

Draft Recommendation for  
Space Data System Standards

|  |
| --- |
| Mission Operations - Mission Data Product Distribution Services |

PROPOSED Draft Recommended Standard

CCSDS 522.2-R-1

RED Book

November 2016

AUTHORITY

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | Issue: | Red Book, Issue 1 |  |
|  | Date: | November 2016 |  |
|  | 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 address below.

This document is published and maintained by:

CCSDS Secretariat

Space Communications and Navigation Office, 7L70

Space Operations Mission Directorate

NASA Headquarters

Washington, DC 20546-0001, USA

Document control

|  |  |  |  |
| --- | --- | --- | --- |
| **Document** | **Title and Issue** | **Date** | **Status** |
| CCSDS 522.2-R-1 | Mission Operations - Mission Data Product Distribution Services, Proposed Draft Recommended Standard, Issue 1 | November 2016 | Current draft |
|  |  |  |  |
|  |  |  |  |

Contents

Section Page

[1 Introduction 1-1](#_Toc466628631)

[1.1 General 1-1](#_Toc466628632)

[1.2 Purpose and scope 1-1](#_Toc466628633)

[1.3 Applicability 1-2](#_Toc466628634)

[1.4 Rationale 1-2](#_Toc466628635)

[1.5 Document structure 1-2](#_Toc466628636)

[1.6 Definitions 1-2](#_Toc466628637)

[1.7 Nomenclature 1-4](#_Toc466628638)

[1.8 Conventions 1-4](#_Toc466628639)

[1.9 References 1-5](#_Toc466628640)

[2 Overview 2-7](#_Toc466628641)

[2.1 General 2-7](#_Toc466628642)

[2.2 Product concept 2-7](#_Toc466628643)

[2.3 Mission Data Product Distribution Services composition 2-8](#_Toc466628644)

[2.4 ProductsManagement service 2-11](#_Toc466628645)

[2.5 MissionDataProductDistribution service 2-14](#_Toc466628646)

[3 Specification: MISSION DATA PRODUCT DISTRIBUTION SERVICES 3-18](#_Toc466628647)

[3.1 Rationale 3-18](#_Toc466628648)

[3.2 Functional Requirements 3-18](#_Toc466628649)

[3.3 Service: MissionDataProductDistribution 3-20](#_Toc466628650)

[3.4 Service: ProductsManagement 3-40](#_Toc466628651)

[4 Data types 4-66](#_Toc466628652)

[4.1 Area data types: distribution 4-66](#_Toc466628653)

[5 Error codes 5-85](#_Toc466628654)

[6 SERVICE SPECIFICATION XML 6-88](#_Toc466628655)

[ANNEX A Implementation Conformance Statement (ICS) Proforma (normative) A-1](#_Toc466624972)

[ANNEX B Security, SANA, and Patent Considerations (Informative) B-1](#_Toc466624973)

Figure

[Figure 2‑1: Mission Operations Services Concept Document Set 2-7](#_Toc466624974)

[Figure 2‑2 The Use Case diagram for the Products Management service. 2-9](#_Toc466624975)

[**Figure 2‑3 The Use Case diagram for the Mission Data Product Distribution service.** 2-10](#_Toc466624976)

[**Figure 2‑4: Actions to create a new product** 2-11](#_Toc466624977)

[**Figure 2‑5 Actions to create a catalogue entry** 2-12](#_Toc466624978)

[**Figure 2‑6 The creation of the product stream.** 2-13](#_Toc466624979)

[**Figure 2‑7 A basic flow of actions required to request the product.** 2-14](#_Toc466624980)

[**Figure 2‑8 A batch product request.** 2-15](#_Toc466624981)

[**Figure 2‑9 A stream product request.** 2-16](#_Toc466624982)

[**Figure 3‑1 A typical use case scenario for retrieval of the mission data product.** 3-21](#_Toc466624983)

[Figure 3‑2: The Mission Data Product Distribution service COM object and event relationships 3-25](#_Toc466624984)

[**Figure 3‑3 A typical use case scenario for creating the mission data product.** 3-41](#_Toc466624985)

[Figure 3‑4: The Products Management service COM object and event relationships 3-48](#_Toc466624986)

Table

Table 3‑1 Mission Data ProductDistribution Service Operations…………………...……3-22

Table 3‑2 Mission Data Product DistributionService Object Types…………….………..3-24

Table 3‑3 Mission Data ProductDistribution Service Events……………………...……...3-24

Table 3‑4 Products Management ServiceOperations……………………………………..3-42

Table 3‑5 Products Management Service ObjectTypes…………………………………..3-45

Table 3‑6 Products Management Service Events……………………………………….…3-47

Table 5‑1 Mission Data Product Distribution Services Error Codes………………...……5-85

[Figure 2‑1: Mission Operations Services Concept Document Set 2-7](#_Toc466624987)

[Figure 2‑2 The Use Case diagram for the Products Management service. 2-9](#_Toc466624988)

[**Figure 2‑3 The Use Case diagram for the Mission Data Product Distribution service.** 2-10](#_Toc466624989)

[**Figure 2‑4: Actions to create a new product** 2-11](#_Toc466624990)

[**Figure 2‑5 Actions to create a catalogue entry** 2-12](#_Toc466624991)

[**Figure 2‑6 The creation of the product stream.** 2-13](#_Toc466624992)

[**Figure 2‑7 A basic flow of actions required to request the product.** 2-14](#_Toc466624993)

[**Figure 2‑8 A batch product request.** 2-15](#_Toc466624994)

[**Figure 2‑9 A stream product request.** 2-16](#_Toc466624995)

[**Figure 3‑1 A typical use case scenario for retrieval of the mission data product.** 3-21](#_Toc466624996)

[Figure 3‑2: The Mission Data Product Distribution service COM object and event relationships 3-25](#_Toc466624997)

[**Figure 3‑3 A typical use case scenario for creating the mission data product.** 3-41](#_Toc466624998)

[Figure 3‑4: The Products Management service COM object and event relationships 3-48](#_Toc466624999)

# Introduction

## General

This Recommended Standard defines the Mission Operations (MO) Mission Data Product Distribution Services (MDPDS) in conformance with the service framework specified in reference [1], *Mission Operations Services Concept.*

The MDPDS are a set of services 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 the product and 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 Message Abstraction Layer (MAL) [2] and utilize Common Object Model (COM) [3] Archive and Event services.

## Purpose and scope

The Recommended Standard defines, in an abstract manner, the MDPDS in terms of:

1. the key requirements of the services;
2. the operations necessary to provide the services;
3. the data types required by the service;
4. the expected behaviour of each operation;
5. the errors, which may occur during the operation execution;
6. the structure of basic products;
7. the use of COM Archive, Events and Objects.

It does not specify:

1. the design of the implementation of the services;
2. the physical location of system components;
3. the programming languages nor technologies needed for the implementation of services;
4. the communication technologies;
5. the means of cooperation with external systems to obtain mission data products for the distribution;
6. the configuration steps needed for setting up the MDPDS.

## Applicability

This specification is applicable to any system that provides mission data to other systems. The Recommended Standard may be used between ground segment systems, between space and ground segment systems and also between space segment systems.

## Rationale

The goals of this Recommended Practice 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 on-line, controlled access to space mission data for the user community (e.g. for the members of the scientific community).

## Document structure

This document is structured as follows:

1. Section 1 contains purpose and scope, applicability, rationale of the document, nomenclature, and lists definitions and referenced documents
2. Section 2 provides an overview of the concepts
3. Section 3 contains a specification of the MDPDS
4. Section 4 is a formal specification of the MDPDS data structures
5. Section 5 is a formal specification of the MDPDS errors
6. Section 6 / Annex A contains Implementation Conformance Statement
7. Section 7 / Annex B contains security, SANA, and patent considerations

## Definitions

**Batch mode:** The provisioning mode used for the retrieval of the archived data.

**Catalogue of products:** A repository containing the information related to the products supported by the MDPDS, so that the consumer can request the product.

**Delivery mode:** One of the three methods of the 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 send directly to the user, without any brokers or third-party distribution services, like the file transfer protocol (FTP).

**Mission Data Product** (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 meaning and allowed values. It can be used to filter on or to sort by the Mission Data Product.

**Product Specification: A** formal specification of all attributes of the product, their data types and meaning, written in terms of MAL data types.

