductRequest	6	No	
PUBLISH-SUBSCRIBE	monitorProduct	7	No	3
REQUEST	getProductsCatalogue	8	No	4
REQUEST	getProductSpecification	9	No	

3.2.2 HIGH LEVEL REQUIREMENTS

3.2.2.1 The Mission Data Product Distribution service shall provide the capability for

a) requesting a product in either of two provisioning modes;

b) retrieving a status of request execution;

c) retrieving parameters of historic requests;

d) controlling the execution of a request;

e) retrieving the catalogue of products;

f) retrieving the specification of a product;

g) retrieving the attributes of a product;

h) storing the resulting product locally; and

i) sending the resulting product to the remote destination.

3.2.2.2 The list of definitions of typical product types that are supported by the Mission Data Product Distribution service shall be declared when deploying that service.

3.2.2.3 Each product request in the batch mode shall allow specifying

a) a product type;

b) a product format;

c) a product source;

d) a destination (optional);

e) a time range (optional) in relation to the UTC time zone;

f) a filter (optional);

g) a schedule (optional) in relation to the UTC time zone;

h) an encryption (optional);

i) a compression (optional);

j) a field to sort the product (optional); and

k) a chunk size (optional).

3.2.2.4 Each product request in the stream mode shall allow specifying

a) a product type;

b) a product format;

c) a product source;

d) a filter (optional);

e) an encryption (optional);

f) a compression (optional); and

g) expiry date (optional).

3.2.2.5 Each item in the catalogue of products shall contain

a) a product type;

b) a product source; and

c) a product provisioning detail.

3.2.2.6 The Mission Data Product Distribution service shall use the COM archive to store objects.

3.2.3 FUNCTIONAL REQUIREMENTS

If a product is not defined or an invalid provisioning mode is selected, the corresponding request shall be cancelled, and an error shall be raised.

3.2.4 COM USAGE

NOTE – All COM objects described in this section are custom MDPDS objects.

3.2.4.1 A BatchProductRequest COM object represents the batch product request. A BatchProductRequest COM object body shall hold the BatchProductRequestDetails structure.

3.2.4.2 The LoginInstance structure of the Login Service shall indicate a user who issued the request.

Table 3-2: Mission Data Product Distribution Service Object Types

Object Name	Object Number	Object Body Type	Related points to
BatchProductRequest	1	BatchProductRequestDetails	Login::LoginInstance

3.2.5 COM EVENT SERVICE USAGE

3.2.5.1 When a new mission data product for the selected type, source, and format is published, the NewProductAvailable event shall be generated.

3.2.5.2 The NewProductAvailable event shall use the related link to indicate the ProductCatalogueEntryDetails of the corresponding product.

3.2.5.3 When the product becomes unavailable, the ProductUnavailable event shall be generated.

3.2.5.4 The ProductUnavailable event shall use the related link to indicate the ProductCatalogueEntryDetails of the corresponding product.

Table 3-3: Mission Data Product Distribution Service Events

Event Name	Object Number	Object Body Type	Related points to	Source points to
NewProductAvailable	101	Not used	MissionDataProductManagement::ProductCatalogueEntry	
ProductUnavailable	102	Not used	MissionDataProductManagement::ProductCatalogueEntry	

3.2.6 DISCUSSION—COM OBJECT RELATIONSHIPS

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

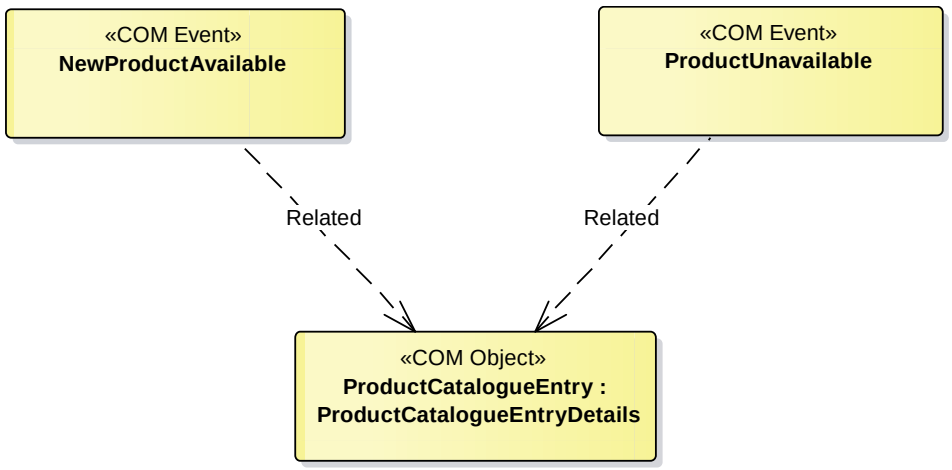

Figure 3-2: The Mission Data Product Distribution Service COM Object and Event Relationships

3.2.7 COM ARCHIVE SERVICE USAGE

BatchProductRequest objects shall be stored in the COM archive.

3.2.8 OPERATION: requestProduct

3.2.8.1 Overview

The requestProduct operation requests the mission data product to be delivered in the batch mode.

The PROGRESS interaction pattern is used as the returned set of data may be quite large, and this allows it to be split over several MAL messages. The matching product is returned in one or more update messages. The response message contains a summary of the transmitted data.

Operation Identifier	requestProduct	
Interaction Pattern	PROGRESS	
Pattern Sequence	Message	Body Type
IN	PROGRESS	request : (BatchProductRequestDetails)
OUT	ACK	requestId : (MAL::Long) subRequestIds : (List<MAL::Long>) productSize : (MAL::Long)
OUT	UPDATE	requestId : (MAL::Long) product : (MAL::Element)
OUT	RESPONSE	transferReport : (TransferReport)

3.2.8.2 Structures

3.2.8.2.1 The request field shall contain the BatchProductRequestDetails.

3.2.8.2.2 The productTypeId field of the supplied BatchProductRequestDetails structure shall match a Product Type object instance identifier in the COM archive; otherwise, an INVALID_PRODUCT_TYPE error shall be raised.

3.2.8.2.3 The productSourceId field of the supplied BatchProductRequestDetails structure shall match a Product Source object instance identifier in the COM archive; otherwise, an INVALID_PRODUCT_SOURCE error shall be raised.

3.2.8.2.4 The productFormatId field of the supplied BatchProductRequestDetails structure shall match a Product Format object instance identifier in the COM archive; otherwise, an INVALID_PRODUCT_FORMAT error shall be raised.

3.2.8.2.5 For the selected Product Type, Product Source, and Product Format, if the corresponding ProductCatalogueEntryDetails does not support the batch mode, a PRODUCT_NOT_SUPPORTED_IN_BATCH error shall be returned.

3.2.8.2.6 The destination field of the supplied BatchProductRequestDetails structure is optional and indicates a local or remote destination of the product delivery. If the destination is different from the default (the Mission Data Product Distribution service consumer), then it shall contain the DestinationDeliveryDetails structure. Otherwise, it shall be NULL.

3.2.8.2.7 If the destination field is not NULL, then the *uri* field of the DestinationDeliveryDetails structure shall contain the fully qualified address of the destination system in the form of the URI (reference [5]).

a) The *uri* shall specify the protocol used for the data transfer, the identifier of the destination system, and optional parameters, such as a desired filename used for saving the response.

b) A reserved keyword ‘localhost’ shall be used in the address part to indicate that the response data shall be stored locally by the Mission Data Product Distribution service.

c) If the specified filename is invalid, an INVALID_FILE_NAME error shall be returned.

3.2.8.2.8 If the destination field is not NULL, the *securityToken* field of the DestinationDeliveryDetails structure may contain authentication and authorization details of the user in the remote system.

3.2.8.2.9 If the destination field is not NULL, the *protocolAttributes* field of the DestinationDeliveryDetails structure may contain any attributes required by the transfer protocol.

3.2.8.2.10 If the *uri* is not in the form of the URI (reference [5]) or the destination does not exist, a DESTINATION_UNKNOWN error shall be returned.

3.2.8.2.11 If an invalid transfer protocol was specified, a TRANSFER_PROTOCOL_NOT_SUPPORTED error shall be returned.

3.2.8.2.12 If the *securityToken* field contains invalid authentication attributes, an AUTHENTICATION_FAIL error shall be returned.

3.2.8.2.13 If the authentication details are not sufficient to execute the requested operation, an AUTHORISATION_FAIL shall be returned.

3.2.8.2.14 If the *protocolAttributes* field contains invalid transfer protocol attributes, an INVALID_TRANSFER_PROTOCOL_ATTRIBUTES error shall be returned.

3.2.8.2.15 If the destination field is valid, but a connection cannot be established or a product cannot be delivered, a DELIVERY_FAILED error shall be returned.

3.2.8.2.16 If endTime is earlier than startTime, an INVALID_TIME_RANGE shall be returned.

3.2.8.2.17 If startTime is NULL, the request concerns the time range from the earliest available mission data product to the endTime value.

3.2.8.2.18 If endTime is NULL, the request concerns the time range from the startTime value to the current time.

3.2.8.2.19 If startTime or endTime value is higher than the current time, an INVALID_TIME_RANGE shall be returned.

3.2.8.2.20 The requestId field in the acknowledge message shall contain the object instance identifier of the request.

3.2.8.2.21 If the request is scheduled, the subRequestIds field shall contain a list of the object instance identifiers of the requests assigned for scheduled sub-requests.

3.2.8.2.22 The productSize field shall contain estimated size of the product response. If the size is unknown, it shall be NULL.

3.2.8.2.23 The requestId field in the update message shall contain the object instance identifier of the request related to the requested product.

3.2.8.2.24 The product field shall contain the matching product.

3.2.8.2.25 The product shall match all filter criteria.

NOTE – This forms a logical AND operation.

3.2.8.2.26 If there is no product matching filter criteria, update messages shall not be sent.

3.2.8.2.27 If the destination field of the BatchProductRequestDetails is used, the *product* field of the update message shall contain a RemoteResponse structure. The RemoteResponse shall contain a direct link to the product response, the size of the response, and the delivery date.

3.2.8.2.28 The response message shall contain the TransferReport.

3.2.8.3 Errors

The operation may return one of the following errors:

a) ERROR: DESTINATION_UNKNOWN: the host specified in the destination field is unreachable.

Error	Error #	ExtraInfo Type
DESTINATION_UNKNOWN	Defined in MAL	Not Used

b) ERROR: DELIVERY_FAILED: the product could not be delivered because of a communication error.

Error	Error #	ExtraInfo Type
DELIVERY_FAILED	Defined in MAL	Not Used

c) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

d) ERROR: AUTHENTICATION_FAIL: the authentication details provided in the securityToken field of the DestinationDeliveryDetails structure are invalid.

