INTERFACE CONTROL DOCUMENT

Copernicus Space Component

Long Term Archive Interface Control Document

Reference	**ESA-EOPG-EOPGC-IF-2**
Issue/Revision	**1.9**
Date of Issue	**06/04/2023**
Status	**Issued**

European Space Agency
Agence spatiale européenne

CHANGE LOG

Long Term Archive Interface Control Document	Issue Nr.	Revision Number	Date
2019 Checkpoint comments	1	1	Dec. 2019
Addition of supported Authentication Methods	1	1	Dec. 2019
Traceability interface introduced	1	1	Jan. 2020
OData metadata document added	1	1	Jan. 2020
LTA Extended Compliance test instructions introduced	1	1	Jan. 2020
Alignment of Attribute entities	1	2	Feb. 2020
Updates following LTA ITT Step 2 RFPs and Clarifications	1	3	Jun. 2020
Updates related to the LTAs start of operations	1	4	Sep. 2020
Updates to Subscription examples and Metrics	1	5	Dec. 2020
LTA operations clarifications and “Checkpoint 2020”	1	6	Apr. 2021
Minor examples corrections	1	7	Oct. 2021
Update for “Checkpoint 2022”	1	8	Sept. 2022
Update of Applicable documentation regarding Traceability service	1	9	Apr. 2023

CHANGE RECORD

Issue Number 1	**Revision Number 1**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Updated OData Id property to Guid type	Dec. 2019	Multiple	Multiple
Removed *datetime* from the query examples	Dec. 2019	Multiple	Multiple
Updated Applicable and Reference Documents	Dec. 2019	8, 9	1.3, 1.4
Added supported Authentication Methods	Dec. 2019	16	3.1
Changed CreationDate to PublicationDate and updated tables and examples	Dec. 2019	Multiple	Multiple
Updated the Attribute Entity Description table, removing ContentType and ContentLength and adding ValueType	Dec. 2019	18	Table 2
Updated Query by Attributes examples	Dec. 2019	24	3.3.1.6
Added additional query options: $orderby, $top, $skip, $count	Dec. 2019	25, 21	3.3.1.7
Improved the Catalog export function description	Dec. 2019	27	3.3.2
Added Product Staging Notification for the Product Download and updated Table 3	Dec. 2019	28, 30	3.4, Table 3

Added *queued* status to the Order Status property	Dec. 2019	30	Table 3
Updated list of HTTP status code responses	Dec. 2019	34	3.4.2
Added *cancelled* to the BulkOrder status	Dec. 2019	37	Table 4
Improved BatchsizeProducts and BatchsizeVolume descriptions	Dec. 2019	37	Table 4
Added BatchOrder Query Request and Response examples	Dec. 2019	43	3.5.2.2
Removed LTA capability to store notifications	Dec. 2019	46	3.6
Added Subscription StageOrder Query Request and Response examples	Dec. 2019	47	3.6
Traceability interface introduced	Jan. 2020	53	3.7
OData metadata description added	Jan. 2020	65	Annex 1
LTA Extended Compliance test instructions introduced	Jan. 2020	75	Annex 2

Issue Number 1	**Revision Number 2**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Update to align Attribute entities	Feb. 2020	Multiple	Figure 3, Annex 1

Issue Number 1	**Revision Number 3**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Update to Query by List	Mar. 2020	23	3.3.1.2
OData naming harmonisation	May 2020	Multiple	Multiple
Update to Query by Name	May 2020	23	3.3.1.1
Update to Query by Geographic Criteria	May 2020	24	3.3.1.5
Update to Query by Attributes	May 2020	24	3.3.1.6
Alignment of Order Status Query example	May 2020	27	3.4.1
Update to Bulk Create	May 2020	42	3.5.2.1
Update to BatchOrder Triggering	May 2020	43	3.5.2.2
Change of LastQueryDate to LastNotificationDate	May 2020	43	Figure 9, Table 6
Update to Subscription Create	May 2020	46	3.6
Update to User Administration	May 2020	53, 54	3.8.1, 3.8.1.1
Addition of AUXIP ICD and Glossary to Reference Documents	Jun. 2020	9	1.4
Alignment of DateAttribute to DateTimeOffsetAttribute	Jun. 2020	Multiple	3.2, Annex 1
Modification of “count” to “@odata.count” in the examples to align with OData v4.01	Jun. 2020	Multiple	3.9.1

Modification of ‘substringof’ to ‘contains’ to align with OData v4.01	Jun. 2020	Multiple	Multiple
Alignment of LTA Extended Compliance Test Suite to LTA ICD and LTA Procurement Step 2 Clarifications	Jun. 2020	Multiple	Annex 2 - 5.2
Update to the use of quotation marks in the Request and Response examples	Jun. 2020	Multiple	Multiple

Issue Number 1	**Revision Number 4**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Addition of BooleanAttribute	Aug. 2020	Multiple	3.2, 3.3.1.6, Annex 1
Update to the LTA Summary Reporting Statistics	Aug. 2020	57	Table 11
Update to Authorization and Authentication	Sep. 2020	16	3.1

Issue Number 1	**Revision Number 5**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Update to Subscription examples for the ProductName property	Dec. 2020	46	3.6
Update to Metrics reporting of Gauge and Counter values	Dec. 2020	60	3.9.2

Issue Number 1	**Revision Number 6**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Alignment of Product Name examples	Apr. 2021	Multiple	Multiple
Update of “FilterParam” and “OrderbyParam” examples for the Bulk Properties	Apr. 2021	33	Table 5
Update of “FilterParam” in Bulk create examples	Apr. 2021	42	3.5.2.1
Update to the BatchOrders within a Bulk Query: Example Request (Bulk→Bulks and removal of quotation marks for the Id)	Apr. 2021	43	3.5.2.2
Update to the BatchOrders within a Bulk Query: Example Request (BatchOrder→BatchOrders and removal of quotation marks for the Id)	Apr. 2021	43	3.5.2.2
Update of “FilterParam” example for the Subscription Properties	Apr. 2021	43	Table 6
Update of “FilterParam” Subscription Create examples	Apr. 2021	46	3.6
Update to the parameters collected for products in relation to the Reporting (CreationDate→PublicationDate)	Apr. 2021	56	3.9.1 (Table 10)
Addition of JobStatus “created” value in the OData Metadata Description	Apr. 2021	65	Annex 1

Addition of ServiceAlias property to the User Entity Model	Apr. 2021	53	3.8

Issue Number 1	**Revision Number 7**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Update of query examples containing “FilterParam”	Oct. 2021	Multiple	Multiple
Minor alignments in the examples	Oct. 2021	Multiple	Multiple

Issue Number 1	**Revision Number 8**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Addition of [AD-8] to [AD-11]	Sept. 2022	8	1.3
Addition of GeoFootprint property	Sept. 2022	19	3.2, Table 1
Update of Attribute Entity Description	Sept. 2022	20	3.2, Table 2
Inclusion of $expand=Attributes query example	Sept. 2022	20	3.2
Update of section 3.4 regarding the retrieval of immediately available (online) products	Sept. 2022	29	3.4
Update of section 3.4.2 for the download of online vs offline products	Sept. 2022	34	3.4.2
Update from BatchOrder “Order” action to “BatchOrder” action	Sept. 2022	37	Figure 7
Significant updates and alignment of Annex 1	Sept. 2022	65	Annex 1

Issue Number 1	**Revision Number 9**		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Update of [AD-7] to the new Traceability Service	Apr. 2023	8	1.3

Table of contents:

1 INTRODUCTION 8
1.1 Purpose and Scope 8
1.2 Structure of the Document 8
1.3 Applicable Documents 8
1.4 Reference Documents 9
1.5 Definitions 9
1.6 List of Acronyms 10
2 OVERVIEW 11
2.1 Context 11
2.2 Nominal Use Scenario Description 13
2.3 Ingest Product 14
2.4 Bulk Product Ingestion 14
2.5 Query Product Catalogue 14
2.6 Retrieve Product 14
2.7 Bulk Product Retrieve 14
2.8 Delete Product 14
2.9 Bulk Delete Product 14
2.10 Subscription 15
2.11 Reporting 15
2.12 Registration and Authorisation 15
2.13 Accounting 15
2.14 Traceability 15
3 LONG TERM ARCHIVE CLIENT API DESCRIPTION 16
3.1 Interface Configuration 16
3.2 Basic LTA Product Entity Model 17
3.3 Query Products Catalogue 23
3.3.1 Query Products 23
3.3.1.1 Query by Name 23
3.3.1.2 Query by List 23
3.3.1.3 Query by Product Publication Date 24
3.3.1.4 Query by Sensing Date 24
3.3.1.5 Query by Geographic Criteria 24
3.3.1.6 Query by Attributes 24
3.3.1.7 Additional Options 25
3.3.1.8 Query Products Response 26
3.3.2 Catalogue Export 27
3.4 Retrieve Product 28
3.4.1 Product Order 30
3.4.2 Product Download 34
3.5 Bulk Product Retrieve 35
3.5.1 Bulk and BatchOrder Model and Properties 37
3.5.2 Bulk and BatchOrder Examples 42
3.5.2.1 Step 1 – Bulk Create 42
3.5.2.2 Step 2 – BatchOrder Query and BatchOrder Triggering 43
3.5.2.3 Step 3 – Product Staging Notification and Product Retrieval 45

3.6 Subscription 46
3.7 Traceability Interface 53
3.8 User Administration 53
3.8.1 Basic LTA User Entity Model 53
3.8.1.1 SystemRole 54
3.8.2 Quota 55
3.9 Monitoring and Reporting 55
3.9.1 Reporting 56
3.9.2 LTA Metrics 60
3.10 LTA Event Recording 63
3.10.1 Event Entity Model 63
4 ANNEX 1: ODATA METADATA DESCRIPTION 65
5 ANNEX 2: COMPLIANCE TEST SUITES 70
5.1 LTA Minimum Compliance Test Suite Instructions 70
5.1.1 Product Listing Query 70
5.1.2 Product Query By Name 72
5.1.3 Product Query By Sensing Date 72
5.1.4 Product Details 73
5.1.5 Product Order 73
5.1.6 Product Download 74
5.2 LTA Extended Compliance Test Suite Instructions 75
5.2.1 Product Listing Query 75
5.2.2 Product Query By Name 76
5.2.3 Product Query By Sensing Date 77
5.2.4 Product Query By Publication Date 77
5.2.5 Product Query By Geographic Criteria 78
5.2.6 Product Details 79
5.2.7 Product Order 79
5.2.8 Product Download by ProductId 80
5.2.9 Product Download by OrderId 80
5.2.10 Order List Query 81
5.2.11 Product Attributes Expand Query 82
5.2.12 Product Query By Attributes 82

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this document is to specify the interfaces of the Long Term Archive (LTA) within the Copernicus Space Component (CSC) Ground Segment (GS) as part of the overall CSC ESA Operations Framework.