**Product Detail:** A composite, which contains information about the availability of a product in the specific data format, the provision mode, compression formats and encryption formats.

**Product Format** (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, e.g. parameter time series, action history.

**Provision mode**: One of the two methods of the data retrieval (a batch mode or a stream mode).

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

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

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

## Nomenclature

### Normative text

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

1. the words ‘shall’ and ‘must’ imply a binding and verifiable specification;
2. the word ‘should’ implies an optional, but desirable, specification;
3. the word ‘may’ implies an optional specification;
4. 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.

### 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.

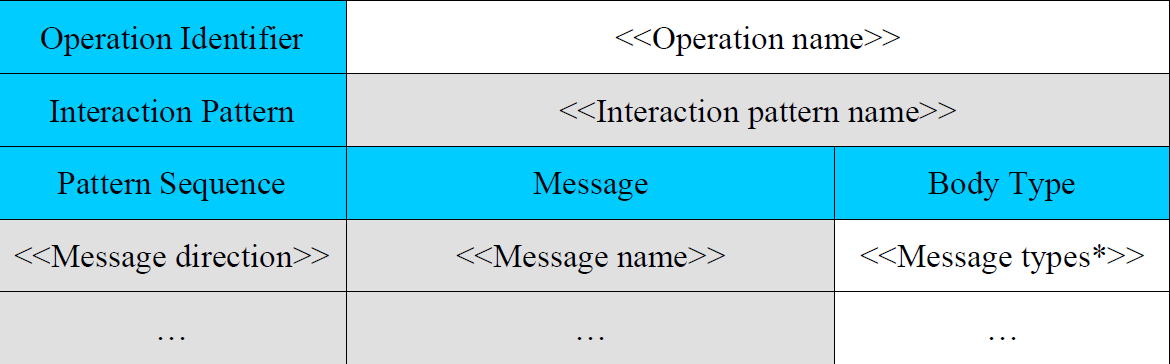
## Conventions

### Figures

In figures illustrating this document, Unified Modelling Language (UML) modelling diagrams are used. Reference [4] provides further information regarding diagrams types and their meaning.

### Tables

The format of tables presented throughout the document to illustrate MDPDS operations is described in [2]. Below is an excerpt from the mentioned document with a table format description.



The message direction denotes the direction of the message relative to the provider of the pattern and is either IN or OUT. So 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.

## 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 Services Concept. Issue 3.* Report Concerning Space Data System Standards (Green Book), CCSDS 520.0-G-3, Washington, D.C., USA: CCSDS, December 2010. |
| [2] | *Mission Operations Message Abstraction Layer. Issue 2.* Recommendation for Space Data System Standards (Blue Book), CCSDS 521.0-B-2. Washington, D.C.: CCSDS, March 2013. |
| [3] | *Mission Operations Common Object Model. Issue 1*. Recommendation for Space Data System Standards (Blue Book), CCSDS 521.1-B-1. Washington, D.C.: CCSDS, February 2014 |
| [4] | *Mission Operations Reference Model. Issue 1.* Recommendation for Space Data System Practices (Magenta Book), CCSDS 520.1-M-1. Washington, D.C.: CCSDS, July 2010. |
| [5] | *Mission Operations Monitor & Control Services. Issue 3.* Draft Recommendation for Space Data System Standards (Red Book), CCSDS 522.1-R-3, Washington, D.C., USA: CCSDS, March 2014. |

# Overview

## General

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

mal-spp-book

Figure 2‑1: Mission Operations Services Concept Document Set

The MDPDS are in compliance with MO Service Framework layers (see reference [1]) and are defined in terms of the MAL [2] Therefore it is possible to deploy them over any supported communication protocol and message transport technology.

## Product concept

The product concept has been introduced, in order to abstract from 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 of space mission data products, without making assumptions about the implementations of the underlying mission data product distribution systems.

The MDPDS specifies three elements, which combination uniquely defines each product. That triple consists of the product type, the product source and the product format. The product type represents a category of mission data, which the product belongs to (e.g. parameter value evolution in a given time period, actions history). Each product type has a list of attributes assigned to it. Attributes define fields (by providing their names, data types and usage) which the product consists of. Some attributes can be selected to filter or sort the product. The next element, the product source, contains an absolute location of an origin of the data. It indicates what exactly generated the data (e.g. particular sensor of the particular satellite of the particular mission). The product format is a standardized description detailing how to encode mission data products (e.g. a raw binary value, XML). It provides means to represent the information, which the product carries, in a standardized way. For each predefined product type, there is assigned a list of basic formats in which the product can be obtained.

In order to browse available products, the MDPDS have introduced the concept of the catalogue of products. Because the product type, product source, and product format uniquely identify every product, a single catalogue entry stores those three elements. Elements of the triple are related with each other. Every product type has assigned a list of product sources. A product format can be added only to the already defined pair of a product type and source. Due to that relation, the same order of adding product type, source and format is required to define new product. For the same reason, removal of a product source results in removal of the related product formats and, in turn, removal of a product type results in removal of the related product sources and formats.

The catalogue stores also additional information. Among them is a list of product attributes, being the subset of all the product attributes, which enumerates only the ones which can be used for filtering and sorting purposes. Furthermore, the catalogue entry contains a provisioning mode, compression methods, encryption methods and a description of the each product.

To make the product available it is required to add it to the catalogue. Optionally, it is also possible to define internal structure of the product, called the product specification. The product specification represents product attributes in a portable way, by the means of the MAL service description language. Without adding the product specification, filtering or sorting products in 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.

## Mission Data Product Distribution Services composition

The Mission Data Product Distribution Services consist of two services: Products 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 Products Management service allows manipulation of the catalogue by providing means for defining available products. It can also be seen as a component responsible for the configuration of the Mission Data Product Distribution solution. The Mission Data Product Distribution service allows 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.

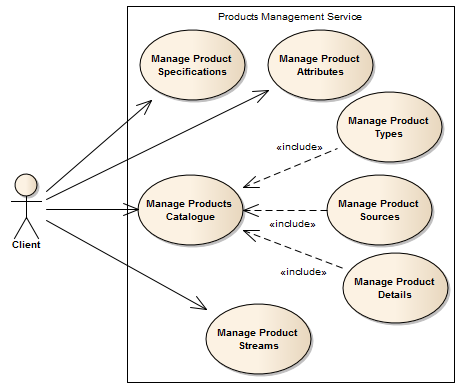
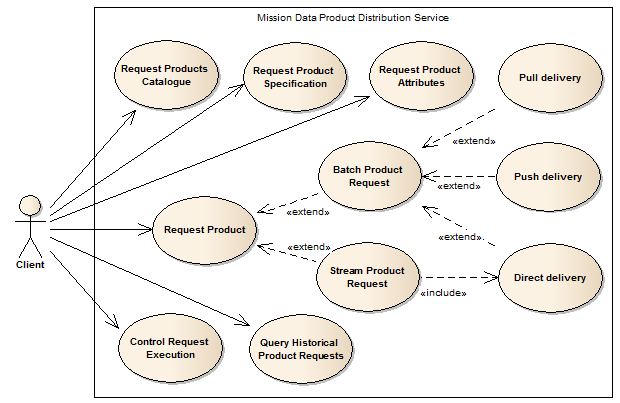


Figure 2‑2 The Use Case diagram for the Products Management service.

The use case diagram in the Figure 2‑2 presents an overview of the functionality provided by Products Management service. The main role of the service is to manage the catalogue of products. Each product, before it can be accessed, must have been created in the catalogue. As mentioned in section 2.2, the product catalogue entry contains the triple of the product type, source and format along with additional details. Hence, the management of the catalogue relies on defining catalogue entries containing all those 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 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. Choosing filtering or sorting attributes can be done only after adding a specification for the particular product.

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



**Figure 2‑3 The Use Case diagram for the Mission Data Product Distribution service.**

The model in the Figure2‑3 illustrates the capabilities offered by the Mission Data Product Distribution service. The service allows the retrieval of the catalogue of products that has been defined in the Product Management service. It provides the functionality to query the catalogue by a given set of criteria, in order to limit the number of catalogue entries only to those, in which the client may be interested.

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

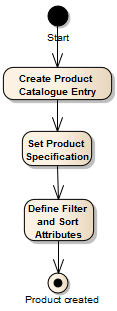
In the batch mode, the mission data product can be delivered in one of three ways: directly by the MDPDS to the requester, to the remote destination, using the third-party transfer protocol (a push delivery) and also stored by the MDPDS locally and made accessible to download to the authenticated users (a pull delivery). In 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. An 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 ability to browse them.