Error	Error #	ExtraInfo Type
AUTHENTICATION_FAIL	Defined in MAL	Not Used

e) ERROR: AUTHORISATION_FAIL: the user has no credentials to execute the requested operation on the remote system.

Error	Error #	ExtraInfo Type
AUTHORISATION_FAIL	Defined in MAL	Not Used

f) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

g) ERROR: COMPRESSION_FORMAT_NOT_SUPPORTED: the compression format for the selected product is invalid or not supported by the implementation.

Error	Error #	ExtraInfo Type
COMPRESSION_FORMAT_NOT_SUPPORTED	11	Not Used

h) ERROR: ENCRYPTION_ALGORITHM_NOT_SUPPORTED: the encryption algorithm for the selected product is invalid or not supported by the implementation.

Error	Error #	ExtraInfo Type
ENCRYPTION_ALGORITHM_NOT_SUPPORTED	12	Not Used

i) ERROR: SCHEDULING_NOT_SUPPORTED: the request execution scheduling is not supported by the implementation.

Error	Error #	ExtraInfo Type
SCHEDULING_NOT_SUPPORTED	13	Not Used

j) ERROR: SORTING_NOT_SUPPORTED: the sorting format for the selected product is invalid or not supported by the implementation.

Error	Error #	ExtraInfo Type
SORTING_NOT_SUPPORTED	14	Not Used

k) ERROR: INVALID_CHUNK_SIZE: the chunk size is invalid or splitting the product into update messages is not supported by the implementation.

Error	Error #	ExtraInfo Type
INVALID_CHUNK_SIZE	15	Not Used

l) ERROR: PRODUCT_NOT_SUPPORTED_IN_BATCH: the product is not supported in the batch mode for the selected combination of the Product Type, the Product Source, and the Product Format.

NOTE – The ExtraInfo field contains an instance identifier of the ProductCatalogueEntryDetails, which would provide available provision modes for the combination of the product type, the source, and the format.

Error	Error #	ExtraInfo Type
PRODUCT_NOT_SUPPORTED_IN_BATCH	16	MAL::Long

m) ERROR: TRANSFER_PROTOCOL_NOT_SUPPORTED: the transfer protocol specified in the request for the remote delivery is not supported by the provider.

Error	Error #	ExtraInfo Type
TRANSFER_PROTOCOL_NOT_SUPPORTED	18	Not Used

n) ERROR: INVALID_TRANSFER_PROTOCOL_ATTRIBUTES: the attributes of the transfer protocol for the remote delivery specified in the request are invalid.

Error	Error #	ExtraInfo Type
INVALID_TRANSFER_PROTOCOL_ATTRIBUTES	19	Not Used

o) ERROR: INVALID_PRODUCT_SOURCE: the product source is not valid for the specified product type.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SOURCE	2	Not Used

p) ERROR: INVALID_FILE_NAME: the file name specified for saving the response is invalid.

NOTE – It may be caused by an invalid character, an invalid path or, when using templates, an invalid special identifier.

Error	Error #	ExtraInfo Type
INVALID_FILE_NAME	20	Not Used

q) ERROR: INVALID_PRODUCT_FORMAT: the format is not supported for the selected combination of the product type and the source.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_FORMAT	3	Not Used

r) ERROR: INVALID_PRODUCT_ATTRIBUTE: one or more filter attributes are invalid.

NOTES

1 The attribute may not have been defined for the product, the selected comparison operation is not allowed, or the attribute value does not match the product specification.

2 The ExtraInfo field contains the first invalid attribute.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_ATTRIBUTE	6	MAL::Attribute

s) ERROR: INVALID_TIME_RANGE: the date or date format in timeRange is invalid.

Error	Error #	ExtraInfo Type
INVALID_TIME_RANGE	7	Not Used

t) ERROR: INVALID_SCHEDULE: the schedule data type contains an invalid date, invalid date format, or wrong expiry date.

NOTE – The ExtraInfo field contains the first invalid parameter.

Error	Error #	ExtraInfo Type
INVALID_SCHEDULE	8	MAL::Attribute

3.2.9 OPERATION: suspendRequest

3.2.9.1 Overview

The suspendRequest operation suspends the execution of the current or scheduled product request until a resumeRequest or cancelRequest operation is invoked.

Operation Identifier	suspendRequest	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	requestId : (MAL::Long)

3.2.9.2 Structures

3.2.9.2.1 The requestId field shall contain the object instance identifier of the request.

3.2.9.2.2 If the requestId field equals ‘0’, all non-completed requests are suspended.

3.2.9.2.3 If requestId does not exist, an INVALID_REQUEST_ID error shall be returned.

3.2.9.2.4 In case of the scheduled request, the main requestId may be used to suspend all sub-requests.

3.2.9.2.5 In case of the scheduled request, suspending one sub-request shall not affect other sub-requests.

3.2.9.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_REQUEST_ID: the request object instance identifier does not exist.

Error	Error #	ExtraInfo Type
INVALID_REQUEST_ID	8	Not Used

3.2.10 OPERATION: resumeRequest

3.2.10.1 Overview

The resumeRequest operation resumes the execution of the product request. The resume operation continues from the last delivered chunk of data.

Operation Identifier	resumeRequest	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	requestId : (MAL::Long)

3.2.10.2 Structures

3.2.10.2.1 The requestId field shall contain the object instance identifier of the request.

3.2.10.2.2 If requestId does not exist, an INVALID_REQUEST_ID error shall be returned.

3.2.10.2.3 If the requestId field equals ‘0’, all suspended requests shall be resumed.

3.2.10.2.4 In the case of the scheduled request, the main requestId may be used to resume all sub-requests.

3.2.10.2.5 In the case of the scheduled request, resuming one sub-request shall not influence other sub-requests.

3.2.10.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_REQUEST_ID: the request object instance identifier does not exist.

Error	Error #	ExtraInfo Type
INVALID_REQUEST_ID	8	Not Used

3.2.11 OPERATION: cancelRequest

3.2.11.1 Overview

The cancelRequest operation cancels the execution of the current or the scheduled request. No delivered data is lost.

Operation Identifier	cancelRequest	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	requestId : (MAL::Long)

3.2.11.2 Structures

3.2.11.2.1 The requestId field shall contain the object instance identifier of the request.

3.2.11.2.2 If requestId does not exist, an INVALID_REQUEST_ID error shall be returned.

3.2.11.2.3 If the requestId field equals ‘0’, all non-completed requests shall be cancelled.

3.2.11.2.4 In case of the scheduled request, the main requestId may be used to cancel all sub-requests.

3.2.11.2.5 In case of the scheduled request, cancelling one sub-request shall not influence other sub-requests.

3.2.11.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_REQUEST_ID: the request object identifier does not exist.

Error	Error #	ExtraInfo Type
INVALID_REQUEST_ID	8	Not Used

3.2.12 OPERATION: getRequestStatus

3.2.12.1 Overview

The getRequestStatus operation returns the current status of processing of a historical, an ongoing, or a scheduled request. The operation allows the user to filter request statuses.

Operation Identifier	getRequestStatus	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	filter : (COM::Archive::CompositeFilterSet)
OUT	RESPONSE	statuses : (List<RequestStatus>)

3.2.12.2 Structures

3.2.12.2.1 The filter field shall contain parameters to filter the current status of requests by a request identifier, a user identifier, a state, or a time range.

3.2.12.2.2 The filter may contain multiple criteria combined by a logical AND operator.

3.2.12.2.3 The filter field may contain comparison operators, ranges of values, and multiple wildcard characters.

3.2.12.2.4 If more than one filter parameter is used, the results shall be combined using logical conjunction.

3.2.12.2.5 The response shall contain a list of statuses of the requests, which match selected filter criteria.

3.2.12.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_ATTRIBUTE: one or more filter attributes are invalid.

NOTES

1 The attribute is invalid when its name or type is different from any field of the BatchProductRequestDetails structure, when the comparison operator is not valid for the type of filter field, or when the range is invalid.

2 The ExtraInfo field contains the first invalid attribute.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_ATTRIBUTE	6	MAL::Composite

c) ERROR: INVALID_REQUEST_ID: the filter contains an invalid object instance identifier of the BatchProductRequestDetails.

Error	Error #	ExtraInfo Type
INVALID_REQUEST_ID	8	Not Used

3.2.13 OPERATION: queryProductRequest

3.2.13.1 Overview

The queryProductRequest operation returns the content of requests that match filter criteria.

Operation Identifier	queryProductRequest	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	filter : (COM::Archive::CompositeFilterSet)
OUT	RESPONSE	requests : (List<BatchProductRequestDetails>)

3.2.13.2 Structures

3.2.13.2.1 The filter shall indicate conditions that the response to the request must meet. The response may be filtered by the request object instance identifier or any combination of the fields: the product type, the format, the source, the destination, the time range, the user, and the state.

3.2.13.2.2 The filter may contain comparison operators, ranges, and multiple wildcard characters.

3.2.13.2.3 If more than one filter parameter is used, the results shall be combined using logical operators.

3.2.13.2.4 The response shall contain the list of requests that match all filter criteria.

NOTE – This forms a logical AND operation.

3.2.13.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_ATTRIBUTE: one or more filter attributes are invalid.

NOTES

1 The attribute is invalid when its name or type is different from any field of the BatchProductRequestDetails structure, when the comparison operator is not valid for the type of filter field, or when the range is invalid.

2 The ExtraInfo field contains the first invalid attribute.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_ATTRIBUTE	6	MAL::Composite

c) ERROR: INVALID_REQUEST_ID: the filter contains an invalid object instance identifier of the BatchProductRequestDetails.

Error	Error #	ExtraInfo Type
INVALID_REQUEST_ID	8	Not Used

3.2.14 OPERATION: monitorProduct

3.2.14.1 Overview

The monitorProduct operation requests a continuous stream of data rather than a finite stored data set.

The PUB-SUB interaction pattern is used in order to allow a consumer to subscribe for products. When new data is available, all subscribers receive a message with a product update. The operation allows the consumer to subscribe for a product by choosing an object instance identifier assigned to the product stream. This identifier is provided by the EntityKey structure:

a) the value of the firstSubKey is NULL;

b) the value of the secondSubKey is identifier of the stream;

c) the value of the thirdSubKey is NULL;

d) the value of the fourthSubKey is NULL.

Operation Identifier	monitorProduct	
Interaction Pattern	PUBLISH-SUBSCRIBE	
Pattern Sequence	Message	Body Type
OUT	PUBLISH/NOTIFY	product : (MAL::Element)

3.2.14.2 Structures

The product field shall contain the matching product.

3.2.14.3 Errors

The operation may return one of the following errors:

a) ERROR: DELIVERY_FAILED: the product could not be delivered because of a communication error.