In particular this document aims to specify an https RESTful Application Programming Interface (API) of the LTA through which Copernicus Sentinel data products may be queried and downloaded by authorised users.

This specification uses the latest OData v4 standard for RESTful. The API interface is intended to be compliant with the OData v4 standard, however, it is not necessary to fully implement the OData v4 standard. Compliance to the ICD can be considered as an API contract with a set of API OData v4 style calls, which have to be supported, that shall follow OData query conventions such as URL syntax, filter syntax, content accessing conventions, etc.

1.2 Structure of the Document

This document is structured as follows:

- Section 1 (this section): Introduction, providing document structure, reference documents and definitions/acronyms
- Section 2: Provides the context of the LTA and overview of the use cases supported
- Section 3: Description of the interfaces
- Sections 4 and 5: Annexes, containing OData metadata description and compliance test suites

1.3 Applicable Documents

[AD-1]	CSC Sentinel-1 Product Unit Definition and Metadata ICD [ESA-EOPG-EOPGC-SP-01]
[AD-2]	CSC Sentinel-2 Product Unit Definition and Metadata ICD [ESA-EOPG-EOPGC-SP-02]
[AD-3]	CSC Sentinel-3 Product Unit Definition and Metadata ICD [ESA-EOPG-EOPGC-SP-03]
[AD-4]	CSC Sentinel-5P Product Unit Definition and Metadata ICD [ESA-EOPG-EOPGC-SP-04]
[AD-5]	CSC POD Product Unit Definition and Metadata ICD [ESA-EOPG-EOPGC-SP-5]
[AD-6]	CSC Common Entity Definition Document [ESA-EOPG-EOPGC-IF-5]
[AD-7]	Copernicus Data Space Ecosystem - Traceability Service Interface Control Document [CDSE-TRC-TSY]

[AD-8] OData Version 4.01. Part 1: Protocol
http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html

[AD-9] OData Version 4.01. Part 2: URL Conventions
http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part2-url-conventions.html

[AD-10] OData JSON Format Version 4.01
http://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.html

[AD-11] "The GeoJSON Format", RFC 7946, August 2016.
http://tools.ietf.org/html/rfc7946

1.4 Reference Documents

[RD-1] OData Documentation – OData – the Best Way to REST – v4.01 ISO standard
http://www.odata.org/documentation/

[RD-2] OData Protocol
http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html

[RD-3] OGC Earth Observation Metadata profile of Observations & Measurements
http://docs.opengeospatial.org/is/10-157r4/10-157r4.html

[RD-4] OGC EO Dataset Metadata GeoJSON(-LD) Encoding Standard
https://docs.ogc.org/is/17-003r2/17-003r2.html

[RD-5] Production Interface Delivery Point ICD [ESA-EOPG-EOPGC-IF-3]

[RD-6] Data Protection (GDPR)
https://ec.europa.eu/info/law/law-topic/data-protection_en

[RD-7] The OAuth 2.0 Authorization Framework – Password Credentials Grant, October 2012, Internet Engineering Task Force (IETF)
https://tools.ietf.org/html/rfc6749#section-4.3

[RD-8] Auxiliary Data Interface Delivery Point ICD [ESA-EOPG-EOPGC-IF-10]

1.5 Definitions

AIP – Archive Interface delivery Point – the interface delivery point for products requested from the LTA

API – Application Programming Interface – a set of routines, protocols, and tools for building software applications.

Use Case – is a list of actions or event steps, typically defining the interactions between an actor and a system, to achieve a goal.

Client – any HTTP client performing HTTP/OData requests to the LTA

Product – Packaged set of data files corresponding to EO or auxiliary product format specification

Order – A client request for a data product to be retrieved by the LTA and staged on the Archive Interface Delivery Point (AIP), thereby making it available for download by the requesting user.

Bulk Request – A client request to trigger the creation of a Bulk, defined by a list of products or query filter parameters and a Batch size. The LTA breaks the Bulk into Batches, based on the Batch size provided and the most efficient ordering of products.

Batch Order – A client request to trigger the delivery of products making up a particular Batch (resulting from a Bulk) from the LTA to the AIP, from where they can subsequently be downloaded.

Subscription – A client request for notification of future products entering the LTA which fulfil particular query filter parameters. A Subscription can optionally request for such products to be automatically Ordered by the LTA, thus staged on the AIP and made available for download.

Priority – The LTA manages a priority scheme for all retrievals. Priority is defined as an integer e.g. 1-100 (with 100 being the highest priority) and product retrievals from the archive are managed in the order of priority. A user is assigned a default priority and a maximum priority.

Quota – Refers to a restriction in the usage of the LTA - e.g. the number of orders being processed in parallel (i.e. in status *in_progress*) which a user may have ongoing at a particular time.

1.6 List of Acronyms

AD	– Applicable Document
AIP	– Archive Interface delivery Point
API	– Application Programming Interface
CSC	– Copernicus Space Component
E2E	– End-to-End
EO	– Earth Observation
ESA	– European Space Agency
FIFO	– First In, First Out
GS	– Ground Segment
HTTP(S)	– Hypertext Transfer Protocol (Secure)
ID	– Identifier
ICD	– Interface Control Document
JSON	– JavaScript Object Notation
LTA	– Long Term Archive
OData	– Open Data Protocol
POD	– Precise Orbit Determination
PRIP	– PRoduction Interface delivery Point
RD	– Reference Document

URI	– Uniform Resource Identifier
UTC	– Coordinated Universal Time
UUID	– Universally Unique Identifier

2 OVERVIEW

2.1 Context

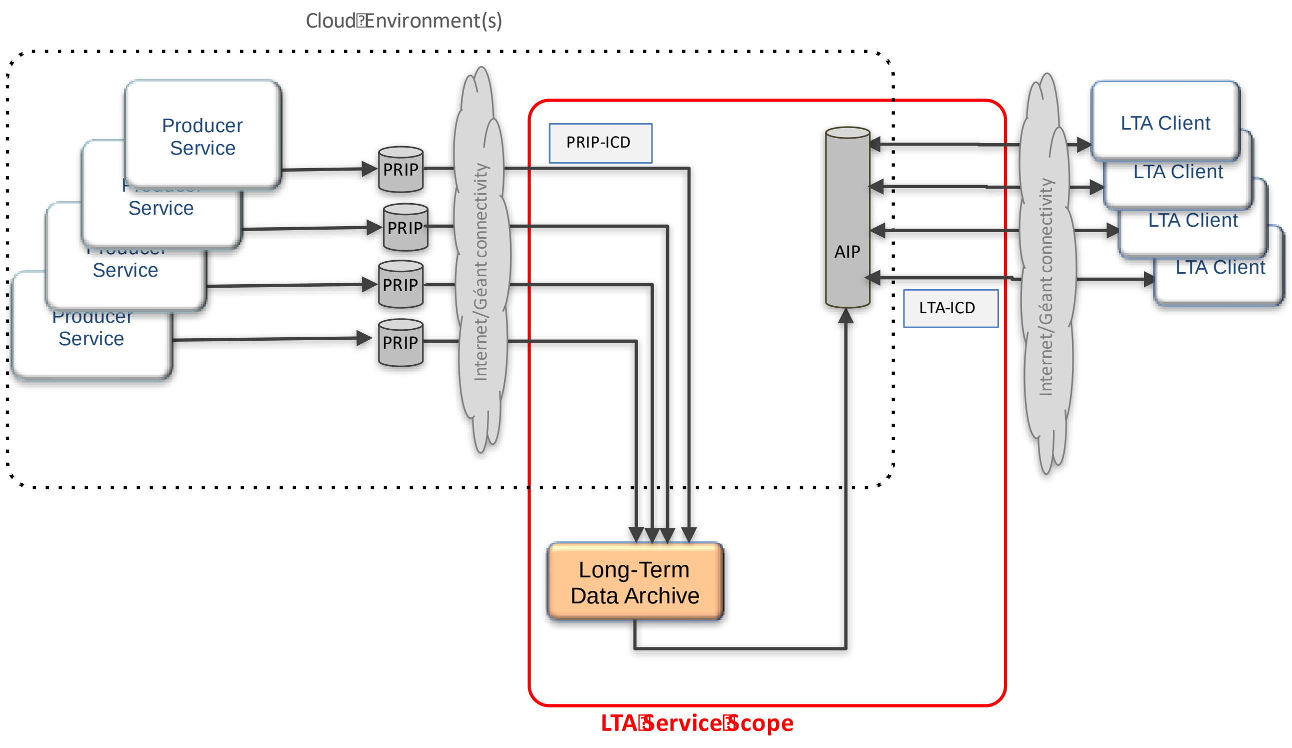


Figure 1: Long Term Archive (LTA) Overall Context

Multiple LTAs are necessary in the Copernicus GS architecture to ensure redundant geographically separated safe storage for needs of all Sentinel missions. The same LTA may serve one single Sentinel mission or several Sentinel missions. Each LTA operates autonomously with respect to the other LTAs.