## ProductsManagement service

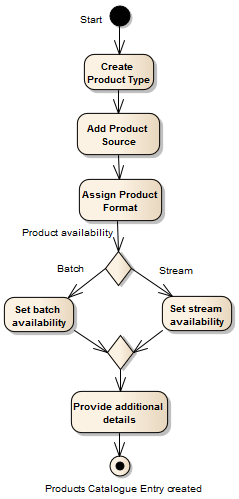
The Products Management service provides the capabilities to manage the catalogue of the products on the service provider side. It presents the consumer a convenient interface to modify catalogue entries, product specifications and product attributes. It can be also used to create a stream of product updates. Storage of any data generated in the service is delegated to the COM archive.

The process of creating a product must be conducted in the proper order, which is illustrated in the Figure 2‑4.



**Figure 2‑4: Actions to create a new product**

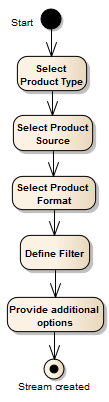
The first step of this process is to create a catalogue entry. It is complex activity that can be decomposed into group of smaller steps shown on the diagram in the Figure2‑4.



**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 could be not sufficient to explain what category of mission data the product belongs to, the consumer may additionally add a description. As sole type does not provide all information that describes the product, next step is to specify all the sources, from which the product is available. Then the only piece of information left to unambiguously distinguish different products is their data representation, which is expressed by a product format. Having defined the product type, the source and the format, 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 to provide a formal description of the product in the product specification. This is illustrated in the step two in the 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. Again, also in this case, the COM archive is used to store provided information.

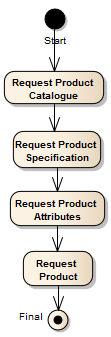


**Figure 2‑6 The creation of the product stream.**

The catalogue keeps information indicating if the particular product can be streamed. For products applicable for streaming, the Product Management service may be used to create their streams. Streams shall be created only when needed and closed afterwards, to save the system resources. Flow of actions presented in the Figure 2‑6 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 pointed. There is also provided an option to automatically close the stream after the specified date has passed.

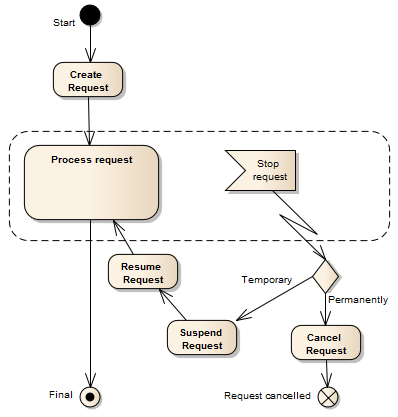
## MissionDataProductDistribution service

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

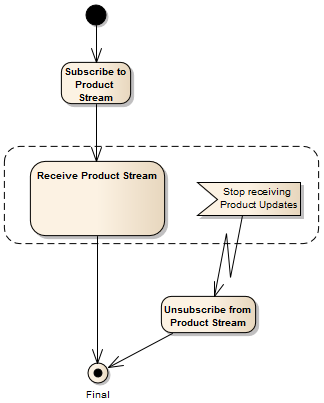


**Figure 2‑8 A batch product request.**

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

The 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 MDPDS convert it to a list of sub-requests and assign request identifiers for each one of them, so that they can be checked and controlled separately. In order not to schedule an infinite number of sub-requests, the maximum number of sub-requests or the maximum period of time to schedule sub-requests shall 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 anytime.

In the batch mode, by default, the MDPDS deliver products directly to the user. No third-party brokers are used. This means that the MDPDS offer an end-to-end delivery of the products, ensuring their integrity and consistency. The service guarantees that all the products, which the user has requested, will be delivered or, in case of the failure, an appropriate error message will be returned. Moreover, the MDPDS allows specifying different delivery modes. In a pull delivery the response data is stored locally by the service and made available for the further download. Finally, there is also an option to deliver the response data to the remote destination (a push delivery). The response is sent to the third-party system using the user specified transfer protocol. In those both last two cases the MDPDS return to the user links that point to the response and which can be used to directly download the data. Those delivery methods are foreseen to provide the data to multiple users with a sufficient access privileges, interested in the same products. Those modes enable cooperation with legacy systems that expect to receive the data using fixed transfer protocol, like the file transfer protocol (FTP). However, the MDPDS does not guarantee that the user eventually downloads the response from the third-party system nor that the third-party system would provide a reliable transfer.



**Figure 2‑9 A stream product request.**

In the Figure 2‑9 there is illustrated a process of subscribing to the product stream. The stream containing instances of the requested product is already created in Product Management service, so the only action left to execute is to subscribe to it to receive updates. Updates are sent till stream is not closed, is 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 Product Managements service, so then all subscriptions become annulled and all the subscribers stop receiving any updates. The same situation happens when the stream expires, where the expiry date is one of the parameters of the stream creation. Finally, updates stop being sent to the subscriber when unsubscribe action is called – then only that particular subscriber is affected, the others still receive the upcoming data.

In the stream mode the MDPDS always deliver products directly to the user, ensuring their integrity and consistency.

# Specification: MISSION DATA PRODUCT DISTRIBUTION SERVICES

## Rationale

The MDPDS specification defines means for distribution of mission data products. Therefore this chapter 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 patterns of the interaction between objects.

## Functional Requirements

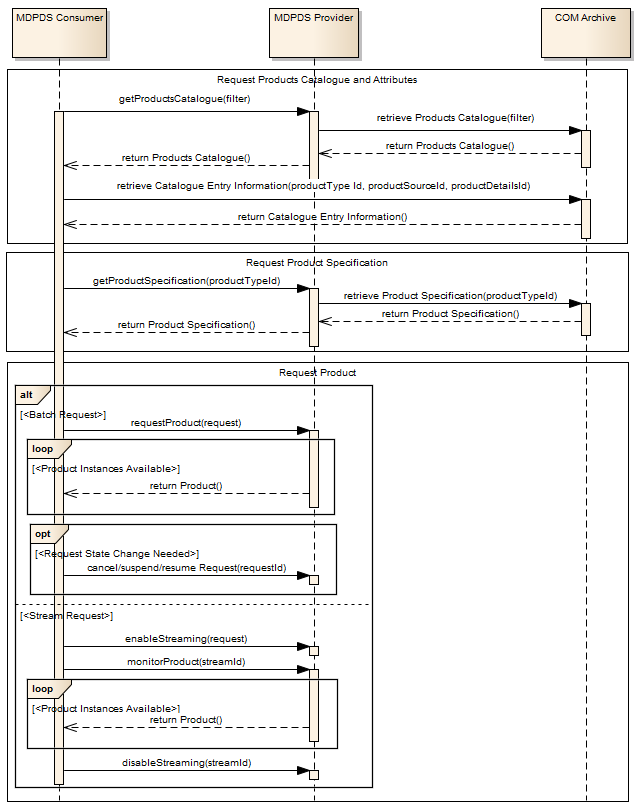
This Recommended Standard specifies an interface to distribute mission data products, which meets the following requirements:

| **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 fulfils the product request and provide it to the consumer in a finite number of chunks. The number of the 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, where 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 |
| **Stream Request** | The MDPDS provider shall generate a stream of mission data product instances that fulfil the product request and provide them to the consumer in a stream of messages. Once new mission data product becomes available, the provider shall send it immediately to all subscribed consumers. The MDPDS shall assign a unique id to each product stream. |
| **Query Product Requests** | The MDPDS provider shall allow generating a list of historical requests of mission data products.  The MDPDS shall allow the consumer to filter requests of mission data products by:   * Time range * Status * Account name |
| **Control Request Execution** | The MDPDS shall allow the consumer to suspend a not completed request.  The MDPDS shall allow the consumer to resume a request.  The MDPDS shall allow the consumer to terminate a not completed request. |
| **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 modifying 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 address. 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 * 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 operations of filtering or sorting a product. The MDPDS consumer shall be able to remove a product attribute, |
| **Definition of standard products** | The MDPDS shall define structures of typical product types. |
| **Definition of standard formats** | The MDPDS shall define typical formats of products. |