Error	Error #	ExtraInfo Type
DELIVERY_FAILED	Defined in MAL	Not Used

b) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

c) ERROR: INVALID_STREAM_ID: the subscription identifier is invalid.

Error	Error #	ExtraInfo Type
INVALID_STREAM_ID	10	Not Used

3.2.15 OPERATION: getProductsCatalogue

3.2.15.1 Overview

The getProductsCatalogue operation provides a list of products available in the catalogue and detailed information about each of them.

Operation Identifier	getProductsCatalogue	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	filter : (COM::Archive::CompositeFilterSet)
OUT	RESPONSE	productsCatalogue : (List<ProductCatalogueEntryDetails>)

3.2.15.2 Structures

3.2.15.2.1 The filter field may contain parameters to filter a catalogue of products by any field of ProductTypeDetails, ProductProvisioningDetails, or ProductSource structures.

3.2.15.2.2 The filter may contain multiple criteria.

3.2.15.2.3 The filter may contain comparison operators, ranges of values, and wildcard characters.

3.2.15.2.4 If more than one filter parameter is used, the results shall be combined using logical operators.

3.2.15.2.5 If the filter field is NULL, the response shall contain the list of all available products.

3.2.15.2.6 The productsCatalogue field shall contain a list of ProductCatalogueEntryDetails objects.

3.2.15.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_ATTRIBUTE: one or more filter attributes are invalid.

NOTES

1 The attribute is invalid when its name or type is different from any field of the ProductTypeDetails, ProductProvisioningDetails, or ProductSource structures; when the selected comparison operation is not allowed; or when the attribute value does not match the attribute type.

2 The ExtraInfo field contains the first invalid attribute.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_ATTRIBUTE	6	MAL::Attribute

3.2.16 OPERATION: getProductSpecification

3.2.16.1 Overview

The getProductSpecification operation returns a specification of the product. The retrieved XML file is the MAL data type specification for the Product Type.

Operation Identifier	getProductSpecification	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productTypeId : (MAL::Long)
OUT	RESPONSE	productSpecification : (MAL::File)

3.2.16.2 Structures

3.2.16.2.1 The productTypeId field shall contain the object instance identifier of the ProductTypeDetails.

3.2.16.2.2 If productTypeId does not exist, an INVALID_PRODUCT_TYPE error shall be returned.

3.2.16.2.3 The response shall contain a specification of the product in the form of a MAL::File structure.

3.2.16.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

3.3 SERVICE: MISSION DATA PRODUCT MANAGEMENT

3.3.1 OVERVIEW

The Mission Data Product Management service allows the consumer to set up the Mission Data Product Distribution Service. All products supported by the system must be defined before they can be obtained. Each product is characterized by its

- type;
- format;
- source;
- internal structure;
- attributes;
- provision modes;
- compression; and
- encryption formats.

The product type, the product source, and the product provisioning details constitute ProductCatalogueEntry; they are kept in the COM archive and can be browsed by the user.

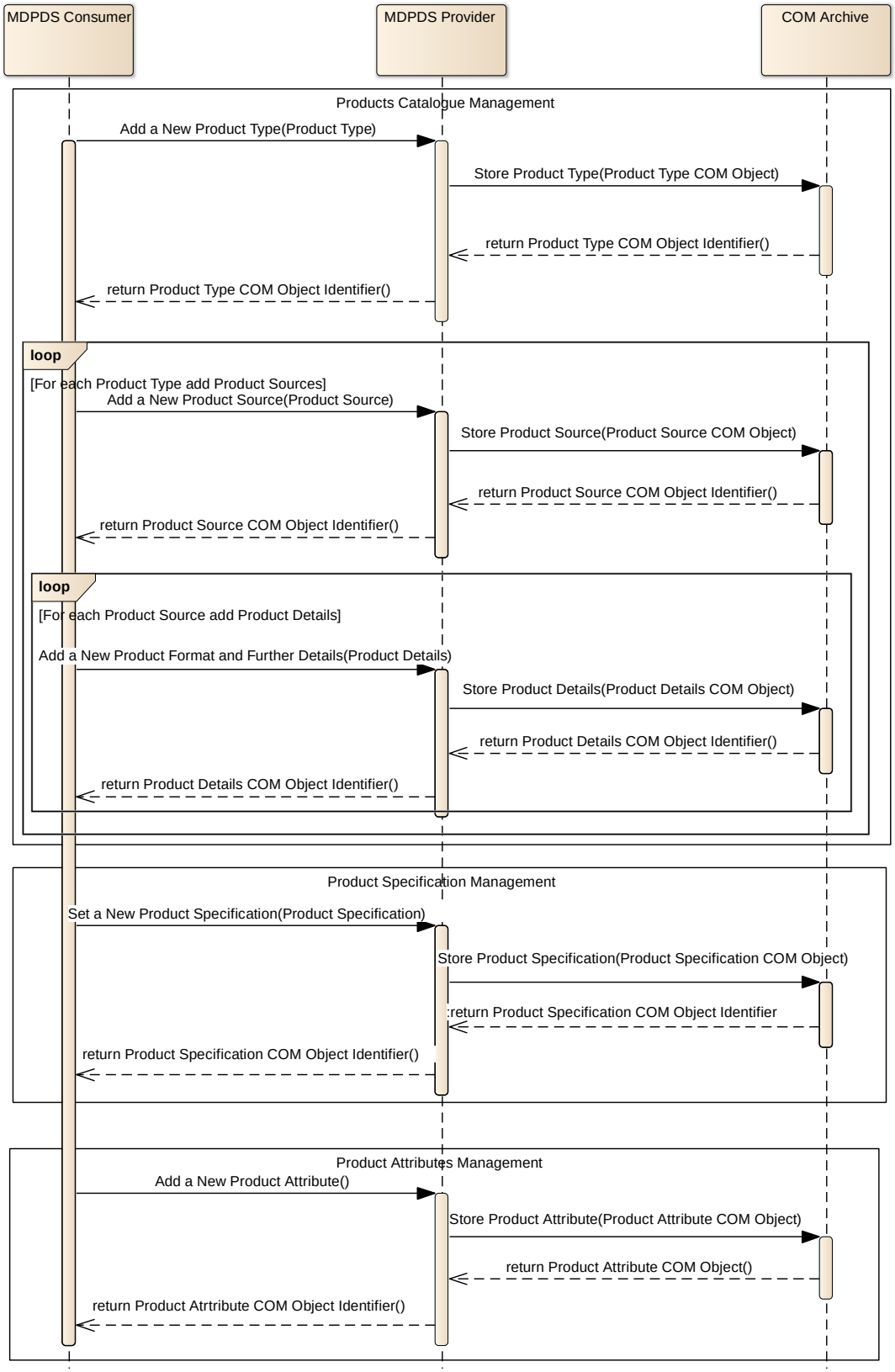

Figure 3-3: A Typical Use Case Scenario for Creating the Mission Data Product

A typical use case scenario of the Product Management service is shown in figure 3-3. For each available product, the consumer sends the provider a series of messages that result in creation of a product catalogue entry in the COM Archive. The first message, passed by the addProductType() operation, contains a request to add a new product type, along with its short description. Next, there are issued addProductSource() messages defining all the sources from which a particular type of product can be obtained. For each combination of the product type and the source, the consumer calls the addProductDetail() operation to specify formats, provisioning modes, and, optionally, other details describing the product. When a unique triad of the product type, source, and details has been provided, the provider automatically creates a catalogue entry. The next step is concerned with setting the product specification by executing the setProductSpecification() operation for each particular product. This action is recommended, but not required, and allows users to obtain from the system an unambiguous product specification. At the end, it is optional to define which attributes of the product specification can be used for product filtering or sorting (addProductAttributes() operation). In case no attributes for the product have been defined, usage of filter or sort fields in a request for this product will cause an error.

Table 3-4: Mission Data Product Management Service Operations

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
distribution	MissionDataProductManagement	9	2	1
Interaction Pattern	**Operation Identifier**	**Operation Number**	**Support in Replay**	**Capability Set**
REQUEST	addProductType	1	No	1
SUBMIT	removeProductType	2	No	
REQUEST	addProductSource	3	No	2
SUBMIT	removeProductSource	4	No	
REQUEST	addProductFormat	5	No	3
SUBMIT	removeProductFormat	6	No	
REQUEST	addProductDetail	7	No	4
SUBMIT	removeProductDetail	8	No	
REQUEST	setProductSpecification	9	No	5
SUBMIT	removeProductSpecification	10	No	
REQUEST	addProductAttribute	11	No	6
SUBMIT	removeProductAttribute	12	No	
REQUEST	enableStreaming	13	No	7
SUBMIT	disableStreaming	14	No	

3.3.2 HIGH LEVEL REQUIREMENTS

3.3.2.1 The Mission Data Product Management service shall provide the capability for

a) maintaining the catalogue of products;

b) maintaining the list of product specifications; and

c) maintaining the list of product attributes.

3.3.2.2 It shall be possible to add and delete

a) product types;

b) product sources;

c) product provisioning details; and

d) product attributes.

3.3.2.3 It shall be possible to set and delete a product specification.

3.3.2.4 The Mission Data Product Management service shall use the COM archive service to store

a) product types;

b) product provisioning details;

c) product sources;

d) product catalogue entries;

e) product specifications; and

f) product attributes.

3.3.3 FUNCTIONAL REQUIREMENTS

3.3.3.1 A product source may be added only to an existing product type.

3.3.3.2 When a product source is removed, corresponding definitions of product provisioning details shall be also removed.

3.3.3.3 A product provisioning detail may be added only to an existing product type and product source.

3.3.3.4 When a product type is removed, corresponding definitions of product sources shall be also removed.

3.3.3.5 Only one product specification may be assigned to a product type.

3.3.3.6 A product attribute may be added only if it is defined in a product specification.

3.3.3.7 A URI field of a product source shall be an address in the form of the URI (reference [5]).

3.3.4 COM USAGE

NOTE – All COM objects described in this section are custom MDPDS objects.

3.3.4.1 A ProductType COM object represents a type of mission data product. A ProductType COM object body shall hold a name, a description, and a list of product attributes.

3.3.4.2 A ProductSource COM object represents a source of mission data product:

a) a ProductSource COM object body shall hold a URI address of a source (see reference [5]);

b) the ProductSource COM object related link shall indicate which ProductType object it uses.

3.3.4.3 A ProductFormat COM object represents a format of mission data product:

a) the ProductFormat COM object shall hold a name and a description of the format;

b) the following formats shall have been preassigned given object instance identifiers:

1) fixed value binary with identifier 1;

2) variable length binary with identifier 2;

3) split binary with identifier 3;

4) XML with identifier 4;

5) JSON with identifier 5;

6) ASCII with identifier 6;

7) PDF with identifier 7;

8) CSV with identifier 8;

9) TIFF with identifier 9;

10) JPEG with identifier 10;

11) MPEG with identifier 11;

12) AVI with identifier 12.