As a consumer of data the LTA interfaces with:

- the Systematic Processing and Routine Quality Control elements, through the Production Interface delivery Point (PRIP) to retrieve the Sentinel products files to be stored in the LTA
- the Auxiliary Data Gathering element, through the Auxiliary Data Gathering Interface delivery Point (AUXIP) to retrieve the auxiliary files to be stored in the LTA
- the Precise Orbit Determination element, through a POD Auxiliary Interface delivery Point (POD AUXIP), to retrieve the POD information to be stored in the LTA

The interfaces to these producer services are governed by the Production Interface delivery Point ICD [RD-5] and the Auxiliary Data Interface Delivery Point [RD-8].

As a provider of data the LTA interfaces with multiple clients:

- the Data Access and On-Demand Production elements, through the Archive Interface delivery Point

- the E2E Operations Performance Monitoring element, to provide the reporting information required for the E2E operations monitoring
- potentially the other LTA instances, and other clients such as Cal/Val users requiring data not otherwise available through the Data Access

The interface to these clients for the access to the API and delivery of Products is the so called Archive Interface delivery Point, governed by this ICD.

2.2 Nominal Use Scenario Description

The nominal use scenarios for the LTA are shown in the figure below and introduced in the following sections.

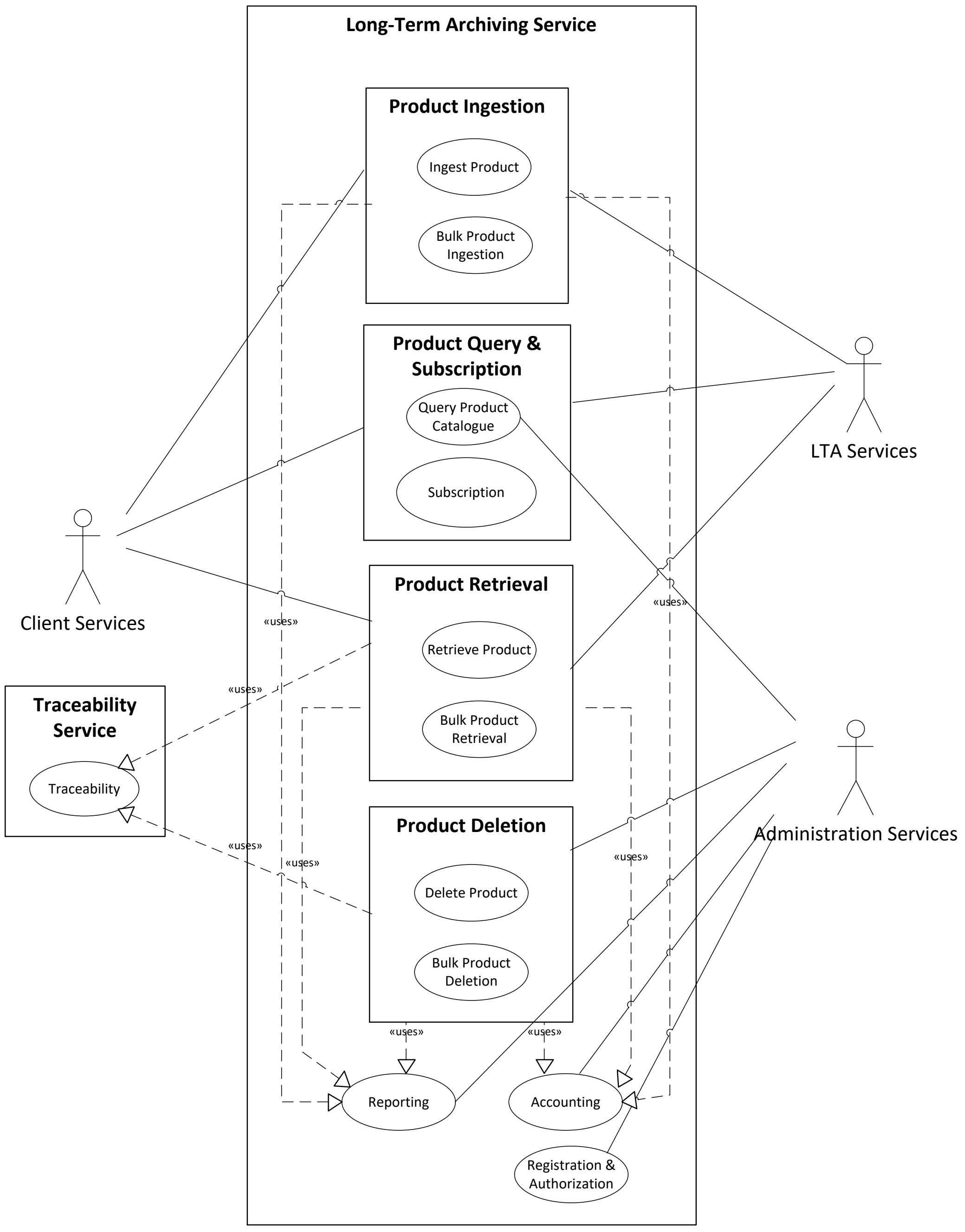


Figure 2: Long-Term Data Archiving Services System Use-Case Diagram

2.3 Ingest Product

The LTA service builds its archive and product catalogue from the ingested products. The LTA routinely downloads the latest products from the Production Interface Delivery Point (PRIP) and Auxiliary data Interface delivery Points (AUXIPs – describing the delivery points of the Auxiliary Data Gathering and POD Services), which follow the specifications described in [RD-5] and [RD-8], respectively. The metadata required for the product catalogue defined in the relevant Product Unit Definition and Metadata ICD [AD-1] to [AD-5].

2.4 Bulk Product Ingestion

The LTA service may take advantage of bulk retrieval interfaces of another reference LTA for the ingestion of large sets of data. The bulk product ingestion may use a catalogue export from the reference LTA to avoid un-necessarily parsing the individual products for the metadata. The catalogue export function is described in section 3.3.2. The bulk product ingestion may use the bulk product retrieval interface of the reference LTA to retrieve the data in the most efficient manner for the reference LTA (e.g. one that optimises tape access or caching strategies). The bulk retrieval interface is described in section 3.5.

2.5 Query Product Catalogue

The query catalogue function is provided through the API described in section 3.3. The API is used to identify required products that may be retrieved from the LTA and, via their identifiers, to trigger the product retrieval. The product catalogue can be queried for specific named products or lists of products, or products matching more complex query criteria. The LTA product catalogue provides an export function to provide the metadata in a bulk format.

2.6 Retrieve Product

The retrieve product function is also provided through the API as described in section 3.4. The retrieve product is implemented via an order triggering the delivery of the required product on the Archive Interface delivery Point (AIP) following which the product may be accessed for a limited amount of time and downloaded from the AIP.

2.7 Bulk Product Retrieve

The bulk product retrieve function is provided through the API as described in section 3.5. The bulk product retrieve provides a filter query to define the list of products to be staged, in general it leaves the LTA to define the most efficient order of data extraction by defining batches of data.

2.8 Delete Product

The product delete function is not provided directly to clients via API. Deleted products are removed from the Product Catalogue triggered by a local procedure, their history and the rationale for deletion is provided through the external product traceability service.

2.9 Bulk Delete Product

The bulk product delete function is not provided through a client API. Deleted products are removed in the Product Catalogue triggered by a local procedure, their history and the rationale for deletion is provided through the external product traceability service.

2.10 Subscription

The subscribe function provides a call-back mechanism by which a client may be informed of any new or deleted product in the LTA matching certain query criteria and optionally the requiring the delivery of the product to the AIP. The API is described in section 3.6

2.11 Reporting

The reporting function enables the generation of LTA service performance reports, and EO data population reports, and may be used for end-to-end monitoring purposes.

2.12 Registration and Authorisation

The registration and authorisation function provides the possibility to register user clients which need to interact with the LTA service. For each registered client, the Service Manager is able to define role-based access rights corresponding to the types of ground segment actors (see section 3.7). Each individual client is assigned to one role, defining the corresponding LTA functions permitted to be executed for the client.

2.13 Accounting

The accounting function provides the possibility to define limits (quota) for the use of the LTA service per client and LTA function to prevent IT resource overflows and to define different priorities for concurrent clients. The following limitations shall be supported:

- Retrieve Product, Bulk Product Retrieval, Product Download: number of concurrent orders/downloads and total per set duration

If a client reaches one of the limits when calling an LTA service function, it is notified with a corresponding return message including information about the current function use, the limitation imposed.

2.14 Traceability

The LTA services interfaces with the CSC Traceability and Certification service to verify and provide traces for data origin and integrity.

3 LONG TERM ARCHIVE CLIENT API DESCRIPTION

The following sections describe the client API of the LTA. The LTA may support further internal interfaces through APIs to support the local LTA service operations, but only the standard interfaces described in this ICD shall be offered towards any client.

3.1 Interface Configuration

Only entitled clients are allowed to request products from the LTA. As part of every request to the API of the LTA, the client triggering the requests will identify themselves via *Explicit identification.*

The API shall support at least the following two methods of authentication:

- HTTP Basic Authentication
- OAuth 2.0

HTTP Basic Authentication represents a simple and easy method, consisting of a client placing the username and password inside the authorization header of the request. The username and password are encoded with base64, an encoding technique that converts the username and password into a set of 64 characters in order to ensure safe transmission. It is mandatory to use Basic Authentication in conjunction with HTTPS in order to provide sufficient security to prevent unauthorized users from discovering the authentication information.