## Service: MissionDataProductDistribution

### General

The Mission Data Product Distribution service provides operations to retrieve a catalogue of available products and to request selected products either in the batch or in the stream provisioning mode from one of configured data sources. It provides also mechanisms to check a 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 MDPD service is following (Figure 3‑1). By invoking the getProductsCatalogue() operation, the Consumer retrieves from the Provider a list of product catalogue entries, which he/she may be interested in. To limit a number of potential matches, criteria that the response must meet may be provided. This is annotated in the picture by 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 to the Provider a customized product request in either a batch or a stream provisioning mode. The request specifies all the conditions that the product to retrieve must meet. In case of the batch mode, 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 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 for 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](#_OPERATION:_requestProduct) | 1 | No | 1 |
| SUBMIT | [suspendRequest](#_OPERATION:_suspendRequest) | 2 | No |
| SUBMIT | [resumeRequest](#_OPERATION:_suspendRequest) | 3 | No |
| SUBMIT | [cancelRequest](#_OPERATION:_cancelRequest) | 4 | No |
| REQUEST | [getRequestStatus](#_OPERATION:_getRequestStatus) | 5 | No | 2 |
| REQUEST | [queryProductRequest](#_OPERATION:_queryProductRequest) | 6 | No |
| PUBLISH-SUBSCRIBE | [monitorProduct](#_OPERATION:_monitorProduct) | 7 | No | 3 |
| REQUEST | [getProductsCatalogue](#_OPERATION:_getProductsCatalogue) | 8 | No | 4 |
| REQUEST | [getProductSpecification](#_OPERATION:_getProductSpecification) | 9 | No |

### HIGH LEVEL REQUIREMENTS

1. The Mission Data Product Distribution service shall provide:
   1. the capability for requesting a product in two provisioning modes;
   2. the capability for retrieving a status of request execution;
   3. the capability for retrieving parameters of historic requests;
   4. the capability for controlling the execution of a request;
   5. the capability for retrieving the catalogue of products;
   6. the capability for retrieving the specification of a product;
   7. the capability for retrieving the attributes of a product;
   8. the capability for storing the response locally;
   9. the capability for sending the response to the remote destination.
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. Each product request in the batch mode shall allow specifying:
   1. a product type;
   2. a product format;
   3. a product source;
   4. a destination (optional);
   5. a time range (optional) in relation to the UTC time zone;
   6. a filter (optional);
   7. a schedule (optional) in relation to the UTC time zone;
   8. an encryption (optional);
   9. a compression (optional);
   10. a field to sort the product (optional);
   11. a chunk size (optional).
4. Each product request in the stream mode shall allow specifying:
   1. a product type;
   2. a product format;
   3. a product source;
   4. a filter (optional);
   5. an encryption (optional);
   6. a compression (optional);
   7. expiry date (optional).
5. Each item in the catalogue of products shall contain:
   1. a product type;
   2. a product source;
   3. a product provisioning detail.
6. The Mission Data Product Distribution service shall use the COM archive to store objects.

### FUNCTIONAL REQUIREMENTS

1. If a product is not defined or the invalid provisioning mode is selected, the corresponding request is cancelled and error risen.

### COM usage

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

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](#_Composite:_BatchProductRequestDetai) | Login::LoginInstance |

### COM Event Service usage

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

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

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

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 | ProductsManagement::5 |  |
| ProductUnavailable | 102 | Not used | ProductsManagement::5 |  |

### COM Object Relationships

The Figure below shows the COM object and event relationships for this service:

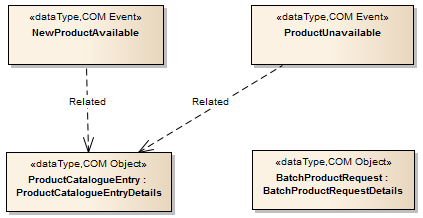


Figure 3‑2: The Mission Data Product Distribution service COM object and event relationships

### COM Archive Service usage

BatchProductRequest objects shall be stored in the COM archive.

### OPERATION: requestProduct

#### General

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](#_Composite:_BatchProductRequestDetai)) |
| OUT | ACK | requestId : (MAL::Long)  subRequestIds : (List<MAL::Long>)  productSize : (MAL::Long) |
| OUT | UPDATE | requestId : (MAL::Long)  product : (MAL::Element) |
| OUT | RESPONSE | transferReport : ([TransferReport](#_Composite:_TransferReport)) |

#### Structures

1. The request field shall contain the BatchProductRequestDetails.
2. The productTypeId field of the supplied BatchProductRequestDetails structure shall match Product Type object instance identifier in the COM archive; otherwise an INVALID\_PRODUCT\_TYPE error shall be raised.
3. The productSourceId field of the supplied BatchProductRequestDetails structure shall match Product Source object instance identifier in the COM archive; otherwise an INVALID\_PRODUCT\_SOURCE error shall be raised.
4. The productFormatId field of the supplied BatchProductRequestDetails structure shall match Product Format object instance identifier in the COM archive; otherwise an INVALID\_PRODUCT\_FORMAT error shall be raised.
5. If for the selected Product Type, Product Source, and Product Format, the corresponding ProductCatalogueEntryDetails does not support the batch mode, a PRODUCT\_NOT\_SUPPORTED\_IN\_BATCH error shall be returned.
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 than the default (the MDPDS Consumer), then it shall contain the DestinationDeliveryDetails structure. Otherwise it shall be NULL.
7. If the destination field is not NULL, then the *uri* field of the DestinationDeliveryDetails structure shall contain fully-qualified address of the destination system in the form of the URI (RFC 3986). The *uri* shall specify the protocol used for the data transfer, the identifier of the destination system and optional parameters, like a desired filename used for saving the response. A reserved keyword “localhost” shall be used in the address part to indicate that the response data shall be stored locally by the MDPDS. If the specified filename is invalid, an INVALID\_FILE\_NAME error shall be returned.
8. If the destination field is not NULL, then the *securityToken* field of the DestinationDeliveryDetails structure may contain authentication and authorization details of the user in the remote system.
9. If the destination field is not NULL, then the *protocolAttributes* field of the DestinationDeliveryDetails structure may contain any attributes required by the transfer protocol.
10. If the *uri* is not in the form of the URI (RFC 3986) or the destination does not exist, a DESTINATION\_UNKNOWN error shall be returned. If an invalid transfer protocol was specified, a TRANSFER\_PROTOCOL\_NOT\_SUPPORTED error shall be returned.
11. If the *securityToken* field contains invalid authentication attributes, an AUTHENTICATION\_FAIL error shall be returned. If the authentication details are not sufficient to execute requested operation, an AUTHORISATION\_FAIL shall be returned.
12. If the *protocolAttributes* field contains invalid transfer protocol attributes, an INVALID\_TRANSFER\_PROTOCOL\_ATTRIBUTES error shall be returned.
13. 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.
14. If endTime is earlier than startTime, an INVALID\_TIME\_RANGE shall be returned.
15. If startTime is NULL, the request concerns the time range from the earliest available mission data product to the endTime value.
16. If endTime is NULL, the request concerns the time range from the startTime value to the current time.
17. If startTime or endTime value is higher than the current time, an INVALID\_TIME\_RANGE shall be returned.
18. The requestId field in the acknowledge message shall contain the object instance identifier of the Request.
19. 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.
20. The productSize field shall contain estimated size of the product response. If the size is unknown, it shall be NULL.
21. The requestId field in the update message shall contain the object instance identifier of the Request related to the requested product.
22. The product field shall contain the matching product.
23. The product shall match all filter criteria. This forms a logical AND operation.
24. If there is no product matching filter criteria, then update messages shall not be sent.
25. If the destination field of the BatchProductRequestDetails is used, then 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.
26. The response message shall contain the TransferReport.