3.3.4.4 A ProductProvisioningDetailsInstance COM object represents detailed parameters availability of the mission data product:

a) the ProductProvisioningDetailsInstance COM object body shall contain the ProductProvisioningDetails structure, which holds the format, provision mode, and compression and encryption algorithms applicable to the mission data product;

b) the ProductProvisioningDetailsInstance COM object related link shall indicate for which ProductType provisioning details are provided;

c) the ProductProvisioningDetailsInstance COM object source link shall indicate for which ProductSource provisioning details are provided.

3.3.4.5 The ProductCatalogueEntryDetails COM object represents a description of the Product. A ProductCatalogueEntryDetails COM object body shall contain the ProductCatalogueDetailsEntry structure, which holds product type, attributes, available sources, formats, and provisioning modes.

3.3.4.6 A ProductSpecification COM object represents a specification of mission data product:

a) a ProductSpecification COM object body shall hold a File object;

b) the ProductSpecification COM object related link shall indicate which ProductType object it uses.

3.3.4.7 A ProductAttribute COM object represents a product attribute:

a) a ProductAttribute COM object body shall hold a ProductAttributeDetails structure, which contains a name, a type, and two Boolean values, which specify whether a product can be sorted or filtered;

b) the ProductAttribute COM object related link shall indicate which ProductType object it uses.

3.3.4.8 A StreamProductRequest COM object represents the body of the stream product request. A StreamProductRequest COM object body shall hold a StreamProductRequestDetails structure.

Table 3-5: Mission Data Product Management Service Object Types

Object Name	Object Number	Object Body Type	Related points to	Source Object
ProductType	1	ProductTypeDetails		
ProductSource	2	MAL::URI	ProductType	
ProductFormat	3	ProductFormatDetails		
ProductProvisioningDetailsInstance	4	ProductProvisioningDetails	ProductType	ProductSource
ProductCatalogueEntry	5	ProductCatalogueEntryDetails		
ProductSpecification	6	MAL::File	ProductType	
ProductAttribute	7	ProductAttributeDetails	ProductType	
StreamProductRequest	8	StreamProductRequestDetails		

3.3.5 COM EVENT SERVICE USAGE

3.3.5.1 When a new ProductTypeDetails is added, an event shall be generated.

3.3.5.2 The event body shall contain an identifier of the created type.

3.3.5.3 When the ProductTypeDetails is removed, an event shall be generated.

3.3.5.4 The event body shall indicate an identifier of the removed type.

3.3.5.5 The related link shall indicate a list of identifiers of ProductCatalogueEntryDetails objects, which were also removed.

3.3.5.6 When the ProductSource is added, an event shall be generated.

3.3.5.7 The event body shall indicate an identifier of the created source.

3.3.5.8 The related link shall indicate the type of the product for which the source has been created.

3.3.5.9 When the ProductSource is removed, an event shall be generated.

3.3.5.10 The event body shall indicate an identifier of the removed source.

3.3.5.11 The related link shall indicate a list of identifiers of ProductCatalogueEntryDetails objects, which were also removed.

3.3.5.12 When the ProductFormatDetails is added, an event shall be generated.

3.3.5.13 The source link shall indicate an identifier of the created format.

3.3.5.14 When the ProductFormatDetails is removed, an event shall be generated.

3.3.5.15 The source link shall indicate an identifier of the removed source.

3.3.5.16 The related link shall indicate a list of identifiers of ProductCatalogueEntry objects, which were also removed.

3.3.5.17 When the ProductProvisioningDetails is added, an event shall be generated.

3.3.5.18 The event body shall indicate an identifier of the created detail.

3.3.5.19 The related link shall indicate the type of the product for which the detail has been created.

3.3.5.20 The source link shall indicate an identifier of the newly created ProductCatalogueEntryDetails.

3.3.5.21 When the ProductProvisioningDetails is removed, an event shall be generated.

3.3.5.22 The event body shall indicate an identifier of the removed detail.

3.3.5.23 The related link shall indicate a list of identifiers of ProductCatalogueEntryDetails objects, which were also removed.

3.3.5.24 When the ProductSpecification is added, an event shall be generated.

3.3.5.25 The event body shall indicate an identifier of the created specification.

3.3.5.26 The related link shall indicate the identifier of the ProductTypeDetails for which the specification has been added.

3.3.5.27 When the ProductSpecification is removed, an event shall be generated.

3.3.5.28 The event body shall indicate an identifier of the removed specification.

3.3.5.29 The related link shall indicate the identifier of the ProductTypeDetails for which the specification is removed.

3.3.5.30 When the ProductAttributeDetails is added, an event shall be generated.

3.3.5.31 The event body shall indicate the identifier of the ProductTypeDetails for which the attribute has been added.

3.3.5.32 When the ProductAttributeDetails is removed, an event shall be generated.

3.3.5.33 The event body shall indicate the identifier of the ProductTypeDetails for which the attribute is removed.

3.3.5.34 When the Product Stream is created, an event shall be generated.

3.3.5.35 The event body shall indicate an identifier of the created stream.

3.3.5.36 When the Product Stream is closed, an event shall be generated.

3.3.5.37 The event body shall indicate an identifier of the removed stream.

Table 3-6: Mission Data Product Management Service Events

Event Name	Object Number	Object Body Type	Related points to	Source points to
ProductTypeCreated	101	MAL::Long		
ProductTypeRemoved	102	MAL::Long	ProductCatalogueEntry	
ProductSourceCreated	103	MAL::Long	ProductType	
ProductSourceRemoved	104	MAL::Long	ProductCatalogueEntry	
ProductFormatCreated	105	MAL::Long	ProductType	
ProductFormatRemoved	106	MAL::Long		
ProductDetailCreated	107	MAL::Long	ProductType	ProductCatalogueEntry
ProductDetailRemoved	108	MAL::Long	ProductCatalogueEntry	
ProductSpecificationCreated	109	MAL::Long	ProductType	
ProductSpecificationRemoved	110	MAL::Long	ProductType	
ProductAttributeCreated	111	MAL::Long	ProductType	
ProductAttributeRemoved	112	MAL::Long	ProductType	
ProductStreamCreated	113	MAL::Long		
ProductStreamClosed	114	MAL::Long		

3.3.6 DISCUSSION—COM OBJECT RELATIONSHIPS

Figure 3-4 shows the COM object and event relationships for the Mission Data Product Management service.

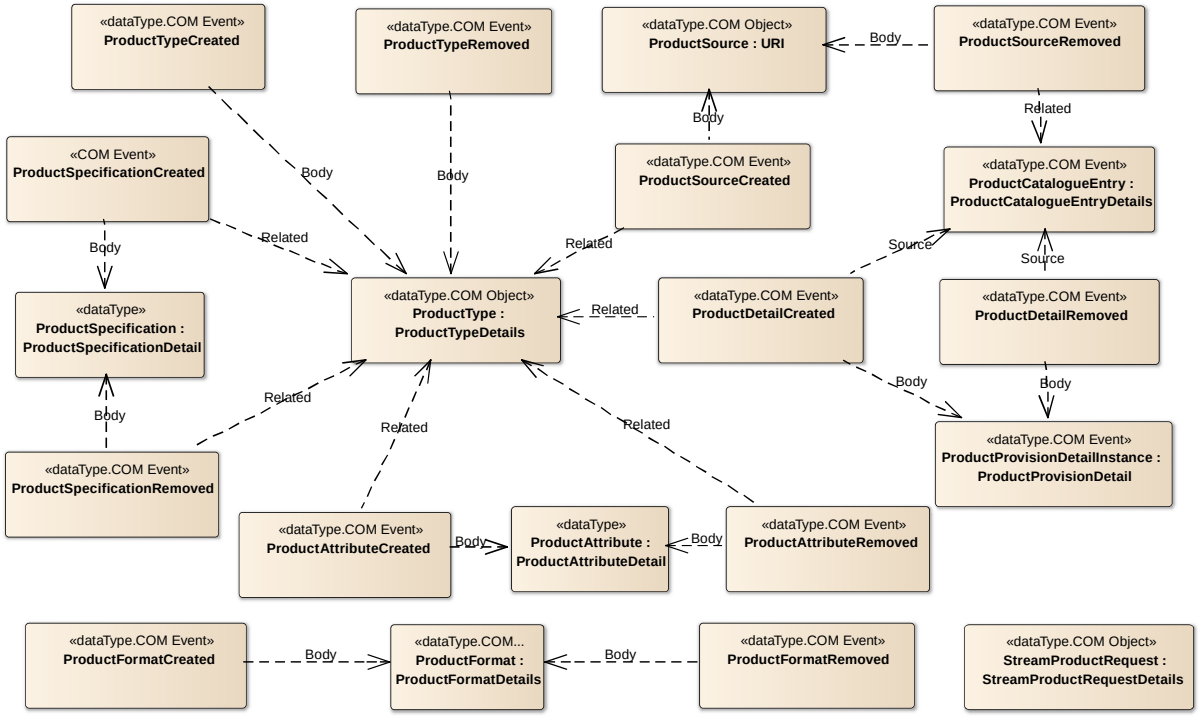

Figure 3-4: The Mission Data Product Management Service COM Object and Event Relationships

3.3.7 COM ARCHIVE SERVICE USAGE

3.3.7.1 When a new ProductTypeDetails is created with the addProductType operation, the ProductType object shall be stored in the COM archive.

3.3.7.2 When the ProductTypeDetails is removed with the removeProductType operation, the corresponding COM object shall be removed from the COM archive.

3.3.7.3 When a new ProductSource is created with the addProductSource operation, the ProductSource COM object shall be stored in the COM archive.

3.3.7.4 When the ProductSource is removed with the removeProductSource operation, the corresponding COM object shall be removed from the COM archive.

3.3.7.5 When a new ProductFormatDetails is created with the addProductFormat operation, the ProductFormat object shall be stored in the COM archive.

3.3.7.6 When the ProductFormatDetails is removed with the removeProductFormat operation, the corresponding COM object shall be removed from the COM archive.

3.3.7.7 When a new ProductProvisioningDetails is created with the addProductDetail operation, the ProductProvisioningDetailsInstance object shall be stored in the COM archive.

3.3.7.8 For each unique combination of the ProductTypeDetails, ProductSource, and ProductProvisioningDetails, a new ProductCatalogueEntry shall be stored in the COM Archive. To get objects referenced by identifier fields of the ProductCatalogueEntry, the COM Archive shall be used.

3.3.7.9 When a new ProductSpecification is created with the setProductSpecification operation, the ProductFormat object shall be stored in the COM archive.