Although HTTP Basic Authentication should be supported for the purpose of testing and initial phases, it shall not represent the main authentication method used. Operationally, OAuth 2.0 is to be used (https://tools.ietf.org/html/rfc6749), as it is a more secure and powerful system, allowing for scalability of security. OAuth allows access tokens and refresh tokens to be issued to clients by an authorization server. The client then uses the access token to access the interface delivery point API. The OAuth flow / grant type, to be used is the Password Credentials grant: https://tools.ietf.org/html/rfc6749#section-4.3.

The definition of new users on the LTA side is done by configuration creating them with the necessary attributes. The attributes to be configured per client may be:

- **User Credentials** (username, password) for authentication. Usernames and any associated attributes e.g. email are to be managed as generic indicators of client and not personal information (see section 3.8).
- **User Roles –** For assigning the functionality available to the client (see section 3.8.1.1)
- **User Quota** – LTA Requests may be limited by a quota mechanism (see section 3.8.2).
- **Order Priority -** Product retrievals from the archive are managed in the order of priority, with priority defined as an integer e.g. 1-100 (with 100 being the highest priority). Although priority may be assigned on a per order basis by the user, its assignation is limited by two parameters set as part of user configuration:
    - *Default Priority* – The priority assigned if the user does not set a priority with an order (e.g. 50)
    - *Maximum Priority* – The maximum priority with a user may assign to an order (e.g. 100)

It should be noted that some of these attributes (e.g. User Credentials) may be managed externally of the OData LTA implementation, e.g. as part of an OAuth 2.0 system. When a new user request arrives from ESA, the LTA Service shall manage this as part of the service initialisation. The password together with any other credentials required for authentication will be communicated by the LTA service provider directly to the user, by using the user e-mail.

3.2 Basic LTA Product Entity Model

The figure below shows the LTA basic Product Entity model.

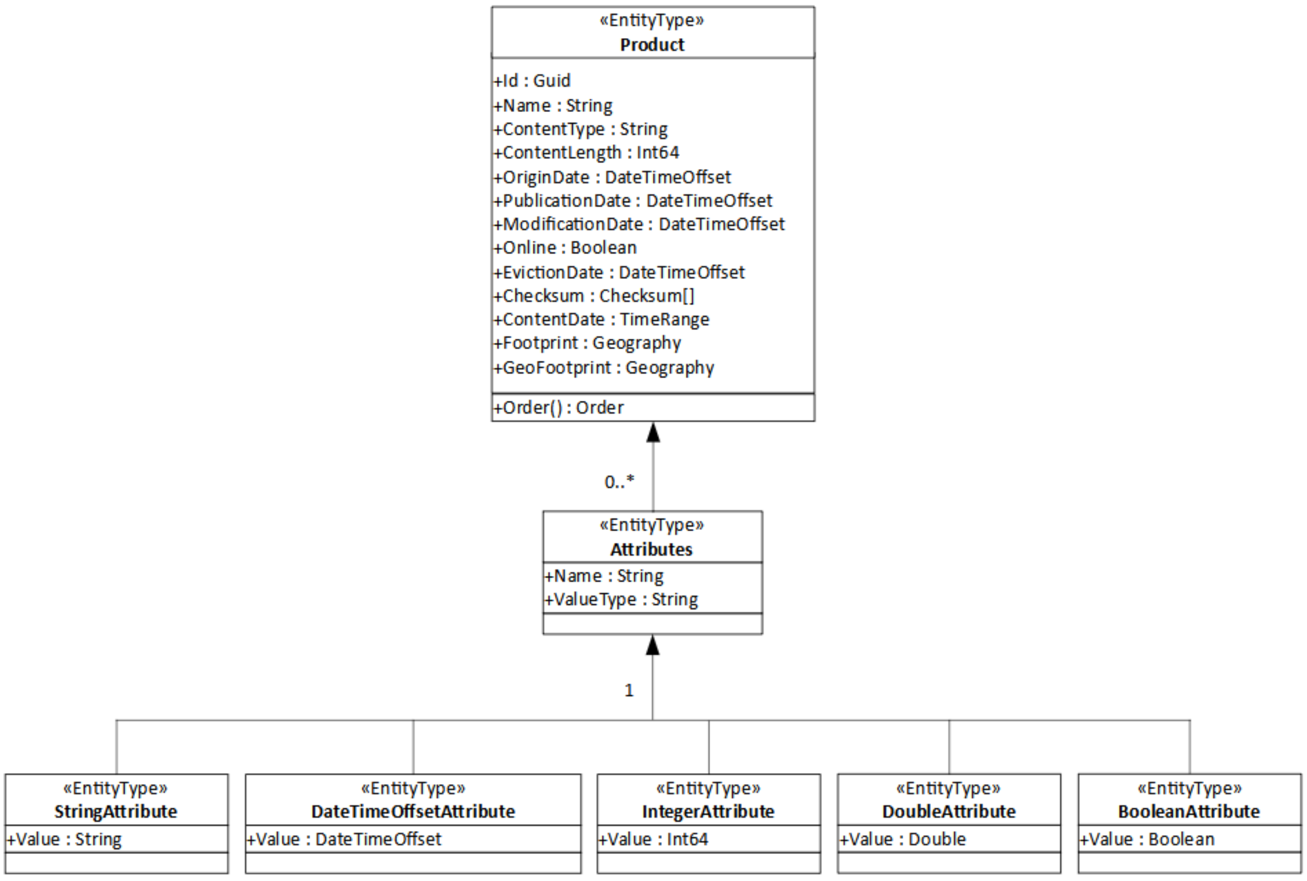


Figure 3: Product Entity Model

Products Properties	Type	M	Description	Example
Id	Guid[1]	Y	It is a universally unique identifier (UUID). The Id is a local identifier for the product instance within the LTA,	`2b17b57d-fff4-4645-b539-91f305c27c69`

[1] For this, and for all other instances of the use of 'Guid' for IDs in this document, it should be noted that the IDs are UUIDs (Universally Unique Identifiers) of the (OData) Guid type.

			assigned during the product ingestion.	
Name	String	Y	Data file name	DD S1A_IW_SLC__1SDV_ 20160117T103451_ 20160117T103518_ 009533_ 0094_D46A.SAFE.zip
ContentType	String	Y	The Mime type of the product	`application/octet-stream`
ContentLength	Int64	Y	Actual size in bytes (B) of the downloadable product package	`4737286945`
OriginDate	DateTimeOffset	Y	Date and time of the product at the source (e.g. Publication date time on the PRIP). Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T12:56:05.232Z`
PublicationDate	DateTimeOffset	Y	Publication date and time of the product (time at which the product becomes accessible for retrieval to the client within the LTA). Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T14:46:03.788Z`
ModificationDate	DateTimeOffset	Y	Date when the product metadata was last modified. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-19T18:00:00.000Z`
Online	Boolean	Y	Indication of the presence of the product in the AIP, available for immediate download. This property can change in time and should be used taking into account EvictionDate.	`true`
EvictionDate	DateTimeOffset	N	Date when the data file will be removed from the Archive delivery Interface Point. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ Only provided if Online = true	`2018-01-22T18:00:00.000Z`

Checksum (Algorithm , Value and ChecksumDate)	Checksum[]	Y	Represents the known checksums for the product's physical data, providing a unique value for supporting download integrity check. At least MD5 checksum is mandatory. Each checksum also includes the ChecksumDate when it was calculated, in UTC in the	"Checksum":
				[
				{
				"Algorithm":"MD5",
				"Value":"E8A303BF3D85200514F727DB60E7DB65"
				"ChecksumDate":"2018-01-22T18:00:00.000Z",
				}
				]

			format YYYY-MM-DDThh:mm:ss.sssZ.	
ContentDate	TimeRange	Y	The sensing range period. Start and end times are in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`"ContentDate": { "Start":"2016-01-17T10:34:51.922Z", "End":"2016-01-17T10:35:18.872Z" }`
Footprint[2]	Geography	N	Footprint of the product. "Polygon" is used only as an example here, as it represents the most common type of footprint. However, this will correspond to the type of footprint of the product, which can also be e.g. LineString, MultiLineString, etc.	`geography'SRID=4326;Polygon((),(-41.15749 66.766701,-31.740927 67.629135,-31.479883 66.860405,-40.616844 66.011871,-41.15749 66.766701))'`
GeoFootprint[3]	Geography	N	Mandatory for georeferenced products, following the definition in [RFC7946], with the following modification, according to [OData JSON Format Version 4.01 – Section 7.1]: - Keys should be ordered with type first, then coordinates, then any other keys "Polygon" is used only as an example here, as it represents the most common type of GeoFootprint. However, the type will correspond to the type of footprint of the product, which can also be e.g. LineString, MultiLineString, etc.	`"GeoFootprint":{ "type": "Polygon", "coordinates": [ [ [-59.3169, 2.6367], [-63.105, -14.0539], [-60.8506, -14.4245], [-57.1309, 2.3269], [-59.3169, 2.6367] ] ] }`

Table 1: Product Entity Description

As Attribute values will require different data types, they are based on derived entity types such as StringAttribute, DateTimeOffsetAttribute, IntegerAttribute etc (see figure 3).

For the full list of applicable attributes, refer to [AD-1], [AD-2], [AD-3], [AD-4], [AD-5].

[2] Will be deprecated in favour of GeoFootprint starting from TBC date

[3] To be implemented starting from January 2023

Attribute Properties	Type	M	Description	Example
Name	String	Y	String name of the attribute	`platformShortName`
ValueType	String	Y	The type of attribute. This shall correspond to the following[4]: - String - Integer - DateTimeOffset - Boolean - Double	`String`
Value	Depends on ValueType		The value of the attribute, depending on the ValueType	`SENTINEL-2`

Table 2: Attribute Entity Description

The response to an *`$expand=Attributes`* query shall include at least the attribute properties listed in Table 2.