#### Errors

The operation may return one of the following errors:

##### ERROR: DESTINATION\_UNKNOWN

The host specified in the destination field is unreachable.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| DESTINATION\_UNKNOWN | Defined in MAL | Not Used |

##### ERROR: DELIVERY\_FAILED

The product could not be delivered due to a communication error.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| DELIVERY\_FAILED | Defined in MAL | Not Used |

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### 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 |

##### 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 |

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

##### 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 |

##### 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 |

##### ERROR: SCHEDULING\_NOT\_SUPPORTED

The request execution scheduling is not supported by the implementation.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| SCHEDULING\_NOT\_SUPPORTED | 13 | Not Used |

##### 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 |

##### 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 |

##### ERROR: PRODUCT\_NOT\_SUPPORTED\_IN\_BATCH

1. The product is not supported in the batch mode for the selected combination of the Product Type, the Product Source and the Product Format.
2. The ExtraInfo field shall contain 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 |

##### 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 |

##### 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 |

##### 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 |

##### ERROR: INVALID\_FILE\_NAME

The file name specified for saving the response is invalid. 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 |

##### 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 |

##### ERROR: INVALID\_PRODUCT\_ATTRIBUTE

1. One or more filter attributes are invalid. 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 returns the first invalid attribute.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_ATTRIBUTE | 6 | MAL::Attribute |

##### ERROR: INVALID\_TIME\_RANGE

The date or date format in timeRange is invalid.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_TIME\_RANGE | 7 | Not Used |

##### ERROR: INVALID\_SCHEDULE

1. Schedule data type contains invalid date, invalid date format or wrong expiry date.
2. The ExtraInfo field returns the first invalid parameter.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_SCHEDULE | 8 | MAL::Attribute |

### OPERATION: suspendRequest

#### General

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

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

#### Structures

1. The requestId field shall contain the object instance identifier of the Request.
2. If the requestId field equals 0, all not completed requests are suspended.
3. If requestId does not exist, an INVALID\_REQUEST\_ID error shall be returned.
4. In case of the scheduled request, the main requestId may be used to suspend all sub-requests.
5. In case of the scheduled request, suspending one sub-request does not affect other sub-requests.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_REQUEST\_ID

The request object instance identifier does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_REQUEST\_ID | 8 | Not Used |

### OPERATION: resumeRequest

#### General

The resume 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) |

#### Structures

1. The requestId field shall contain the object instance identifier of the Request.
2. If requestId does not exist, an INVALID\_REQUEST\_ID error shall be returned.
3. If the requestId field equals 0, all suspended requests are resumed.
4. In case of the scheduled request, the main requestId may be used to resume all sub-requests.
5. In case of the scheduled request, resuming one sub-request does not influence other sub-requests.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_REQUEST\_ID

The request object instance identifier does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_REQUEST\_ID | 8 | Not Used |

### OPERATION: cancelRequest

#### General

The cancel operation cancels the execution of the current or the scheduled request. Not delivered data is lost.

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

#### Structures

1. The requestId field shall contain the object instance identifier of the Request.
2. If requestId does not exist, an INVALID\_REQUEST\_ID error shall be returned.
3. If the requestId field equals 0, all not completed requests are cancelled.
4. In case of the scheduled request, the main requestId may be used to cancel all sub-requests.
5. In case of the scheduled request, cancelling one sub-request does not influence other sub-requests.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_REQUEST\_ID

The request object identifier does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_REQUEST\_ID | 8 | Not Used |

### OPERATION: getRequestStatus

#### General

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](#_Composite:_RequestStatus)>) |

#### Structures

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.
2. The filter may contain multiple criteria combined by a logical AND operator.
3. The filter filed may contain comparison operators, ranges of values and multiple wildcard characters.
4. If more than one filter parameter is used, the results shall be combined using logical conjunction.
5. The response shall contain a list of statuses of the requests, which match selected filter criteria.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_ATTRIBUTE

One or more filter attributes are invalid. The attribute is invalid when its name or type is different than 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.

The first invalid attribute is returned.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_ATTRIBUTE | 6 | MAL::Composite |

##### 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 |

### OPERATION: queryProductRequest

#### General

The queryProductRequest operation returns the content of requests which 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](#_Composite:_BatchProductRequestDetai)>) |

#### Structures

1. The filter indicates conditions that the response with the request must meet. The response can 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.
2. The filter may contain comparison operators, ranges and multiple wildcard characters.
3. If more than one filter parameter is used, the results shall be combined using logical operators.
4. The response shall contain the list of requests that match all filter criteria. This forms a logical AND operation.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_ATTRIBUTE

One or more filter attributes are invalid. The attribute is invalid when its name or type is different than 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.

The first invalid attribute is returned.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_ATTRIBUTE | 6 | MAL::Composite |

##### 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 |

### OPERATION: monitorProduct

#### General

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:

1. the value of the firstSubKey is NULL;
2. the value of the secondSubKey is identifier of the stream;
3. the value of the thirdSubKey is NULL;
4. 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) |

#### Structures

1. The product field shall contain the matching product.

#### Errors

The operation may return one of the following errors:

##### ERROR: DELIVERY\_FAILED

The product could not be delivered due to a communication error.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| DELIVERY\_FAILED | Defined in MAL | Not Used |

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_STREAM\_ID

The subscription identifier is invalid.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_STREAM\_ID | 10 | Not Used |

### OPERATION: getProductsCatalogue

#### General

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](#_Composite:_ProductCatalogueEntryDet)>) |

#### Structures

1. The filter field may contain parameters to filter a catalogue of products by any field of ProductTypeDetails, ProductProvisioningDetails or ProductSource structures.
2. The filter may contain multiple criteria.
3. The filter may contain comparison operators, ranges of values and wildcard characters.
4. If more than one filter parameter is used, the results shall be combined using logical operators.
5. If the filter field is NULL, the response shall contain the list of all available products.
6. The productsCatalogue field shall contain a list of ProductCatalogueEntryDetails objects.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_ATTRIBUTE

1. One or more filter attributes are invalid. The attribute is invalid when its name or type is different than any field of the ProductTypeDetails, ProductProvisioningDetails or ProductSource structures, the selected comparison operation is not allowed or the attribute value does not match the attribute type.
2. The ExtraInfo field returns the first invalid attribute.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_ATTRIBUTE | 6 | MAL::Attribute |

### OPERATION: getProductSpecification

#### General

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) |

#### Structures

1. The productTypeId field shall contain the object instance identifier of the ProductTypeDetails.
2. If productTypeId does not exist, an INVALID\_PRODUCT\_TYPE error shall be returned.
3. The response shall contain a specification of the product in the form of a MAL::File structure.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

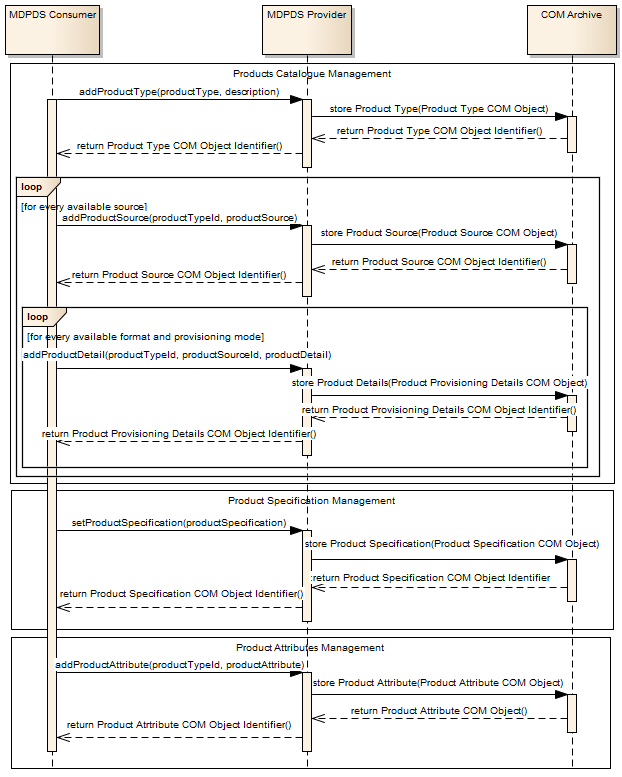
## Service: ProductsManagement

### General

The Products Management service allows the consumer to set up the Mission Data Product Distribution service. All products which the system supports must be defined prior they can be obtained. Each product is characterized by its:

* type,
* format
* source
* internal structure
* attributes
* provision modes
* compression
* encryption formats

The product type, the product source and the product provisioning details constitute ProductCatalogueEntry, 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 as follows (Figure3‑3). For each available product, the Consumer sends to the Provider a series of messages that result in creation of a product catalogue entry in the COM Archive. First message, passed by 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 where particular type of product can be obtained. For each combination of the product type and the source, the Consumer calls addProductDetail() operation to specify formats, provisioning modes and optionally other details describing the product. When a unique triple of the product type, source and details has been provided, the Provider automatically creates catalogue entry. Next step is concerned with setting the product specification by executing setProductSpecification() operation for each particular product. This action is recommended, but is 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 has been defined, usage of filter or sort fields in a request for this product will cause an error.