3.3.7.10 When the ProductFormatDetails is removed with the removeProductType operation, the corresponding COM object shall be removed from the COM archive.

3.3.7.11 When the ProductType is removed, the corresponding ProductCatalogueEntry, ProductSource, and ProductProvisioningDetailsInstance shall be removed as well.

3.3.7.12 When the ProductSource is removed, the corresponding ProductCatalogueEntry and ProductProvisioningDetailsInstance shall be removed as well.

3.3.7.13 When the ProductProvisioningDetailsInstance is removed, the corresponding ProductCatalogueEntry shall be removed.

3.3.7.14 When a new ProductAttributeDetails is created with the addProductAttribute operation, the ProductAttribute object shall be stored in the COM archive.

3.3.7.15 When the ProductAttributeDetails is removed with the removeProductAttribute operation, the corresponding COM object shall be removed from the COM archive.

3.3.7.16 When a new stream is created with the enableStreaming operation, the StreamProductRequest object instance identifier shall be stored in the COM archive.

3.3.8 OPERATION: addProductType

3.3.8.1 Overview

The addProductType operation adds a new type of the product. The type of the product is used to inform to which category of space mission data the product belongs, for example, parameter, action, or alert.

Operation Identifier	addProductType	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productType : (MAL::Identifier) description : (MAL::String)
OUT	RESPONSE	productTypeId : (MAL::Long)

3.3.8.2 Structures

3.3.8.2.1 The productType field shall contain a self-descriptive name of the product type.

3.3.8.2.2 The description field may contain additional information about the product type.

3.3.8.2.3 The productTypeId field shall contain an instance identifier of the product type.

3.3.8.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: DUPLICATE: the product type has already been defined.

NOTE – The ExtraInfo Type field contains an instance identifier of the already existing product type.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	MAL::Long

3.3.9 OPERATION: removeProductType

3.3.9.1 Overview

The removeProductType operation removes a selected product type. Also, all corresponding ProductCatalogueEntry objects are removed.

Operation Identifier	removeProductType	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	productTypeId : (MAL::Long)

3.3.9.2 Structures

3.3.9.2.1 The productTypeId field shall contain the object instance identifier of the product type.

3.3.9.2.2 If productTypeId does not exist, an INVALID_PRODUCT_TYPE error shall be returned.

3.3.9.2.3 The ProductCatalogueEntry objects which contain the identifier of the product type shall be removed from the COM.

3.3.9.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

3.3.10 OPERATION: addProductSource

3.3.10.1 Overview

The addProductSource operation adds a new source of the product.

Operation Identifier	addProductSource	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productTypeId : (MAL::Long) productSource : (MAL::URI)
OUT	RESPONSE	productSourceId : (MAL::Long)

3.3.10.2 Structures

3.3.10.2.1 The productTypeId field shall contain the object instance identifier of the product type.

3.3.10.2.2 If productTypeId does not exist, an INVALID_PRODUCT_TYPE error shall be returned.

3.3.10.2.3 The productSource field shall contain an address in the form of the URI (see reference [5]).

3.3.10.2.4 The productSourceId field shall contain the object instance identifier of the product source.

3.3.10.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: DUPLICATE: the product source at the selected URI address is already defined.

NOTE – The ExtraInfo field contains an instance identifier of the already existing product source.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	MAL::Long

c) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

d) ERROR: INVALID_PRODUCT_SOURCE: the specified product source is not a valid URI address.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SOURCE	2	Not Used

3.3.11 OPERATION: removeProductSource

3.3.11.1 Overview

The removeProductSource operation removes the selected Product Source. Also, the corresponding ProductCatalogueEntry is removed.

Operation Identifier	removeProductSource	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	productSourceId : (MAL::Long)

3.3.11.2 Structures

3.3.11.2.1 The productSourceId field shall contain an instance identifier of the product source.

3.3.11.2.2 The ProductCatalogueEntry objects that contain the identifier of the product source shall be removed from the COM.

3.3.11.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_SOURCE: the selected product source does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SOURCE	2	Not Used

3.3.12 OPERATION: addProductFormat

3.3.12.1 Overview

The addProductFormat operation adds the format of data that can be used for obtaining products.

Operation Identifier	addProductFormat	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productFormat : (MAL::Identifier) description : (MAL::String)
OUT	RESPONSE	productFormatId : (MAL::Long)

3.3.12.2 Structures

3.3.12.2.1 The productFormat field shall contain a unique name of the format of the mission data product.

3.3.12.2.2 If a product format that contains the same value of the productFormat field already exists, a DUPLICATE error shall be raised.

3.3.12.2.3 The description field may contain a short description of the product format.

3.3.12.2.4 The productFormatId field shall contain an object instance identifier of the product format.

3.3.12.3 Errors

The operation may return one of the following errors:

a) ERROR: DUPLICATE: the given set of product details has already been defined.

NOTE – The ExtraInfo field contains an instance identifier of the already existing product type.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	MAL::Long

b) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

3.3.13 OPERATION: removeProductFormat

3.3.13.1 Overview

The removeProductFormat operation removes the selected product format.

Operation Identifier	removeProductFormat	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	productFormatId : (MAL::Long)

3.3.13.2 Structures

3.3.13.2.1 The productFormatId field shall contain the object instance identifier of the ProductFormat.

3.3.13.2.2 If the last ProductFormat is removed from the related ProductDetail, the ProductCatalogueEntry objects that contain the identifier of the ProductDetail shall be removed from the COM.

3.3.13.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_FORMAT: the specified product format does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_FORMAT	3	Not Used

3.3.14 OPERATION: addProductDetail

3.3.14.1 Overview

The addProductDetail operation assigns the format of the product, the support for provision modes, and a list of compression and encryption algorithms to the combination of the product type and the product source. The triad of the product type, product source, and product detail uniquely describes the available product and is kept in a new ProductCatalogueEntry object.

Operation Identifier	addProductDetail	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productTypeId : (MAL::Long) productSourceId : (MAL::Long) productDetail : (ProductProvisioningDetails)
OUT	RESPONSE	productDetailId : (MAL::Long)

3.3.14.2 Structures

3.3.14.2.1 The productTypeId field shall contain the object instance identifier of the product type.

3.3.14.2.2 If productTypeId does not exist, an INVALID_PRODUCT_TYPE error shall be returned.

3.3.14.2.3 The productSourceId field shall contain the object instance identifier of the product source.

3.3.14.2.4 If productSourceId does not exist, an INVALID_PRODUCT_SOURCE error shall be returned.

3.3.14.2.5 The productDetail shall contain the details about the particular format of the product, its provision modes, and compression and encryption methods.

3.3.14.2.6 If a product detail containing the same value of productFormat field already exists for the selected product type and product source, a DUPLICATE error shall be raised.

3.3.14.2.7 The productDetailId field shall contain an object instance identifier of the ProductProvisioningDetailInstance.

3.3.14.3 Errors

The operation may return one of the following errors:

a) ERROR: DUPLICATE: the given set of product details has already been defined.

NOTE – The ExtraInfo field contains an instance identifier of the already existing product type.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	MAL::Long

b) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

c) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

d) ERROR: INVALID_PRODUCT_SOURCE: the specified product source does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SOURCE	2	Not Used

3.3.15 OPERATION: removeProductDetail

3.3.15.1 Overview

The removeProductDetail operation removes the selected product detail. Also, the corresponding ProductCatalogueEntry is removed.

Operation Identifier	removeProductDetail	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	productDetailId : (MAL::Long)

3.3.15.2 Structures

3.3.15.2.1 The productDetailId field shall contain the object instance identifier of the ProductDetail.

3.3.15.2.2 The ProductCatalogueEntry objects that contain the object instance identifier of the ProductDetail shall be removed from the COM.

3.3.15.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_DETAIL: the selected product detail does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_DETAIL	4	Not Used

3.3.16 OPERATION: setProductSpecification

3.3.16.1 Overview

The setProductSpecification operation sets a specification of what the selected product consists. If a specification has been already set, the new specification overrides the previous one.

Operation Identifier	setProductSpecification	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productTypeId : (MAL::Long) productSpecification : (MAL::File)
OUT	RESPONSE	productSpecificationId : (MAL::Long)

3.3.16.2 Structures

3.3.16.2.1 The productTypeId field shall contain the object instance identifier of the Product Type.

3.3.16.2.2 If productTypeId does not exist, an INVALID_PRODUCT_TYPE error shall be returned.

3.3.16.2.3 The productSpecification field shall contain a specification of the product.

3.3.16.2.4 The productSpecificationId field shall contain an object instance identifier of the ProductSpecification.

3.3.16.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

c) ERROR: INVALID_PRODUCT_SPECIFICATION: the structure describing the product type is invalid.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SPECIFICATION	5	Not Used

3.3.17 OPERATION: removeProductSpecification

3.3.17.1 Overview

The removeProductSpecification operation removes a specification of what the selected product consists.

Operation Identifier	removeProductSpecification	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	productSpecificationId : (MAL::Long)

3.3.17.2 Structures

The productSpecificationId field shall contain the object instance identifier of the product specification.

3.3.17.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_SPECIFICATION: the selected product specification does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SPECIFICATION	5	Not Used

3.3.18 OPERATION: addProductAttribute

3.3.18.1 Overview

The addProductAttribute operation defines an attribute that can be used to filter or sort a product request.

Operation Identifier	addProductAttribute	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	productTypeId : (MAL::Long) productAttribute : (ProductAttributeDetails)
OUT	RESPONSE	productAttributeId : (MAL::Long)

3.3.18.2 Structures

3.3.18.2.1 The productTypeId field shall contain the object instance identifier of the product type.

3.3.18.2.2 If productTypeId does not exist, an INVALID_PRODUCT_TYPE error shall be returned.

3.3.18.2.3 The productAttribute field shall contain a new attribute of the product.

3.3.18.2.4 The productAttributeId field shall contain an object instance identifier of the product attribute.

3.3.18.3 Errors

The operation may return one of the following errors:

a) ERROR: DUPLICATE: the name field of Product Attribute already exists for the selected Product Type.

NOTE – The ExtraInfo field contains an object instance identifier of the duplicated attribute.

Error	Error #	ExtraInfo Type
DUPLICATE	Defined in COM	MAL::Long

b) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

c) ERROR: INVALID_PRODUCT_TYPE: the specified product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

3.3.19 OPERATION: removeProductAttribute

3.3.19.1 Overview

The removeProductAttribute operation removes an attribute that can be used to filter or sort a product request.

Operation Identifier	removeProductAttribute	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	productAttributeId : (MAL::Long)

3.3.19.2 Structures

3.3.19.2.1 The productAttributeId field shall contain the object instance identifier of the product attribute.

3.3.19.2.2 If productAttributeId does not exist, an INVALID_PRODUCT_ATTRIBUTE error shall be returned.

3.3.19.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_ATTRIBUTE: the attributeName field shall contain the name of the attribute.