Example of $expand=Attributes Query:

```
https://<service-root-uri>/odata/v1/Products?$filter=contains(Name,'IW_RAW__0N') and
ContentDate/Start gt 2022-06-26T00:00:00.000Z and ContentDate/End lt 2022-06-
26T10:00:00.000Z&$top=1$$expand=Attributes
```

Example of $expand=Attributes Query Response:

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Products(Attributes())",
  "value": [
        {
            "Id": "06c91249-b275-4811-9a55-a8d11b534945",
            "Name":
"S1A_IW_RAW__0NSH_20220626T050533_20220626T051038_043829_053B7F_203C.SAFE.zip",
            "EvictionDate": "9999-12-31T23:59:59.999Z",
            "ModificationDate": "2022-06-26T06:30:34.558Z",
            "OriginDate": "2022-06-26T06:14:55.468Z",
            "PublicationDate": "2022-06-26T06:30:34.558Z",
            "ContentLength": 47649797,
            "Online": true,
            "ContentType": "application/zip",
            "ContentDate": {
                "Start": "2022-06-26T05:05:33.863Z",
                "End": "2022-06-26T05:10:38.849Z"
            },
            "Checksum": [
                {
                    "Algorithm": "MD5",
                    "Value": "235f8cac56705aee78b4c8bbb45c7604",
                    "ChecksumDate": "2022-06-26T06:14:55.468Z"
                }
            ],
            "Footprint": "geography'SRID=4326;POLYGON((-59.3169 2.6367,-63.105 -14.0539,-
60.8506 -14.4245,-57.1309 2.3269,-59.3169 2.6367))",
```

[4] Only these values shall be allowed (e.g. Edm.DateTimeOffset or DateTimeOffsetAttribute is not allowed)

```
            "GeoFootprint":{
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            -59.3169,
                            2.6367
                        ],
                        [
                            -63.105,
                            -14.0539
                        ],
                        [
                            -60.8506,
                            -14.4245
                        ],
                        [
                            -57.1309,
                            2.3269
                        ],
                        [
                            -59.3169,
                            2.6367
                        ]
                    ]
                ],
            },
            "Attributes": [
                {
                    "Name": "polarisationChannels",
                    "ValueType": "String",
                    "Value": "HH"
                },
                {
                    "Name": "productType",
                    "ValueType": "String",
                    "Value": "IW_RAW__0N"
                },
                {
                    "Name": "cycleNumber",
                    "ValueType": "Integer",
                    "Value": 265
                },
                {
                    "Name": "relativeOrbitNumber",
                    "ValueType": "Integer",
                    "Value": 43829
                },
                {
                    "Name": "processingCenter",
                    "ValueType": "String",
                    "Value": "S1 Production Service-SERCO"
                },
                {
                    "Name": "completionTimeFromAscendingNode",
                    "ValueType": "Double",
                    "Value": 4646455.264
                },
                {
                    "Name": "processingDate",
                    "ValueType": "DateTimeOffset",
                    "Value": "2022-06-26T06:11:22.535Z"
                },
                {
                    "Name": "datatakeID",
                    "ValueType": "Integer",
                    "Value": 342911
                },
                {
                    "Name": "orbitNumber",
                    "ValueType": "Integer",
                    "Value": 43829
                },
                {
                    "Name": "productConsolidation",
                    "ValueType": "String",
                    "Value": "FULL"
                },
                {
                    "Name": "endingDateTime",
                    "ValueType": "DateTimeOffset",
                    "Value": "2022-06-26T05:10:38.849Z"
                },
                {
                    "Name": "qualityDataObjectID",
                    "ValueType": "String",
                    "Value": "measurementData"
                },
                {
                    "Name": "coordinates",
                    "ValueType": "String",
                    "Value": "-59.3169 2.6367,-63.105 -14.0539,-60.8506 -14.4245,-57.1309
2.3269,-59.3169 2.6367'"
                },
                {
                    "Name": "orbitDirection",
                    "ValueType": "String",
                    "Value": "ASCENDING"
                },
                {
                    "Name": "sliceProductFlag",
                    "ValueType": "Boolean",
                    "Value": false
                },
                {
                    "Name": "beginningDateTime",
                    "ValueType": "DateTimeOffset",
                    "Value": "2022-06-26T05:05:33.863Z"
                },
                {
                    "Name": "instrumentConfigurationID",
                    "ValueType": "Integer",
                    "Value": 7
                },
                {
                    "Name": "startTimeFromAscendingNode",
                    "ValueType": "Double",
                    "Value": 4341469.328
                },
                {
                    "Name": "instrumentShortName",
                    "ValueType": "String",
                    "Value": "SAR"
                },
                {
                    "Name": "productClass",
                    "ValueType": "String",
                    "Value": "N"
                },
                {
                    "Name": "platformShortName",
                    "ValueType": "String",
                    "Value": "SENTINEL-1"
                },
                {
                    "Name": "platformSerialIdentifier",
                    "ValueType": "String",
                    "Value": "A"
                }
            ]
        }
    ]
}
```

3.3 Query Products Catalogue

The Query Products Catalogue function is achieved through standard APIs according to the Entity Model described in section 3.2.

3.3.1 Query Products

Some of the main query possibilities are outlined below, filter conditions may be combined.

3.3.1.1 Query by Name

The most basic way to query products from within the LTA is to perform queries based on the filename, using string functions.

Function	Description
contains	The contains function returns records with names containing a particular string at any position
endswith	The endswith function returns true if the first parameter string value ends with the second parameter string value, otherwise it returns false
startswith	The startswith function returns true if the first parameter string value starts with the second parameter string value, otherwise it returns false

The list of products starting with the filename “S1B_EW_”can be retrieved as follows:

```
https://<service-root-uri>/odata/v1/Products?$filter=startswith(Name,'S1B_EW_')
```

The following query will return all products containing “S3B_ SL_1_RBT__” :

```
https://<service-root-uri>/odata/v1/Products?$filter=contains(Name,'S3B_ SL_1_RBT')
```

3.3.1.2 Query by List

An alternative to using Query by Name for a multiple product query, in cases where the list of queried products is longer than what can be contained in the URI, the Query by List is to be used. Such a query would use the POST method to execute a ‘query by list’ action on a list of products contained in the body. For example:

```
POST http://<service-root-uri>/odata/v1/Products/OData.CSC.FilterList
{
   "FilterProducts":
   [
     {"Name": "product1"},
     {"Name": "product2"},
     {"Name": "product3"}
     etc.
   ]
}
```

3.3.1.3 Query by Product Publication Date

The list of products published in the AIP between two dates can be obtained as follows:

```
https://<service-root-uri>/odata/v1/Products?$filter=PublicationDate gt 2017-05-
15T00:00:00.000Z and PublicationDate lt 2017-05-16T00:00:00.000Z
```

3.3.1.4 Query by Sensing Date

The list of products filtered by sensing date criteria can be retrieved for example as follows:

```
https://<service-root-uri>/odata/v1/Products?$filter=ContentDate/Start gt 2019-05-
15T00:00:00.000Z and ContentDate/End lt 2019-05-16T00:00:00.000Z
```

3.3.1.5 Query by Geographic Criteria

The list of products filtered by geographical criteria can be retrieved as follows:

```
https://<service-root-
uri>/odata/v1/Products?$filter=OData.CSC.Intersects(area=geography'SRID=4326;POLYGON((-
127.89734578345 45.234534534,-127.89734578345 45.234534534,-127.89734578345
45.234534534,-127.89734578345 45.234534534))')
```

It must be noted that this is a non-native OData function. It is introduced as a custom function extension: http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html#sec_ActionFunctionExtensibility; with the following signature:
Edm.Boolean OData.CSC.Intersects(Edm.GeographyPolygon,Edm.GeographyPolygon)

3.3.1.6 Query by Attributes

In order that the LTA provides a precise metadata model it is important that each LTA use precisely the same metadata element from the product. [AD-1], [AD-2], [AD-3], [AD-4], [AD-5] provide the definition of the minimum metadata that shall be catalogued for each product, and their origin within the product.

The list of products associated with a particular attribute Name value can be obtained as follows:

```
https://<service-root-
uri>/odata/v1/Products?$filter=Attributes/OData.CSC.ValueTypeAttribute/any(att:att/Name
eq '[Attribute.Name]' and att/OData.CSC.ValueTypeAttribute/Value eq '[Attribute.Value]')
```

Products can also be queried by using a combination of attributes as in the example below:

```
https://<service-root-uri>/odata/v1/Products?$filter=Attributes/OData.CSC.StringAttribute/any(att:att/Name eq 'productType' and att/OData.CSC.StringAttribute/Value eq 'MSI_L0__GR') and Attributes/OData.CSC.StringAttribute/any(att:att/Name eq 'processorVersion' and att/OData.CSC.StringAttribute/Value eq '02.13')
```

3.3.1.7 Additional Options

Query functions shall be supported on all properties of the Product according to the OData $filter query option. Results ordering shall be supported according to the $orderby option; result paging shall be supported according to the $top, $skip and $count options, with the page size configurable to support at least 1000 returned results per page. More details on these options are provided below.

$orderby

The $orderby system query option allows clients to request resources in either ascending order using asc or descending order using desc. If asc or desc not specified, then the resources will be ordered in ascending order. For example:

```
        https://<service-root-uri>/Products?$orderby=PublicationDate desc
```

Will return a list of products ordered by PublicationDate in descending order

$top

The $top system query option allows clients to specify the maximum (non-negative integer) number of items returned from a query. In the case of the LTA AIP, $top shall accept a maximum value of at least 1000.

For example:

```
        https://<service-root-uri>/Products$top=1000&$filter=startswith(Name,'S2')
```

Will return the first 1000 Sentinel-2 products currently available on the AIP.

$skip

The $skip system query option allows clients to specify a (non-negative integer) number of items excluded from the start of a query result, i.e. the query will start returning items from the provided integer+1.

For example:

```
        https://<service-root-uri>/Products$skip=100&$filter=startswith(Name,'S2')
```

Will return the all Sentinel-2 products currently available on the LTA AIP, excluding the first 100.

$top and $skip are often applied together; in this case $skip is always applied first regardless of the order in which they appear in the query.

$count

The $count system query option allows clients to request a count of the matching resources included with the resources in the response: the number of the matching resources is returned as result.

The $count query option is useful to know the number of entities which are identified by the resource path section of the URI after having applied some filters.

For example:

```
       https://<service-root-uri>/Products$count=true&$filter=startswith(Name,'S2')
```

Will return the number of Sentinel-2 products currently available on the AIP

3.3.1.8 Query Products Response

For each product element in the Products List, the full set of properties detailed in Table 1 are provided:

Example (JSON format):

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Products/$entity",
  "value":
  [
  {
    "@odata.mediaContentType": "application/octet-stream",
    "Id": "e872683a-ea5b-455d-bfb2-fb304cbfbacb",
    "Name":
"S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6.SAFE.zip",
    "ContentType": "application/octet-stream",
    "ContentLength": 4063869137,
    "OriginDate": "2016-12-25T11:43:00.000Z",
    "PublicationDate": "2016-12-25T13:18:00.048Z",
    "ModificationDate": "2016-12-26T13:02:30.096Z",
    "Online": true,
    "EvictionDate": "2016-12-28T13:18:00.048Z",
    "Checksum":
    [
    {
      "Algorithm": "MD5",
      "Value": "E8A303BF3D85200514F727DB60E7DB65",
      "ChecksumDate": "2016-01-20T10:34:51.922Z",
    }
    ],
    "ContentDate":
    {
      "Start": "2016-01-17T10:34:51.922Z",
      "End": "2016-01-17T10:35:18.872Z"
    }
    "Footprint": "geography'SRID=4326;Polygon((-41.15749 66.766701,-31.740927 67.629135,-
31.479883 66.860405,-40.616844 66.011871,-41.15749 66.766701))'",
    "GeoFootprint":{
      "type": "Polygon",
      "coordinates": [
              [
                     [-41.15749, 66.766701],
                     [-31.740927, 67.629135],
                     [-31.479883, 66.860405],
                     [-40.616844, 66.011871],
                     [-41.15749, 66.766701]
              ]
       ]
      }
    }
  ]
}
```

3.3.2 Catalogue Export

Catalogue data may be exported from the LTA in a compact format for cross-reference with other GS elements. The elements included in the catalogue export are the product properties and attribute properties, and shall be presented in JSON format corresponding to the query:

```
Products?$expand=Attributes&$format=json
```

Recognising that this may be a resource consuming query it may be preferred to cache such results in sorted lists of JSON files as routinely updated views e.g. per sensing date or per creation date, whereby exported files are organized in folders according to the following hierarchical structure:

- [SSS]: Satellite Platform short name, 3 characters (e.g. S1A, S2A, …)
- [YYYY]: Sensing Year, 4 digits (e.g. 2014, 2015, …)
- [MM]: Sensing Month, 2 digits (e.g. 01, 02, …)

The month folder contains the set of exported files. There is a data file for each day of the month.
Each data file lists the products having the same 'sensing start' date.

The exported file naming convention and format is:

<SSS>_<YYYYMMDD>_<LTA>_<PROV>_catalogue_<yyyymmddhhmmss>.json

where:

- <SSS> 3 characters corresponding to the satellite platform short name S1A, S1B, S2A, etc.
- <YYYYMMDD> 8 digits corresponding to the sensing start date of all the products listed in the file
- <LTA>_<PROV> 8 characters corresponding to the LTA system (e.g. LTA_WXYZ, where WXYZ represents the LTA instance of the LTA provider)
- <yyyymmddhhmmss> 14 digits corresponding to the year, month, day and UTC time of the catalogue snapshot. The products listed in the exported file are those available in the LTA at this date.

3.4 Retrieve Product

The Retrieve Product function allows to trigger the delivery of products from the LTA to the AIP, from where they can subsequently be downloaded. This is illustrated in the sequence diagram below.

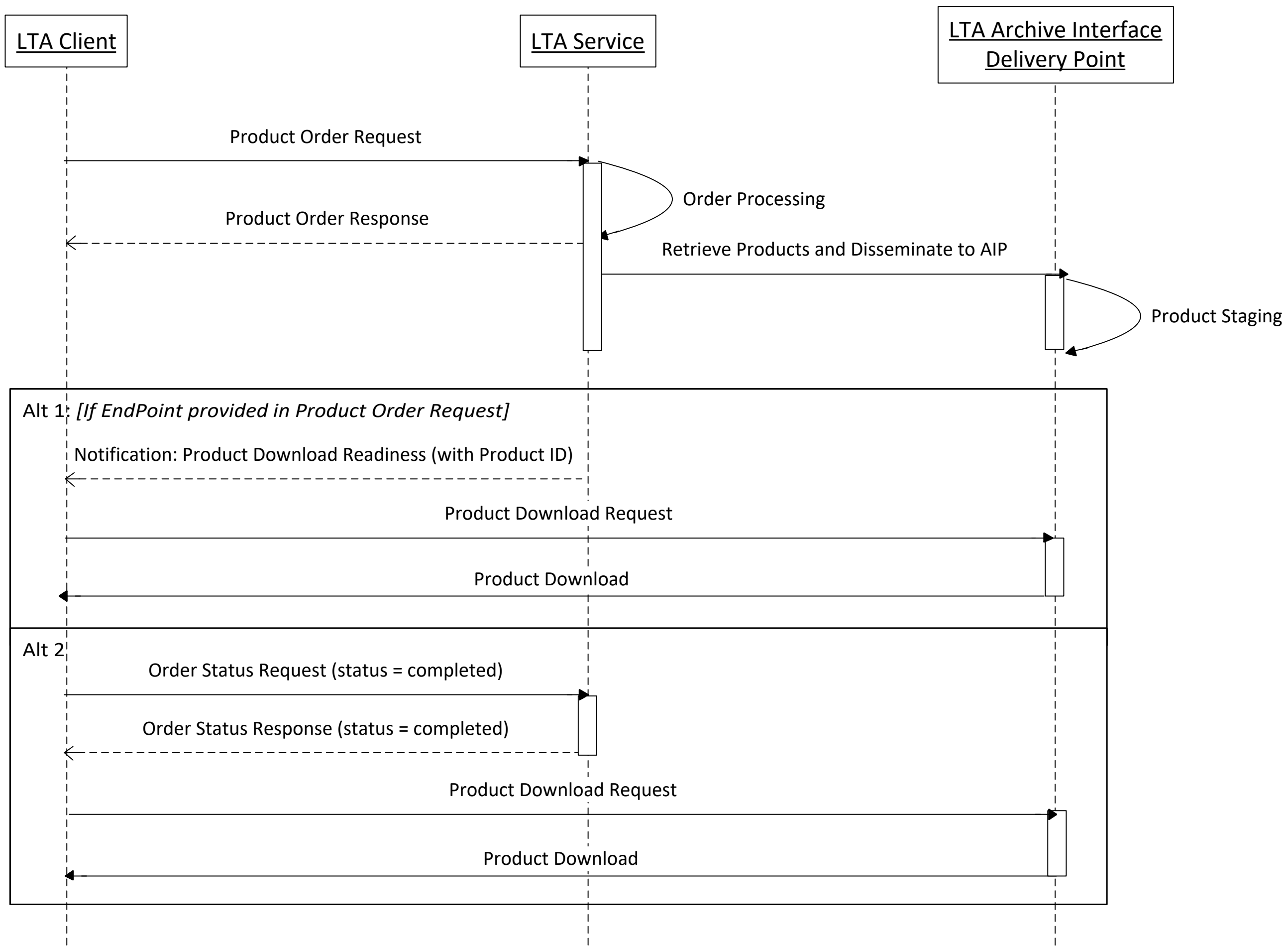

LTA Client
LTA Service
LTA Archive Interface Delivery Point
Product Order Request
Order Processing
Product Order Response
Retrieve Products and Disseminate to AIP
Product Staging
Alt 1: [If EndPoint provided in Product Order Request]
Notification: Product Download Readiness (with Product ID)
Product Download Request
Product Download
Alt 2
Order Status Request (status = completed)
Order Status Response (status = completed)
Product Download Request
Product Download

Figure 4: Product Retrieval Nominal Sequence

The scenario can be described in the following steps:

- **STEP 1 – Product Order**
  The scenario is initiated by an LTA Client submitting an Order to the LTA service. The LTA Service processes the Order, taking into account the user's quota and priority applied, and returns a response to the Client containing the unique ID of the Order, the status (at this stage '*in_progress*'), the submission date/time of the Order, the estimated date/time at which the product will be available for download, and the priority of the Order. In the nominal scenario, the Order is accepted and the LTA Service retrieves the product and disseminates to the AIP, an activity which will be queued according to the priority or the order and other orders received (FIFO).

- **STEP 2 – Product Download**

  - *ALTERNATIVE 1 – Product Staging Notification*
    Following the delivery of the product to the AIP the Order status is updated to '*completed*', and the LTA Service sends the Client a Product Download Readiness Notification containing the Product ID of the staged product (and an Eviction Date). The Client is then free to submit a Product Download Request to trigger the download the product, using the Order ID or the Product ID (see section 3.4.2). This option is applicable if an Endpoint URI has been provided as part of the Order Request.