**Table 3‑4 Products Management Service Operations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Identifier | Service Identifier | Area Number | Service Number | Area Version |
| distribution | ProductsManagement | 9 | 2 | 1 |
| Interaction Pattern | Operation Identifier | Operation Number | Support in Replay | Capability Set |
| REQUEST | [addProductType](#_OPERATION:_addProductType) | 1 | No | 1 |
| SUBMIT | [removeProductType](#_OPERATION:_removeProductType) | 2 | No |
| REQUEST | [addProductSource](#_OPERATION:_addProductSource) | 3 | No | 2 |
| SUBMIT | [removeProductSource](#_OPERATION:_removeProductSource) | 4 | No |
| REQUEST | [addProductFormat](#_OPERATION:_addProductFormat) | 5 | No | 3 |
| SUBMIT | [removeProductFormat](#_OPERATION:_removeProductFormat) | 6 | No |
| REQUEST | [addProductDetail](#_OPERATION:_addProductDetail) | 7 | No | 4 |
| SUBMIT | [removeProductDetail](#_OPERATION:_removeProductDetail) | 8 | No |
| REQUEST | [setProductSpecification](#_OPERATION:_setProductSpecification) | 9 | No | 5 |
| SUBMIT | [removeProductSpecification](#_OPERATION:_removeProductSpecificati) | 10 | No |
| REQUEST | [addProductAttribute](#_OPERATION:_addProductAttribute) | 11 | No | 6 |
| SUBMIT | [removeProductAttribute](#_OPERATION:_removeProductAttribute) | 12 | No |
| REQUEST | [enableStreaming](#_OPERATION:_enableStreaming) | 13 | No | 7 |
| SUBMIT | [disableStreaming](#_OPERATION:_disableStreaming) | 14 | No |

### HIGH LEVEL REQUIREMENTS

1. The Products Management service shall provide:
   1. the capability for maintaining the catalogue of products;
   2. the capability for maintaining the list of product specification;
   3. the capability for maintaining the list of product attributes.
2. It shall be possible to add and delete:
   1. product types;
   2. product sources;
   3. product provisioning details;
   4. product attributes.
3. It shall be possible to set and delete product specification.
4. The Products Management shall use the COM archive service to store:
   1. a product type;
   2. a product provisioning details;
   3. a product source;
   4. a product catalogue entry;
   5. a product specification;
   6. a product attribute.

### FUNCTIONAL REQUIREMENTS

1. A product source can be added only to the existing product type.
2. When a product source is removed, corresponding definitions of product provisioning details shall be also removed.
3. A product provisioning detail can be added only to the existing product type and the product source.
4. When a product type is removed, corresponding definitions of product sources and product provisioning details shall be also removed.
5. Only one product specification can be assigned to the product type.
6. A product attribute can be added only if it is defined in a product specification.
7. An URI field of a product source shall be an address in the form of the URI (RFC 3986).

### COM usage

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.

A ProductSource COM object represents a source of mission data product. A ProductSource COM object body shall hold an URI address of a source.

The ProductSource COM object related link shall indicate which ProductType object it uses.

A ProductFormat COM object represents a format of mission data product. The ProductFormat COM object shall hold a name and a description of the format. 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

A ProductProvisioningDetailsInstance COM object represents detailed parameters availability of the mission data product. The ProductProvisioningDetailsInstance COM object body shall contain the ProductProvisioningDetails structure, which holds the format, provision mode, compression and encryption algorithms applicable to mission data product.

The ProductProvisioningDetailsInstance COM object related link shall indicate for which ProductType provisioning details are provided.

The ProductProvisioningDetailsInstance COM object source link shall indicate for which ProductSource provisioning details are provided.

The ProductCatalogueEntryDetails COM object represents a description of the Product.

A ProductCatalogueEntry COM object body shall contain the ProductCatalogueDetailsEntry structure, which holds a product type, attributes, available sources, formats and provisioning modes.

A ProductSpecification COM object represents a specification of mission data product. A ProductSpecification COM object body shall hold a File object.

The ProductSpecification COM object related link shall indicate which ProductType object it uses.

A ProductAttribute COM object represents a product attribute. 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.

The ProductAttribute COM object related link shall indicate which ProductType object it uses.

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

**Table 3‑5 Products Management Service Object Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object Name | Object Number | Object Body Type | Related points to | Source Object |
| ProductType | 1 | [ProductTypeDetails](#_DATATYPE_ProductTypeDetails) |  |  |
| ProductSource | 2 | MAL::URI | 1 |  |
| ProductFormat | 3 | [ProductFormatDetails](#_DATATYPE_ProductFormatDetails) |  |  |
| ProductProvisioningDetailsInstance | 4 | [ProductProvisioningDetails](#_DATATYPE_ProductProvisioningDetails) | 1 | 2 |
| ProductCatalogueEntry | 5 | [ProductCatalogueEntryDetails](#_DATATYPE_ProductCatalogueEntryDetails) |  |  |
| ProductSpecification | 6 | MAL::File | 1 |  |
| ProductAttribute | 7 | [ProductAttributeDetails](#_DATATYPE_ProductAttributeDetails) | 1 |  |
| StreamProductRequest | 8 | [StreamProductRequestDetails](#_DATATYPE_StreamProductRequestDetails) |  |  |

### COM Event Service usage

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Table 3‑6 Products Management Service Events**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event Name | Object Number | Object Body Type | Related points to | Source points to |
| ProductTypeCreated | 101 | MAL::Long |  |  |
| ProductTypeRemoved | 102 | MAL::Long | 5 |  |
| ProductSourceCreated | 103 | MAL::Long | 1 |  |
| ProductSourceRemoved | 104 | MAL::Long | 5 |  |
| ProductFormatCreated | 105 | MAL::Long | 1 |  |
| ProductFormatRemoved | 106 | MAL::Long |  |  |
| ProductDetailCreated | 107 | MAL::Long | 1 | 5 |
| ProductDetailRemoved | 108 | MAL::Long | 5 |  |
| ProductSpecificationCreated | 109 | MAL::Long | 1 |  |
| ProductSpecificationRemoved | 110 | MAL::Long | 1 |  |
| ProductAttributeCreated | 111 | MAL::Long | 1 |  |
| ProductAttributeRemoved | 112 | MAL::Long | 1 |  |
| ProductStreamCreated | 113 | MAL::Long |  |  |
| ProductStreamClosed | 114 | MAL::Long |  |  |

### COM Object Relationships

The Figure below shows the COM object and event relationships for this service:

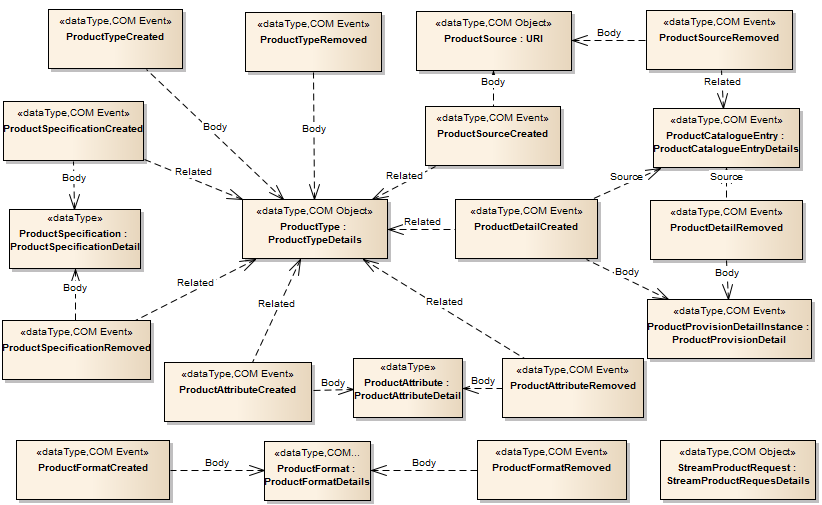


Figure 3‑4: The Products Management service COM object and event relationships

### COM Archive Service usage

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

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

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

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

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

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

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

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

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

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

When the ProductType is removed, the corresponding ProductCatalogueEntry, ProductSource and ProductProvisioningDetailsInstance are removed as well.