NOTE – The ExtraInfo field contains the first invalid attribute.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_ATTRIBUTE	6	MAL::Attribute

3.3.20 OPERATION: enableStreaming

3.3.20.1 Overview

The enableStreaming operation creates a new stream of the product instances. The stream contains the space mission data that match the selected type, source, and format, and that satisfy filter criteria. Additionally, an encryption or a compression of the stream data may be enabled.

Operation Identifier	enableStreaming	
Interaction Pattern	REQUEST	
Pattern Sequence	Message	Body Type
IN	REQUEST	request : (StreamProductRequestDetails)
OUT	RESPONSE	streamId : (MAL::Long)

3.3.20.2 Structures

3.3.20.2.1 The request field shall contain the StreamProductRequestDetails.

3.3.20.2.2 The productTypeId field of the supplied StreamProductRequestDetails structure shall match product type object instance identifier in the COM Archive; otherwise an INVALID_PRODUCT_TYPE error shall be raised.

3.3.20.2.3 The productSourceId field of the supplied StreamProductRequestDetails structure shall match product source object instance identifier in the COM Archive; otherwise an INVALID_PRODUCT_SOURCE error shall be raised.

3.3.20.2.4 The productFormatId field of the supplied StreamProductRequestDetails structure shall match product format object instance identifier in the COM Archive; otherwise an INVALID_PRODUCT_FORMAT error shall be raised.

3.3.20.2.5 If for the selected product type, product source, and product format the corresponding ProductCatalogueEntryDetails does not support the stream mode, a PRODUCT_NOT_SUPPORTED_IN_STREAM error shall be returned.

3.3.20.2.6 The streamId field in the acknowledge message shall contain the object instance identifier of the created product stream.

3.3.20.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_PRODUCT_TYPE: the selected product type does not exist.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_TYPE	1	Not Used

c) ERROR: COMPRESSION_FORMAT_NOT_SUPPORTED: the compression format for a selected product is invalid or not supported by the implementation.

Error	Error #	ExtraInfo Type
COMPRESSION_FORMAT_NOT_SUPPORTED	11	Not Used

d) ERROR: ENCRYPTION_ALGORITHM_NOT_SUPPORTED: the encryption algorithm for a selected product is invalid or not supported by the implementation.

Error	Error #	ExtraInfo Type
ENCRYPTION_ALGORITHM_NOT_SUPPORTED	12	Not Used

e) ERROR: PRODUCT_NOT_SUPPORTED_IN_STREAM: the product is not supported in the stream mode for the specified combination of a product type, format, and source.

NOTE – The ExtraInfo field contains the instance identifier of the ProductCatalogueEntryDetails that provides available provision modes for the combination of the product type, source, and format.

Error	Error #	ExtraInfo Type
PRODUCT_NOT_SUPPORTED_IN_STREAM	17	MAL::Long

f) ERROR: INVALID_PRODUCT_SOURCE: the product source is not valid for the specified combination of product type and format.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_SOURCE	2	Not Used

g) ERROR: INVALID_PRODUCT_FORMAT: the format is not supported for the specified product type.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_FORMAT	3	Not Used

h) ERROR: INVALID_PRODUCT_ATTRIBUTE: one or more filter attributes are invalid.

NOTES

1 The attribute may not have been defined for the product, the specified comparison operation is not allowed, or the attribute value does not match the product type or format.

2 The ExtraInfo field contains the first invalid attribute.

Error	Error #	ExtraInfo Type
INVALID_PRODUCT_ATTRIBUTE	6	MAL::Attribute

3.3.21 OPERATION: disableStreaming

3.3.21.1 Overview

The disableStreaming operation closes the product stream.

Operation Identifier	disableStreaming	
Interaction Pattern	SUBMIT	
Pattern Sequence	Message	Body Type
IN	SUBMIT	streamId : (MAL::Long)

3.3.21.2 Structures

3.3.21.2.1 The streamId field shall contain the object instance identifier of the stream.

3.3.21.2.2 If requestId does not exist, an INVALID_STREAM_ID error shall be returned.

3.3.21.2.3 If the requestId field equals ‘0’, all the streams shall be closed.

3.3.21.3 Errors

The operation may return one of the following errors:

a) ERROR: UNKNOWN: an unknown error has occurred.

Error	Error #	ExtraInfo Type
UNKNOWN	Defined in MAL	Not Used

b) ERROR: INVALID_STREAM_ID: the stream object instance identifier does not exist.

Error	Error #	ExtraInfo Type
INVALID_STREAM_ID	10	Not Used

4 DATA TYPES—AREA DATA TYPES: DISTRIBUTION

4.1 ENUMERATION: ProvisionMode

The ProvisionMode enumeration shall be used to hold the set of methods of the data retrieval.

Name	ProvisionMode	
Short Form Part	101	
Enumeration Value	Numerical Value	Comment
BATCH	1	The batch mode.
STREAM	2	The stream mode.

4.2 ENUMERATION: StateEnum

The StateEnum enumeration shall be used to hold the states of the request processing.

Name	StateEnum	
Short Form Part	102	
Enumeration Value	Numerical Value	Comment
QUEUED	1	The request has been approved and is waiting to be processed.
SCHEDULED	2	The request has been scheduled for the execution within the specified time.
IN_PROGRESS	3	The request is being processed.
SUSPENDED	4	The request processing is suspended.
CANCELED	5	The request has been cancelled.
COMPLETED	6	The request has been successfully completed.
ERROR	7	The request has failed with an error.

4.3 ENUMERATION: CompressionAlgorithmEnum

4.3.1 The CompressionAlgorithmEnum enumeration shall be used to hold the supported compression algorithms which may be applied to the requested product.

4.3.2 Numerical values assigned to algorithms shall be used in product requests.

4.3.3 For support of additional algorithms, their names and assigned numerical values shall be communicated through an out-of-band agreement.

Name	CompressionAlgorithmEnum	
Short Form Part	103	
Enumeration Value	Numerical Value	Comment
NONE	1	The compression is disabled.
TAR	2	The TAR compression format.
ZIP	3	The ZIP compression format.
TAR_GZIP	4	The GZIP compression format.

4.4 ENUMERATION: EncryptionFormatEnum

4.4.1 The EncryptionFormatEnum enumeration shall be used to hold the data encryption algorithms that may be applied to the requested product.

4.4.2 Numerical values assigned to formats shall be used in product requests.

4.4.3 For support of additional formats, their names and assigned numerical values shall be communicated through an out-of-band agreement.

Name	EncryptionFormatEnum	
Short Form Part	104	
Enumeration Value	Numerical Value	Comment
NONE	1	The encryption is disabled.
AES	2	The AES encryption algorithm.
TWOFISH	3	The TwoFish encryption algorithm.
DES	4	The DES encryption algorithm.

4.5 ENUMERATION: MalTypesEnum

The MalTypesEnum enumeration shall be used to hold the basic MAL types.

Name	MalTypesEnum	
Short Form Part	105	
Enumeration Value	Numerical Value	Comment
BLOB	1	MAL::Blob
BOOLEAN	2	MAL::Boolean
DURATION	3	MAL::Duration
FLOAT	4	MAL::Float
DOUBLE	5	MAL::Double
IDENTIFIER	6	MAL::Identifier
OCTET	7	MAL::Octet
UOCTET	8	MAL::UOctet
SHORT	9	MAL::Short
USHORT	10	MAL::UShort
INTEGER	11	MAL::Integer
UINTEGER	12	MAL::UInteger
LONG	13	MAL::Long
ULONG	14	MAL::ULong
STRING	15	MAL::String
TIME	16	MAL::Time
FINETIME	17	MAL::FineTime
URI	18	MAL::URI

4.6 COMPOSITE: ProductTypeDetails

4.6.1 The ProductTypeDetails structure shall be used to hold the Product Type and a short description.

4.6.2 It may additionally be used to hold a list of product attributes used to filter or sort a product.

Name	ProductTypeDetails		
Extends	MAL::Composite		
Short Form Part	1		
Field	Type	Nullable	Comment
name	MAL::Identifier	No	The unique name of the type of the mission data product.
description	MAL::String	Yes	The short description of the mission data product.
productAttributesIds	List<MAL::Long>	Yes	The list of object instance identifiers of product attributes.

4.7 COMPOSITE: ProductFormatDetails

4.7.1 The ProductFormatDetails structure shall be used to hold the format in which any product can occur.

4.7.2 It shall contain a name of the format and its description.

Name	ProductFormatDetails		
Extends	MAL::Composite		
Short Form Part	2		
Field	Type	Nullable	Comment
name	MAL::Identifier	No	The unique name of the format of the mission data product.
description	MAL::String	Yes	The short description of the mission data product format.

4.8 COMPOSITE: ProductProvisioningDetails

4.8.1 The ProductProvisioningDetails structure shall be used to hold a group of parameters of the related product.

4.8.2 It shall contain a list of formats, a provision mode, a list of compression methods, and a list of encryption methods applicable for the product request.

Name	ProductProvisioningDetails		
Extends	MAL::Composite		
Short Form Part	3		
Field	Type	Nullable	Comment
productFormats	List<MAL::Long>	Yes	The list of formats in which the product can be obtained.
provisionMode	ProvisionMode	No	The availability of the product data in the batch mode and in the stream mode.
compression	List<MAL::UInteger>	Yes	The list containing all supported compression formats applicable to the product.
encryption	List<MAL::UInteger>	Yes	The list containing all supported encryption formats applicable to the product.

4.9 COMPOSITE: ProductAttributeDetails

4.9.1 The ProductAttributeDetails structure shall be used to hold a single product attribute.

4.9.2 It shall contain a name of the attribute and a type of data.

NOTE – The product attribute is one of the product specification fields and is used to indicate that this particular field can be used in product requests for filtering or sorting purposes.

Name	ProductAttributeDetails		
Extends	MAL::Composite		
Short Form Part	4		
Field	Type	Nullable	Comment
name	MAL::Identifier	No	The name of an attribute.
type	MalTypesEnum	No	The type of an attribute.
filterApplicable	MAL::Boolean	No	The availability to filter by the attribute.
sortApplicable	MAL::Boolean	No	The availability to sort by the attribute.

4.10 COMPOSITE: ProductCatalogueEntryDetails

The ProductCatalogueEntryDetails shall be used to hold object instance identifiers of the ProductType, ProductSource, and ProductProvisioningDetailsInstance.

NOTE – The ProductCatalogueEntryDetails defines an available product in terms of a type, a source, and additional details. COM Archive is used to retrieve referenced objects.