When the ProductSource is removed, the corresponding ProductCatalogueEntry and ProductProvisioningDetailsInstance are removed as well.

When the ProductProvisioningDetailsInstance is removed, the corresponding ProductCatalogueEntry is removed.

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

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

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

### OPERATION: addProductType

#### General

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, e.g. 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) |

#### Structures

1. The productType field shall contain a self-descriptive name of the product type.
2. The description field may contain additional information about the product type.
3. The productTypeId field shall contain an instance identifier of the product type.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: DUPLICATE

1. The product type has already been defined.
2. ExtraInfo Type field contains an instance identifier of the already existing product type.

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

### OPERATION: removeProductType

#### General

The removeProduct 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) |

#### Structures

1. The productTypeId field shall contain the object instance identifier of the product type.
2. If productTypeId does not exist, an INVALID\_PRODUCT\_TYPE error shall be returned.
3. The ProductCatalogueEntry objects which contain the identifier of the product type shall be removed from the COM.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

### OPERATION: addProductSource

#### General

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) |

#### Structures

1. The productTypeId field shall contain the object instance identifier of the product type.
2. If productTypeId does not exist, an INVALID\_PRODUCT\_TYPE error shall be returned.
3. The productSource field shall contain an address in the form of the URI (RFC 3986).
4. The productSourceId field shall contain the object instance identifier of the product source.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: DUPLICATE

1. The product source at the selected URI address is already defined.
2. ExtraInfo Type field contains an instance identifier of the already existing product source.

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

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

##### ERROR: INVALID\_PRODUCT\_SOURCE

The specified product source is not valid URI address.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_SOURCE | 2 | Not Used |

### OPERATION: removeProductSource

#### General

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) |

#### Structures

1. The productSourceId field shall contain an instance identifier of the product source.
2. The ProductCatalogueEntry objects which contain the identifier of the product source shall be removed from the COM.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_SOURCE

The selected product source does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_SOURCE | 2 | Not Used |

### OPERATION: addProductFormat

#### General

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) |

#### Structures

1. The productFormat field shall contain a unique name of the format of the mission data product.
2. If a product format which contains the same value of the productFormat field already exists, a DUPLICATE error shall be raised.
3. The description field may contain a short description of the product format.
4. The productFormatId field shall contain an object instance identifier of the product format.

#### Errors

The operation may return one of the following errors:

##### ERROR: DUPLICATE

The given set of product details has already been defined.

ExtraInfo Type contains an instance identifier of the already existing product type.

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

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

### OPERATION: removeProductFormat

#### General

The removeProductFormat operation removes the selected product format.

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

#### Structures

1. The productFormatId field shall contain the object instance identifier of the ProductFormat.
2. If the last ProductFormat is removed from the related ProductDetail, the ProductCatalogueEntry objects which contain the identifier of the ProductDetail shall be removed from the COM.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_FORMAT

The specified product format does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_FORMAT | 3 | Not Used |

### OPERATION: addProductDetail

#### General

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 triple of the product type, the product source and the 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 : ([ProductProvisioningDetail](#_Composite:_ProductProvisioningDetai)s) |
| OUT | RESPONSE | productDetailId : (MAL::Long) |

#### Structures

1. The productTypeId field shall contain the object instance identifier of the product type.
2. If productTypeId does not exist, an INVALID\_PRODUCT\_TYPE error shall be returned.
3. The productSourceId field shall contain the object instance identifier of the product source.
4. If productSourceId does not exist, an INVALID\_PRODUCT\_SOURCE error shall be returned.
5. The productDetail shall contain the details about particular format of the product, its provision modes, compression and encryption methods.
6. If a product detail, which contains the same value of productFormat field, already exists for the selected product type and product source, a DUPLICATE error shall be raised.
7. The productDetailId field shall contain an object instance identifier of the ProductProvisioningDetailInstance.

#### Errors

The operation may return one of the following errors:

##### ERROR: DUPLICATE

The given set of product details has already been defined.

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

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

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

##### ERROR: INVALID\_PRODUCT\_SOURCE

The specified product source does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_SOURCE | 2 | Not Used |

### OPERATION: removeProductDetail

#### General

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) |

#### Structures

1. The productDetailId field shall contain the object instance identifier of the ProductDetail.
2. The ProductCatalogueEntry objects which contain the object instance identifier of the ProductDetail shall be removed from the COM.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_DETAIL

1. The selected product detail does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_DETAIL | 4 | Not Used |

### OPERATION: setProductSpecification

#### General

The setProductSpecification operation sets a specification of what the selected product consists. If a specification has been already set up, 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) |

#### Structures

1. The productTypeId field shall contain the object instance identifier of the Product Type.
2. If productTypeId does not exist, an INVALID\_PRODUCT\_TYPE error shall be returned.
3. The productSpecification field shall contain a specification of the product.
4. The productSpecificationId field shall contain an object instance identifier of the ProductSpecification.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

##### ERROR: INVALID\_PRODUCT\_SPECIFICATION

The structure describing the product type is invalid.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_SPECIFICATION | 5 | Not Used |

### OPERATION: removeProductSpecification

#### General

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) |

#### Structures

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

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_SPECIFICATION

The selected product specification does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_SPECIFICATION | 5 | Not Used |

### OPERATION: addProductAttribute

#### General

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 : ([ProductAttribute](#_DATATYPE_ProductAttribute)Details) |
| OUT | RESPONSE | productAttributeId : (MAL::Long) |

#### Structures

1. The productTypeId field shall contain the object instance identifier of the product type.
2. If productTypeId does not exist, an INVALID\_PRODUCT\_TYPE error shall be returned.
3. The productAttribute field shall contain a new attribute of the product.
4. The productAttributeId field shall contain an object instance identifier of the product attribute.

#### Errors

The operation may return one of the following errors:

##### ERROR: DUPLICATE

The name field of Product Attribute already exists for the selected Product Type.

The ExtraInfoType field contains an object instance identifier of the duplicated attribute.

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

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_TYPE

The specified product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

### OPERATION: removeProductAttribute

#### General

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) |

#### Structures

1. The productAttributeId field shall contain the object instance identifier of the product attribute.
2. If productAttributeId does not exist, an INVALID\_PRODUCT\_ATTRIBUTE error shall be returned.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_ATTRIBUTE

1. The attributeName field shall contain the name of the attribute.
2. The Extra info returns the first invalid attribute.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_ATTRIBUTE | 6 | MAL::Attribute |

### OPERATION: enableStreaming

#### General

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 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](#_Composite:_StreamProductRequestDeta)) |
| OUT | RESPONSE | streamId : (MAL::Long) |

#### Structures

1. The request field shall contain the StreamProductRequestDetails.
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. 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.
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.
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.
6. The streamId field in the acknowledge message shall contain the object instance identifier of the created product stream.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_PRODUCT\_TYPE

The selected product type does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_TYPE | 1 | Not Used |

##### 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 |

##### 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 |

##### ERROR: PRODUCT\_NOT\_SUPPORTED\_IN\_STREAM

1. The product is not supported in the stream mode for the specified combination of a product type, a format and a source.
2. The ExtraInfo fiels contains the instance identifier of the ProductCatalogueEntryDetails which 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 |

##### 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 |

##### ERROR: INVALID\_PRODUCT\_FORMAT

The format is not supported for the specified product type.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_FORMAT | 3 | Not Used |

##### ERROR: INVALID\_PRODUCT\_ATTRIBUTE

1. One or more filter attributes are invalid. 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 first invalid attribute is returned.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_PRODUCT\_ATTRIBUTE | 6 | MAL::Attribute |

### OPERATION: disableStreaming

#### General

The disableStreaming operation closes the product stream.

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

#### Structures

1. The streamId field shall contain the object instance identifier of the stream.
2. If requestId does not exist, an INVALID\_STREAM\_ID error shall be returned.
3. If the requestId field equals 0, all the streams are closed.

#### Errors

The operation may return one of the following errors:

##### ERROR: UNKNOWN

An unknown error has occurred.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| UNKNOWN | Defined in MAL | Not Used |

##### ERROR: INVALID\_STREAM\_ID

The stream object instance identifier does not exist.

|  |  |  |
| --- | --- | --- |
| Error | Error # | ExtraInfo Type |
| INVALID\_STREAM\_ID | 10 | Not Used |

# Data types

## Area data types: distribution

### ENUMERATION: ProvisionMode

The ProvisionMode enumeration holds 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. |

### ENUMERATION: StateEnum

The StateEnum is an enumeration holding 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. |

### ENUMERATION: CompressionAlgorithmEnum

The CompressionAlgorithmEnum enumeration holds the supported compression algorithms which may be applied to the requested product. Numerical values assigned to algorithms shall be used in product requests. 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. |

### ENUMERATION: EncryptionFormatEnum

The EncryptionFormatEnum enumeration holds the data encryption algorithms which may be applied to the requested Product. Numerical values assigned to formats shall be used in product requests. 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. |

### ENUMERATION: MalTypesEnum

The MalTypesEnum enumerates 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 |

### Composite: ProductTypeDetails

The ProductTypeDetails structure contains the Product Type and a short description. It can additionally 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. |

### Composite: ProductFormatDetails

The ProductFormatDetails structure keeps the format in which any product can occur. It contains 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. |

### Composite: ProductProvisioningDetails

The ProductProvisioningDetails structure keeps a group of parameters of the related product. It contains 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](#_DATATYPE_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. |

### Composite: ProductAttributeDetails

The ProductAttributeDetails structure contains a single product attribute. It stores a name of the attribute, a type of data, and possibility to filter or sort by the attribute. 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](#_DATATYPE_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. |

### Composite: ProductCatalogueEntryDetails

The ProductCatalogueEntryDetails defines an available product in terms of a type, a source, a format and additional details. The structure is used to store object instance identifiers of the ProductType, the ProductSource and the ProductProvisioningDetailsInstance. To retrieve referenced objects, the COM Archive shall be used.

|  |  |  |  |
| --- | --- | --- | --- |
| 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. |

### Composite: TimeRange

The TimeRange structure represents a period from the start time to the end time, inclusive.

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

### Composite: Schedule

The Schedule structure is used to set up parameters 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 the time in the UTC time zone when a transmission shall be started. |
| repeatable | MAL::Boolean | Yes | The flag, which 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. |

### Composite: BatchProductRequestDetails

The BatchProductRequestDetails contains details of a product request. It specifies criteria, which product to retrieve 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](#_Composite:_DestinationDeliveryDetai) | 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](#_Composite:_TimeRange) | Yes | The period of the time, from 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](#_Composite:_Schedule) | Yes | The structure, which 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. |
| sortField | MAL::String | Yes | The name of the field to use 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 - in descending one. |
| chunkSize | MAL::Long | Yes | The maximum size of the response message. If the response message exceeds the specified value, data is split into chunks. |

### Composite: DestinationDeliveryDetails

The DestinationDeliveryDetails structure is used to provide details needed for the local or remote product delivery. The structure contains 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 (RFC 3986). 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](#_Composite:_ProtocolAttributeDetails) | Yes | The protocolAttributes is an abstract structure, which may contain any additional attributes required by the protocol or remote destination to handle the data transfer. |

### Composite: ProtocolAttributeDetails

The ProtocolAttributeDetails is an abstract structure, which may contain any additional attributes required by the protocol or remote destination to handle the data transfer.

|  |  |
| --- | --- |
| Name | ProtocolAttributeDetails |
| Extends | MAL::Composite |
| Short Form Part | 10 |

### Composite: RemoteResponse

The RemoteResponse structure is used to provide a location, a size and a delivery date of the product response. The 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 (RFC 3986). |
| 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. |

### Composite: StreamProductRequestDetails

The StreamProductRequestDetails contains details of a stream product request. It specifies criteria, which product to retrieve 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. |

### Composite: TransferReport

The TransferReport structure contains a summary of 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. |

### Composite: RequestStatus

The RequestStatus structure is used to store 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](#_ENUMERATION:_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](#_Composite:_Schedule) | Yes | Parameters of the schedule of the request. |

### Composite: ParameterValueTimePair

The ParameterValueTimePair structure is used to store 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. |

### Composite: ParameterTimeline

The ParameterTimeline structure holds values of the specific parameter along with their timestamps.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | ParameterTimeline | | |
| Extends | MAL::Composite | | |
| Short Form Part | 202 | | |
| Field | Type | Nullable | Comment |
| timestampedParameterValues | List<[ParameterValueTimePair](#_DATATYPE_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. |

### Composite: AggregationValueTimePair

The AggregationValueTimePair structure is used to store 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. |

### Composite: AggregationTimeline

The AggregationTimeline structure holds values of the specific aggregation along with their timestamps.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | AggregationTimeline | | |
| Extends | MAL::Composite | | |
| Short Form Part | 204 | | |
| Field | Type | Nullable | Comment |
| timestampedAggregationValues | List<[AggregationValueTimePair](#_DATATYPE_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. |

### Composite: ActionStageTimePair

The ActionStageTimePair is used to store 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. |

### Composite: ActionInstanceReport

The ActionElement structure is used to report on the status of the each verification of the action instance. 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](#_DATATYPE_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. |

### Composite: ActionsHistory

The ActionsHistory structure holds a list of ActionInstanceReports.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | ActionsHistory | | |
| Extends | MAL::Composite | | |
| Short Form Part | 207 | | |
| Field | Type | Nullable | Comment |
| actions | List<[ActionInstanceReport](#_DATATYPE_ActionInstanceReport)> | No | A time-tagged history of actions. |

### Composite: AlertReport

The AlertReport structure holds 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 AlertDefinitionDetailsId provides the reference to the definition of an alert, including any argument definitions. |
| alertEventDetailsId | MAL::Long | No | The AlertEventDetailsId provides the reference to the structure holding the details of the instance of an alert. |

### Composite: AlertsHistory

The AlertsHistory structure holds a list of AlertReport objects.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | AlertsHistory | | |
| Extends | MAL::Composite | | |
| Short Form Part | 209 | | |
| Field | Type | Nullable | Comment |
| alerts | List<[AlertReport](#_DATATYPE_AlertReport)> | No | A time-tagged history of alerts. |

### Composite: CheckReport

The CheckReport structure holds the parameter check report.

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

### Composite: ChecksHistory

The ChecksHistory structure holds a list of CheckReport objects.

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

### Composite: MultipleParameterTimeline

The MultipleParameterTimeline is used to store a group of ParameterTimeline objects.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MultipleParameterTimeline | | |
| Extends | MAL::Composite | | |
| Short Form Part | 212 | | |
| Field | Type | Nullable | Comment |
| parameterTimelineList | List<[ParameterTimeline](#_DATATYPE_ParameterTimeline)> | No | A list of multiple parameter timelines. |

# Error codes

The following table 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 may not have 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 invalid date, the invalid date format or the 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. |
| 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, a format and a 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. |

[Normative specifications appear in sections 3 through *n*. See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014).

All sections and annexes should be separated by Word continuous section breaks.]

# SERVICE SPECIFICATION XML

The following section defines the service in the XML notation as specified in reference [2].

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

[SANA Registry location to be supplied]

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

[SANA Registry location to be supplied]

1. Implementation Conformance   
   Statement (ICS) Proforma  
     
   (normative)
   1. INTRODUCTION

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

* + 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.

* + 1. 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.

* + 1. 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.

* 1. ICS PROFORMA FOR [SPECIFICATION]
     1. GENERAL INFORMATION
        1. Identification of ICS

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

* + - 1. Identification of Implementation Under Test

|  |  |
| --- | --- |
| Implementation Name |  |
| Implementation Version |  |
| Special Configuration |  |
| Other Information |  |

* + - 1. Identification of Supplier

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

* + - 1. 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 [  ] |

* + 1. REQUIREMENTS LIST

[See CCSDS A20.1-Y-1, *CCSDS Implementation Conformance Statements* (Yellow Book, Issue 1, April 2014).]

1. Security, SANA, and Patent Considerations  
     
   (Informative)
   1. Security Considerations

The security considerations of this specification are the same as those of reference [2]. Specifically authentication and authorisation of a participating consumer or provider is provided by the MAL access control concept and is covered in sections 3.6, 5.2 and 5.3 of the Reference Model (R[5]).

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

* 1. SANA Considerations

The recommendations of this document request SANA populate the registry specified in reference [2] with the schema and XML detailed in section 6 of this document.

As stated in reference [2], the registration rule for change to this registry requires an engineering review by a designated expert. The expert shall be assigned by the WG Chair, or in absence, Area Director.

* 1. Patent Considerations

The recommendations of this document have no patent issues.