Name	ProductCatalogueEntryDetails		
Extends	MAL::Composite		
Short Form Part	5		
Field	Type	Nullable	Comment
productTypeId	MAL::Long	No	The object instance identifier of the ProductType.
productSourceId	MAL::Long	No	The object instance identifier of the ProductSource.
productProvisioningDetailId	MAL::Long	No	The object instance identifier of the ProductProvisioningDetailsInstance.

4.11 COMPOSITE: TimeRange

The TimeRange structure shall be used to hold a start time and an end time.

Name	TimeRange		
Extends	MAL::Composite		
Short Form Part	6		
Field	Type	Nullable	Comment
startTime	MAL::FineTime	Yes	The timestamp, in picoseconds, that represents a point in time from which the data matching the condition will be sent.
endTime	MAL::FineTime	Yes	The timestamp, in picoseconds, that represents a point in time until which the data matching the condition will be sent.

4.12 COMPOSITE: Schedule

The Schedule structure shall be used to hold parameters needed to schedule an execution of a request.

Name	Schedule		
Extends	MAL::Composite		
Short Form Part	7		
Field	**Type**	**Nullable**	**Comment**
startTime	MAL::Time	No	The point in time in the UTC time zone when a transmission shall be started.
repeatable	MAL::Boolean	Yes	The flag that determines whether data shall be retransmitted periodically with given interval.
interval	MAL::Duration	Yes	The interval between sub-requests.
expiryDate	MAL::Time	Yes	The point in time in the UTC time zone when the request becomes expired.

4.13 COMPOSITE: BatchProductRequestDetails

The BatchProductRequestDetails shall be used to hold parameters needed to make a product request.

NOTE – The parameters specify criteria that the product to be retrieved must meet. The structure is used both in batch and stream requests.

Name	BatchProductRequestDetails		
Extends	MAL::Composite		
Short Form Part	8		
Field	**Type**	**Nullable**	**Comment**
productTypeId	MAL::Long	No	The object instance identifier of the ProductType.
productSourceId	MAL::Long	No	The object instance identifier of the ProductSource.
productFormatId	MAL::Long	No	The object instance identifier of the ProductFormat.
destination	DestinationDeliveryDetails	Yes	The structure used to indicate a local ('pull delivery') or remote ('push delivery') destination of the product delivery. The structure contains an address of the product delivery, an optional security token, and attributes of the transport protocol. If NULL, the product is delivered directly to the consumer.
timeRange	TimeRange	Yes	The period of time during which the mission data product is requested.
filter	COM::Archive::CompositeFilterSet	Yes	The filter conditions that the product to retrieve must meet. The response data may be filtered only on attributes defined in the list of product attributes. If NULL, all the data available in the selected time range are returned.
schedule	Schedule	Yes	The structure that defines the schedule of the request.
encryption	MAL::UInteger	Yes	The encryption algorithm to be applied to response data.
compression	MAL::UInteger	Yes	The compression algorithm to be applied to response data.

Name	BatchProductRequestDetails		
Extends	MAL::Composite		
Short Form Part	8		
Field	Type	Nullable	Comment
sortField	MAL::String	Yes	The name of the field to be used to sort the response. Must comply with the name defined in a list of product attributes. If true, the response is sorted in the ascending order; otherwise, it is sorted in the descending order.
chunkSize	MAL::Long	Yes	The maximum size of the response message. If the response message exceeds the specified value, data is split into chunks.

4.14 COMPOSITE: DestinationDeliveryDetails

The DestinationDeliveryDetails structure provides details needed for local or remote product delivery and shall be used to hold an address of the destination, an optional security token, and attributes of the transport protocol.

Name	DestinationDeliveryDetails		
Extends	MAL::Composite		
Short Form Part	9		
Field	**Type**	**Nullable**	**Comment**
uri	MAL::URI	No	The fully qualified address in the form of the URI (reference [5]). It specifies where to deliver requested mission data product. The address shall indicate the protocol used for sending the data and the identifier of the host where the data shall be delivered. If the host identifier contains reserved word 'localhost', then the response shall be stored locally by the MDPDS. The name used for storing the response is optional. If provided, it may be a literal or templated string. The template contains special identifiers, enclosed in brackets, that are automatically replaced by the provider according to the following pattern (the case size does not matter): [P] - the product type; [S] - the product source; [F] - the product format; [D] - the UTC date of the response delivery; [nP] - the part number of n digits width, padded with leading zeros; [C] - the compression format; [E] - the encryption format. If the name is not provided, then the provider generates it automatically, using the following pattern: [T]_[S]_[D].[F].[C].[E].[P] (expanded to: type_source_date.format.compression.encryption.partN).
securityToken	MAL::Blob	Yes	The securityToken used to provide details needed for the authentication to the remote destination.
protocolAttributes	ProtocolAttributeDetails	Yes	The protocolAttributes is an abstract structure that may contain any additional attributes required by the protocol or remote destination to handle the data transfer.

4.15 COMPOSITE: ProtocolAttributeDetails

The ProtocolAttributeDetails abstract structure may be used to hold any additional attributes required by the protocol or remote destination to handle the data transfer.

Name	ProtocolAttributeDetails
Extends	MAL::Composite
Short Form Part	10

4.16 COMPOSITE: RemoteResponse

The RemoteResponse structure shall be used to hold a location, size, and delivery date of the product response.

NOTE – The RemoteResponse structure is sent in update messages of the requestProduct operation only when the pull or push product delivery was chosen.

Name	RemoteResponse		
Extends	MAL::Composite		
Short Form Part	11		
Field	Type	Nullable	Comment
responseLink	MAL::URI	No	The responseLink field shall contain the fully qualified address of the product response. The address is a direct link used for downloading the response, wherever it is stored locally or remotely. The format of the address is in the form of the URI (reference [5]).
responseSize	MAL::Long	No	The responseSize field shall contain the size (in bytes) of the product response.
deliveryDate	MAL::Time	No	The deliveryDate field shall contain the date of the delivery of the response. The date is specified in the CCSDS format, in relation to the UTC time zone.

4.17 COMPOSITE: StreamProductRequestDetails

The StreamProductRequestDetails structure shall be used to hold parameters needed for batch and stream product requests.

NOTE – The parameters specify criteria that the product to be retrieved must meet. The structure is used both in batch and stream requests.

Name	StreamProductRequestDetails		
Extends	MAL::Composite		
Short Form Part	12		
Field	**Type**	**Nullable**	**Comment**
productTypeId	MAL::Long	No	The object instance identifier of the product type.
productSourceId	MAL::Long	No	The object instance identifier of the product source.
productFormatId	MAL::Long	Yes	The object instance identifier of the product format.
filter	COM::Archive::CompositeFilterSet	Yes	The filter conditions that the product to retrieve must meet. The response data may be filtered only on attributes defined in the list of product attributes.
encryption	MAL::UInteger	Yes	The encryption algorithm to be applied to response data.
compression	MAL::UInteger	Yes	The compression algorithm to be applied to response data.
expiryDate	MAL::FineTime	Yes	The point in the time in the UTC time zone when the stream shall be automatically closed.

4.18 COMPOSITE: TransferReport

The TransferReport structure shall be used to hold parameters summarizing the transmitted data.

Name	TransferReport		
Extends	MAL::Composite		
Short Form Part	13		
Field	**Type**	**Nullable**	**Comment**
requestsIds	List<MAL::Long>	No	Object instance identifiers assigned by the system for every accepted request.
messages	MAL::Long	No	The total number of update messages matching the request.
responseSize	MAL::Long	No	The total size in bytes of all the update messages matching the request.
startTime	MAL::Time	Yes	The start time in the UTC time zone of the request processing.
endTime	MAL::Time	Yes	The end time in the UTC time zone of the request processing.

4.19 COMPOSITE: RequestStatus

The RequestStatus structure shall be used to hold parameters indicating the current progress of the request.

Name	RequestStatus		
Extends	MAL::Composite		
Short Form Part	14		
Field	**Type**	**Nullable**	**Comment**
requestId	MAL::Long	No	The object instance identifier of the request.
userId	MAL::Long	No	The object instance identifier of the user who issued the request. The identifier shall correspond to the Profile structure defined in the Login Service.
state	StateEnum	No	The current state of the request processing.
percentage	MAL::Float	No	The progress of the request processing.
updatesDelivered	MAL::Long	No	The number of update messages delivered so far.
dataDelivered	MAL::Long	No	The size (in bytes) of all data delivered so far.
creationTime	MAL::Time	No	The time in the UTC time zone when request was received by system.
startTime	MAL::Time	Yes	The start time for the execution of the request processing. If it is NULL, the endTime must be NULL.
endTime	MAL::Time	Yes	The end time in the UTC time zone for the execution of the request processing. If it is not NULL, the startTime must not be NULL.
schedule	Schedule	Yes	Parameters of the schedule of the request.

4.20 COMPOSITE: ParameterValueTimePair

The ParameterValueTimePair structure shall be used to hold an instance of the parameter value and the time of its generation.

Name	ParameterValueTimePair		
Extends	MAL::Composite		
Short Form Part	201		
Field	Type	Nullable	Comment
parameterValue	MC::Parameter::ParameterValue	No	A value of the parameter.
time	MAL::FineTime	No	An accurate on-board time in the UTC time zone of the generation of the parameter.

4.21 COMPOSITE: ParameterTimeline

The ParameterTimeline structure shall be used to hold values of the specific parameter along with their timestamps, the parameter source, and the object instance identifier of the parameter definition.

Name	ParameterTimeline		
Extends	MAL::Composite		
Short Form Part	202		
Field	Type	Nullable	Comment
timestampedParameterValues	List<ParameterValueTimePair>	No	A list containing values of the specific parameter along with accurate on-board times in the UTC time zone of their generation.
provider	MAL::URI	No	A source of the parameter. It uniquely points a mission, a domain, a spacecraft, and any further details describing where the parameter comes from.
parameterDefinitionId	MAL::Long	No	The object instance identifier of the parameter definition.

4.22 COMPOSITE: AggregationValueTimePair

The AggregationValueTimePair structure shall be used to hold an instance of the aggregation value and the time of its generation.

Name	AggregationValueTimePair		
Extends	MAL::Composite		
Short Form Part	203		
Field	Type	Nullable	Comment
aggregationValue	MC::Aggregation::AggregationValue	No	A value of the aggregation.
time	MAL::FineTime	No	An accurate on-board time in the UTC time zone of the generation of the aggregation.

4.23 COMPOSITE: AggregationTimeline

The AggregationTimeline structure shall be used to hold values of the specific aggregation along with their timestamps, the parameter source, and the object instance identifier of the AggregationDefinitionDetails.

Name	AggregationTimeline		
Extends	MAL::Composite		
Short Form Part	204		
Field	Type	Nullable	Comment
timestampedAggregationValues	List<AggregationValueTimePair>	No	A list containing values of the specific aggregation along with accurate on-board times in the UTC time zone of their generation.
provider	MAL::URI	No	A provider of the parameter. It uniquely points a mission, a domain, a spacecraft, and any further details describing where the aggregation comes from.
aggregationDefinitionId	MAL::Long	No	The object instance identifier of the AggregationDefinitionDetails.

4.24 COMPOSITE: ActionStageTimePair

The ActionStageTimePair structure shall be used to hold the execution stage of the action and its time.

Name	ActionStageTimePair		
Extends	MAL::Composite		
Short Form Part	205		
Field	Type	Nullable	Comment
stageTime	MAL::FineTime	No	An accurate on-board time in the UTC time zone of the action execution stage.
stageSuccess	MAL::UInteger	No	An execution status of the stage. A zero value means a success, while a non-zero value means an error.

4.25 COMPOSITE: ActionInstanceReport

The ActionInstanceReport structure shall be used to hold parameters reporting on the status of each verification of the action instance.

NOTE – Therefore it contains time-stamped statuses for the particular action instance.

Name	ActionInstanceReport		
Extends	MAL::Composite		
Short Form Part	206		
Field	Type	Nullable	Comment
actionInstanceDetails	MC::Action::ActionInstanceDetails	No	Details of the specific action.
executionStages	List<ActionStageTimePair>	No	An ordered list of action execution stages containing the UTC times and execution statuses assigned to each stage.
invoker	MAL::URI	No	A source where action was generated.
destination	MAL::URI	No	A destination system to execute the action. It uniquely points a mission, a domain, a spacecraft, and a system where the action is to be executed.
actionDefinitionId	MAL::Long	No	The ID of the action definition.

4.26 COMPOSITE: ActionsHistory

The ActionsHistory structure shall be used to hold a list of ActionInstanceReports.

Name	ActionsHistory		
Extends	MAL::Composite		
Short Form Part	207		
Field	Type	Nullable	Comment
actions	List<ActionInstanceReport>	No	A time-tagged history of actions.

4.27 COMPOSITE: AlertReport

The AlertReport structure shall be used to hold parameters making up the alert report.

Name	AlertReport		
Extends	MAL::Composite		
Short Form Part	208		
Field	Type	Nullable	Comment
name	MAL::Identifier	No	An alert name.
category	MAL::String	No	An alert category.
time	MAL::FineTime	No	An alert timestamp in the UTC time zone.
source	MAL::URI	No	A source where the alert was generated.
alertDefinitionDetailsId	MAL::Long	No	The reference to the definition of an alert, including any argument definitions.
alertEventDetailsId	MAL::Long	No	The reference to the structure holding the details of the instance of an alert.

4.28 COMPOSITE: AlertsHistory

The AlertsHistory structure shall be used to hold a list of AlertReport objects.

Name	AlertsHistory		
Extends	MAL::Composite		
Short Form Part	209		
Field	Type	Nullable	Comment
alerts	List<AlertReport>	No	A time-tagged history of alerts.

4.29 COMPOSITE: CheckReport

The CheckReport structure shall be used to hold parameters making up the parameter check report.

Name	CheckReport		
Extends	MAL::Composite		
Short Form Part	210		
Field	Type	Nullable	Comment
checkDefinitionDetailsId	MAL::Long	No	The reference to the structure holding the definition of the check.
checkSummary	MC::Check::CheckSummary	No	Details about a specific check link and its evaluated result.
constantCheckDefinitionId	MAL::Long	Yes	The reference to the consistency check.
referenceCheckDefinitionId	MAL::Long	Yes	The reference to the reference check.
deltaCheckDefinitionId	MAL::Long	Yes	The reference to the delta transition check.
limitCheckDefinitionId	MAL::Long	Yes	The reference to the low and high limit check.
compountCheckDefinitionId	MAL::Long	Yes	The reference to the compound check.
timestamp	MAL::FineTime	No	The timestamp in the UTC time zone of the check report.

4.30 COMPOSITE: ChecksHistory

The ChecksHistory structure shall be used to hold a list of CheckReport objects.

Name	ChecksHistory		
Extends	MAL::Composite		
Short Form Part	211		
Field	Type	Nullable	Comment
checks	List<CheckReport>	No	A time-tagged history of check reports.

4.31 COMPOSITE: MultipleParameterTimeline

The MultipleParameterTimeline structure shall be used to hold a group of ParameterTimeline objects.

Name	MultipleParameterTimeline		
Extends	MAL::Composite		
Short Form Part	212		
Field	Type	Nullable	Comment
parameterTimelineList	List<ParameterTimeline>	No	A list of multiple parameter timelines.

5 ERROR CODES

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

Table 5-1: Mission Data Product Distribution Services Error Codes

Error	Error #	Comment
INVALID_PRODUCT_TYPE	1	The product type does not exist.
INVALID_PRODUCT_SOURCE	2	The source does not exist or is not valid for the selected product type.
INVALID_PRODUCT_FORMAT	3	The product format does not exist or is not supported for the combination of the selected product type and the source.
INVALID_PRODUCT_DETAIL	4	The product detail does not exist.
INVALID_PRODUCT_SPECIFICATION	5	The product specification does not exist or is not correct.
INVALID_PRODUCT_ATTRIBUTE	6	One or more filter attributes are invalid. The attribute has not been defined for the product, the selected comparison operation is not allowed, or the attribute value does not match the product type or the format.
INVALID_TIME_RANGE	7	The date or the date format in the TimeRange structure is invalid.
INVALID_SCHEDULE	8	The schedule structure contains an invalid date, invalid date format, or invalid expiry date.
INVALID_REQUEST_ID	9	The object instance identifier of the BatchProductRequestDetails does not exist.
INVALID_STREAM_ID	10	The object instance identifier of the StreamProductRequestDetails does not exist.
COMPRESSION_FORMAT_NOT_SUPPORTED	11	The compression format for the selected product is invalid or not supported by the implementation.
ENCRYPTION_ALGORITHM_NOT_SUPPORTED	12	The encryption format for the selected product is invalid or not supported by the implementation.
SCHEDULING_NOT_SUPPORTED	13	The request execution scheduling is not supported by the implementation.
SORTING_NOT_SUPPORTED	14	The sorting attribute for the selected product is invalid or not supported by the implementation.

Error	Error #	Comment
INVALID_CHUNK_SIZE	15	The chunk size is invalid, or splitting a product into multiple update messages is not supported by the implementation.
PRODUCT_NOT_SUPPORTED_IN_BATCH	16	The product is not supported in the batch mode for the selected combination of a product type, format, and source.
PRODUCT_NOT_SUPPORTED_IN_STREAM	17	The product is not supported in the stream mode.
TRANSFER_PROTOCOL_NOT_SUPPORTED	18	The transfer protocol specified in the request is not supported by the provider.
INVALID_TRANSFER_PROTOCOL_ATTRIBUTES	19	The attributes of the transfer protocol specified in the request are invalid.
INVALID_FILE_NAME	20	The file name of the response is invalid. It may be caused by an invalid character, an invalid path, or, when using templates, an invalid special identifier.

6 SERVICE SPECIFICATION XML

6.1 OVERVIEW

This section provides a link to the definition of the service in XML notation, as specified in reference [1].

The use of XML for service specification provides a machine-readable format rather than the text-based document format. The published specifications and XML schemas are held in an online SANA registry, located at

[SANA Registry location to be supplied].

6.2 NORMATIVE XML SERVICE SPECIFICATION

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

[SANA Registry location to be supplied].

ANNEX A

IMPLEMENTATION CONFORMANCE STATEMENT (ICS) PROFORMA

(NORMATIVE)

A1 INTRODUCTION

[The required ICS Proforma annex will be supplied prior to final publication.]

A1.1 OVERVIEW

This annex provides the Implementation Conformance Statement (ICS) Requirements List (RL) for an implementation of [Specification]. The ICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation claiming conformance must satisfy the mandatory requirements referenced in the RL.

A1.2 ABBREVIATIONS AND CONVENTIONS

The RL consists of information in tabular form. The status of features is indicated using the abbreviations and conventions described below.

Item Column

The item column contains sequential numbers for items in the table.

Feature Column

The feature column contains a brief descriptive name for a feature. It implicitly means, 'Is this feature supported by the implementation?'

Status Column

The status column uses the following notations:

- M mandatory;
- O optional;
- C conditional;
- X prohibited;
- I out of scope;
- N/A not applicable.

Support Column Symbols

The support column is to be used by the implementer to state whether a feature is supported by entering Y, N, or N/A, indicating:

Y Yes, supported by the implementation.

N No, not supported by the implementation.

N/A Not applicable.

The support column should also be used, when appropriate, to enter values supported for a given capability.

A1.3 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the Recommended Standard by completing the RL; that is, the state of compliance with all mandatory requirements and the options supported are shown. The resulting completed RL is called an ICS. The implementer shall complete the RL by entering appropriate responses in the support or values supported column, using the notation described in A1.2. If a conditional requirement is inapplicable, N/A should be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference X*i*, where *i* is a unique identifier, to an accompanying rationale for the noncompliance.

A2 ICS PROFORMA FOR [SPECIFICATION]

A2.1 GENERAL INFORMATION

A2.1.1 Identification of ICS

Date of Statement (DD/MM/YYYY)	
ICS serial number	
System Conformance statement cross-reference	

A2.1.2 Identification of Implementation Under Test

Implementation Name	
Implementation Version	
Special Configuration	
Other Information	

A2.1.3 Identification of Supplier

Supplier	
Contact Point for Queries	
Implementation Name(s) and Versions	
Other information necessary for full identification, for example, name(s) and version(s) for machines and/or operating systems; System Name(s)	

A2.1.4 Identification of Specification

[CCSDS Document Number]	
Have any exceptions been required? NOTE – A YES answer means that the implementation does not conform to the Recommended Standard. Non-supported mandatory capabilities are to be identified in the ICS, with an explanation of why the implementation is non-conforming.	Yes [ ] No [ ]

A2.2 REQUIREMENTS LIST

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

The security considerations of this specification are the same as those of reference [1]. Specifically, authentication and authorization of a participating consumer or provider is provided by the MAL access control concept and is covered in subsections 3.6, 5.2, and 5.3 of the Reference Model (reference [4]).

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

B2 SANA CONSIDERATIONS

The recommendations of this document request that SANA populate the registry specified in reference [1] with the schema and XML detailed in section 6 of this document.

As stated in reference [1], the registration rule for change to this registry requires an engineering review by a designated expert. The expert shall be assigned by the WG Chair or, in their absence, the Area Director.

B3 PATENT CONSIDERATIONS

The recommendations of this document have no patent issues.

ANNEX C

INFORMATIVE REFERENCES

(INFORMATIVE)

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