  - *ALTERNATIVE 2 – Order Status Query*
    At user-defined intervals, the Client requests the status of the Order to the LTA Service. Once the LTA delivers the product to the delivery point the status is updated to '*completed*', the eviction time is set and the product is available for download from the AIP. The Client may request the product download using the ID of the Order, or ID of the Product (see section 3.4.2).

Regarding products that are already immediately available for download from the LTA AIP (i.e. product property "Online" = "true") the product retrieval sequence and ordering does not need to be followed. Instead, products can be directly downloaded, given that the ProductId (UUID) is known, following the query described in Section 3.4.2.

3.4.1 *Product Order*

The order action of the identified product triggers the staging of the data and dissemination to the AIP.

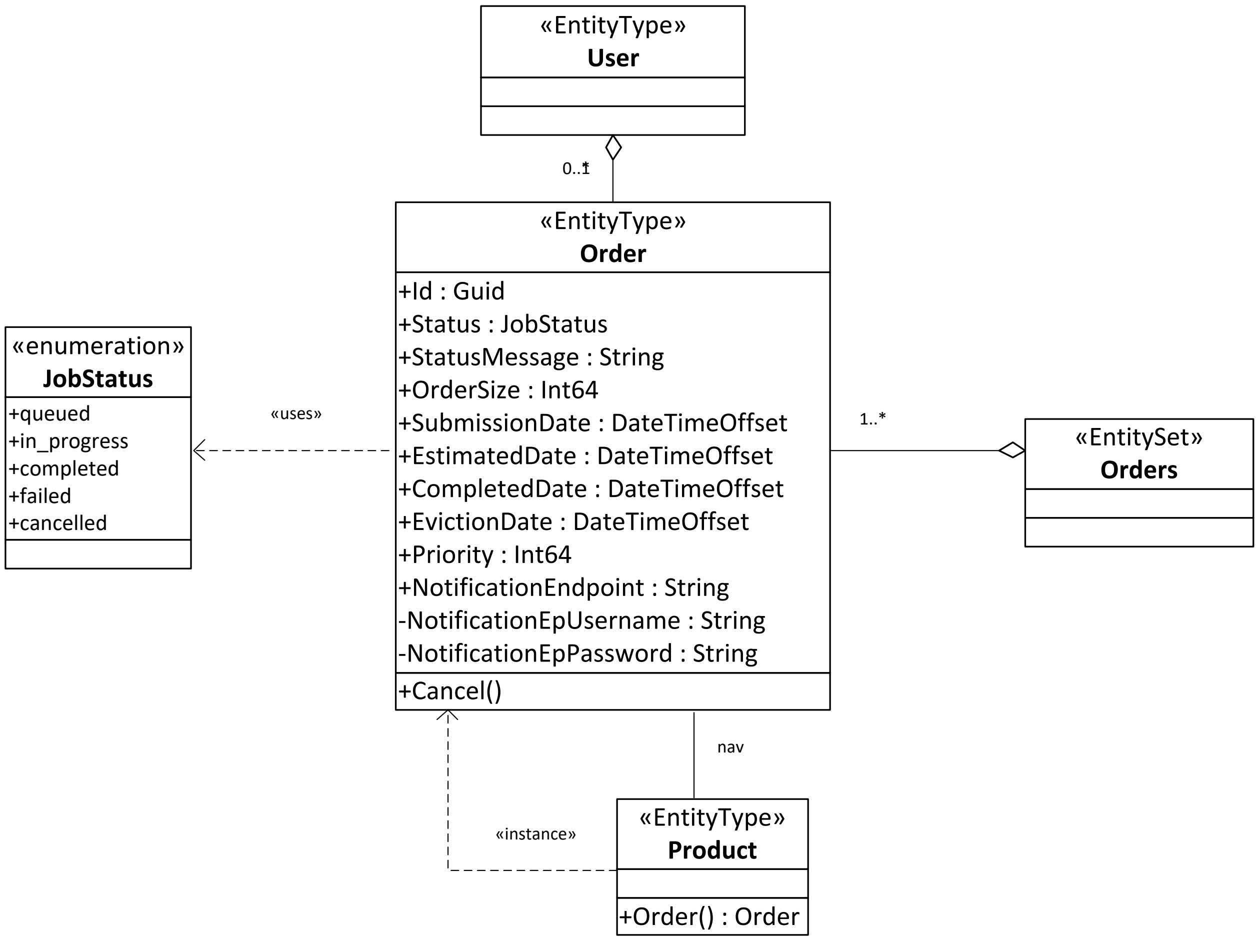


Figure 5: Order Model

Order Properties	Type	M	Description	Example
Id	Guid	Y	It is a universally unique identifier (UUID). The Id is a local identifier for the Order instance within the LTA, assigned upon Order creation.	`2b17b57d-fff4-4645-b539-91f305c27c69`
Status	JobStatus enumeration	Y	JobStatus value: - *queued* - *in_progress* - *completed* - *failed* - *cancelled* Where:	`in_progress`

			- *queued* means that the order has been accepted by the LTA Service, but it has not yet been triggered[5] - *in_progress* means that the ordered product is being retrieved and disseminated to the AIP - *completed* means that that the ordered product was successfully disseminated to the AIP and is available for download - *failed* means the retrieval and dissemination of the ordered product has failed - *cancelled* means that the order has been cancelled by the user (while in status *in_progress*)	
StatusMessage	String	Y	Text message providing additional information on the returned status, e.g. the reason for a failure. Example values are: *If status = 'queued':* 'request is queued' *If status = 'in_progress':* 'request is under processing' *If status = 'completed':* 'requested product is available' *If status = 'failed':* 'product retrieval has failed' 'product currently unavailable' 'product not found on LTA' *If status = cancelled:* 'request cancelled by user'	`requested product is available`
OrderSize	Int64	N	Actual size in bytes (B) of the data composing the Order (which would be the ProductSize unless any transformation is performed by the LTA, e.g. band extraction) Only provided if the order is in status *completed*	`4737286945`
SubmissionDate	DateTimeOffset	Y	Date and time at which the order was received by the LTA. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T14:46:03.788Z`
EstimatedDate	DateTimeOffset	Y	Estimated date and time when the product will be available for	`2018-01-22T18:00:00.000Z`

[5] The LTA service may consider a maximum queue size after which HTTP status code 429 Too Many Requests (max queue is exceeded) can be returned. The queue size should be considered as a buffer to manage e.g. a few days' worth of orders that the LTA may autonomously process.

			download from the Archive delivery Interface Point. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ The EstimatedDate is assigned upon initial order response and is not subsequently updated. The EstimatedDate is assigned on the basis of the retrieval queue when the order is processed (the actual CompletedDate could be impacted by subsequent incoming higher priority orders)	
CompletedDate	DateTimeOffset	N	Date and time when the product was available for download from the Archive delivery Interface Point. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ Only provided if the order is in status *completed*	`2018-01-22T18:00:00.000Z`
EvictionDate	DateTimeOffset	N	Date when the Product related to the order will be removed from the Archive delivery Interface Point. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ Only provided if the order is in status *completed*	`2018-01-22T18:00:00.000Z`
Priority	Int64	Y	Priority of the order. It is an integer from 1-100, where 100 is the highest priority. There is always a priority associated to an order: if it is not set within the Order Request, then it is automatically set to the default priority assigned to the user. If the Order Request sets a priority higher than the highest priority assigned to the user then it is automatically set at the highest priority assigned.	`50`
NotificationEndpoint	String	N	URI used by the LTA for product download readiness notifications, should these be required. If not provided, no notifications will be sent.	`Any URI, e.g.` *`https://myservice/notification`*
NotificationEpUsername	String	N	The username associated with the Endpoint URI provided. Mandatory if Endpoint is provided, and requires authentication.	`myserviceuser`

NotificationEpPassword	String	N	The password associated with the Endpoint URI provided. Shall be suitably secured to not be disclosed in subsequent queries via the OData interfaces.	***********

Table 3: Order Entity Description

Example Order Request:

```
POST \
http://<service-root-uri>/odata/v1/Products(ProductId)/OData.CSC.Order
{
  "Priority": 50
}
```

NB: ‘Priority’ is an optional argument in the order request

Example Order Response:

```
HTTP/1.1 201 Created
{
  "@odata.context": "$metadata#OData.CSC.Order",
  "Id": "2b17b57d-fff4-4645-b539-91f305c26x53",
  "Status": "in_progress",
  "StatusMessage": "request is under processing",
  "SubmissionDate": "2019-03-27T13:55:12.390Z",
  "EstimatedDate": "2019-03-27T14:02:51.390Z",
  "Priority": 50
}
```

Example Order Status Query:

The following URI will return all Orders with *completed* status:

```
https://<service-root-uri>/odata/v1/Orders?$filter=Status eq
OData.CSC.JobStatus'completed'
```

Example Order Status Query Response:

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Orders",
  "value":
  [
  {
    "Id": "2b17b57d-fff4-4645-b539-91f305c26x53",
    "Status": "completed",
    "StatusMessage": "requested product is available",
    "OrderSize": 4737286945,
    "SubmissionDate": "2019-03-27T13:55:12.390Z",
    "EstimatedDate": "2019-03-27T14:00:00.000Z",
    "CompletedDate": "2019-03-27T14:02:51.390Z",
    "EvictionDate": "2019-03-30T14:02:51.390Z",
    "Priority": 10
  }
  {
    "Id": "5e26g57d-fff4-4645-b539-91f305c72h49",
    "Status": "completed",
    "StatusMessage": "requested product is available",
    "OrderSize": 42562869174,
    "SubmissionDate": "2019-03-29T18:12:00.420Z",
    "EstimatedDate": "2019-03-29T20:12:00.000Z",
    "CompletedDate": "2019-03-29T20:12:18.055Z",
    "EvictionDate": "2019-04-01T14:02:51.390Z",
    "Priority": 10
  }
  ]
}
```

3.4.2 Product Download

When the ProductId (UUID) is known and the “Online” product property is “true”, a user can proceed with the product download request directly using the following query:

```
https://<service-root-uri>/odata/v1/Products(ProductId)/$value
```

On the other hand, if the product was offline and had to be ordered, once the data are staged the download over the OData API is initiated using the ‘Id’ for the Order. The correct query syntax for the download of a single product is:

```
https://<service-root-uri>/odata/v1/Orders(OrderId)/Product/$value
```

The AIP response supports the following HTTP status codes as a minimum:

- *200 OK*
- *202 Accepted (Product is being retrieved as part of an existing order but is not yet online)*
- *307 Temporary Redirect*
- *400 Bad Request (Product Id is not known in the LTA)*
- *404 Not Found (Product is not online)*
- *401 Unauthorized*
- *429 Too Many Requests (Quotas are exceeded)*
- *500 Internal Server Error*

http byte range requests on full product downloads are allowed, enabling pause/resume of downloads or the efficient split of large download into multiple chunks, and may be added as a Range Header to the Download Request, eg: “`Range: bytes=0-1023`”

3.5 Bulk Product Retrieve

The Bulk Product Retrieve function allows to trigger the bulk delivery of products from the LTA to the AIP, from where they can subsequently be downloaded. The bulk is defined by a list of products or query filter parameters and a batch size. The LTA is free to determine the order of the listing of products in the batches. The client triggers the staging of the batches using the BatchOrder action trigger. Instead of the standard Product Order, a BatchOrder internally optimizes access to and retrieval from the LTA backend. As a consequence, a BatchOrder is the smallest entity of a Bulk Request.

The sequence diagram below shows the process of a Bulk Request being split by the LTA into BatchOrders, from which products are downloaded following the triggering of a BatchOrder and associated product staging.

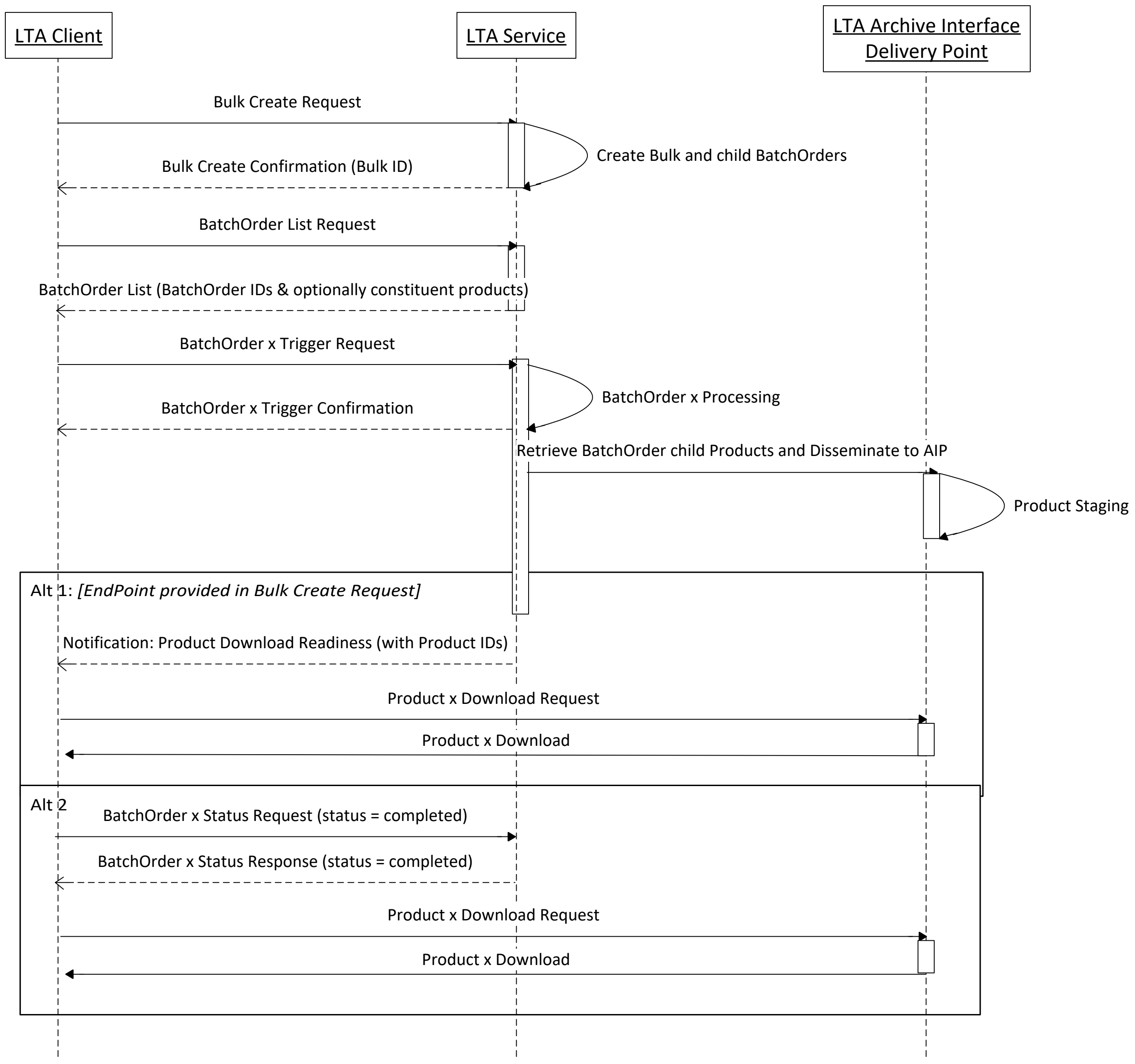

LTA Client
LTA Service
LTA Archive Interface Delivery Point
Bulk Create Request
Create Bulk and child BatchOrders
Bulk Create Confirmation (Bulk ID)
BatchOrder List Request
BatchOrder List (BatchOrder IDs & optionally constituent products)
BatchOrder x Trigger Request
BatchOrder x Processing
BatchOrder x Trigger Confirmation
Retrieve BatchOrder child Products and Disseminate to AIP
Product Staging
Alt 1: [EndPoint provided in Bulk Create Request]
Notification: Product Download Readiness (with Product IDs)
Product x Download Request
Product x Download
Alt 2
BatchOrder x Status Request (status = completed)
BatchOrder x Status Response (status = completed)
Product x Download Request
Product x Download

Figure 6: Bulk Product Retrieval Nominal Sequence

The nominal scenario can be described in the following steps:

- **STEP 1 – Bulk Create**
  The scenario is initiated by an LTA Client submitting a Bulk Create Request to the LTA service. While the standard Order described in Section 3.4.1 is a request for a single existing product from the LTA Service, a Bulk is a request for all existing products stored in an LTA which fulfil a defined set of filter parameters. As such, the Bulk request contains the filter parameters of the order (product types, time range, footprint, etc.), a batch size defining the size of individual BatchOrders making up the Bulk (in values for total batch volume and number of products making up the BatchOrder) and an optional endpoint through which the LTA Client can be notified about product download readiness.

  The LTA Service processes the Bulk Create Request, identifying all products fulfilling the Bulk and dividing them into 'BatcheOrders'. The BatchOrder division depends on the parameters of the initial Bulk Request and any further criteria (e.g. sorting to maximize retrieval efficiency etc.) imposed at LTA Service level, for example the quota available to the user. A confirmation response is sent to the LTA Client, containing the unique identifier of the Bulk, a status (initially *created*), the filter parameters, the batch size (products and volume), a submission date/time.

- **STEP 2 – BatchOrder Query and BatchOrder Triggering**
  The Client performs a query to obtain the list of child BatchOrders making up the Bulk, each containing a unique BatchOrder identifier. Prior to triggering, the BatchOrders are initially in the *created* status. Using the BatchOrder IDs, the list of products making up each BatchOrder may also be retrieved.

  The LTA Client may now trigger individual BatchOrders in the list, via the submission of the unique BatchOrder identifier and, optionally, a priority which applies to the while BatchOrder. Triggering the BatchOrder causes the status to be updated to *in_progress* and the LTA Service to begin the process of retrieving the child products and disseminating them to the AIP. A confirmation response is sent to the client. The LTA Service organizes the retrieval/dissemination of all products within any particular BatchOrder in a single sequence, without interruption from other requests.

- **STEP 3 – Product Staging Notification and Product Retrieval**
  There are two options via which the LTA Client can be made aware of product availability for download:

  - *ALTERNATIVE 1 – Product Staging Notification*
    Following the staging of every product making up the BatchOrder, the LTA Service sends the Client a Product Download Readiness Notification containing the Product ID of the staged product (and an Eviction Date (TBC)). The Client is then free to submit a product Download Request to trigger the download of each product, using the Product ID (see section 3.4.2). This option is applicable if an Endpoint URI has been provided with the Bulk Create Request.

  - *ALTERNATIVE 2 – BatchOrder Status Query*
    The Client routinely sends queries requesting the status of the BatchOrders. Once a BatchOrder status is *completed*, the products have been successfully staged. The Client may then submit a product Download Request to trigger the download of any product within the BatchOrder using the specific Product ID (see section 3.4.2).

- The LTA Client can request the status of the Bulk or particular BatchOrder at any time. A *completed* status for the BatchOrder signifies that all products associated with that particular Batch Order have been successfully staged on the AIP. A *completed* status for Bulk signifies that all BatchOrders making up the Bulk are themselves in the *completed* status. However, a completed status for the Bulk does not signify that the products of a specific BatchOrder are still available via the AIP, because although the AIP must be large enough to hold an entire batch of products, it is not required that it can also hold an entire bulk of products. It is the responsibility of the client to employ flow control if bulks are too large by deferring BatchOrders while products are still being downloaded from the AIP.

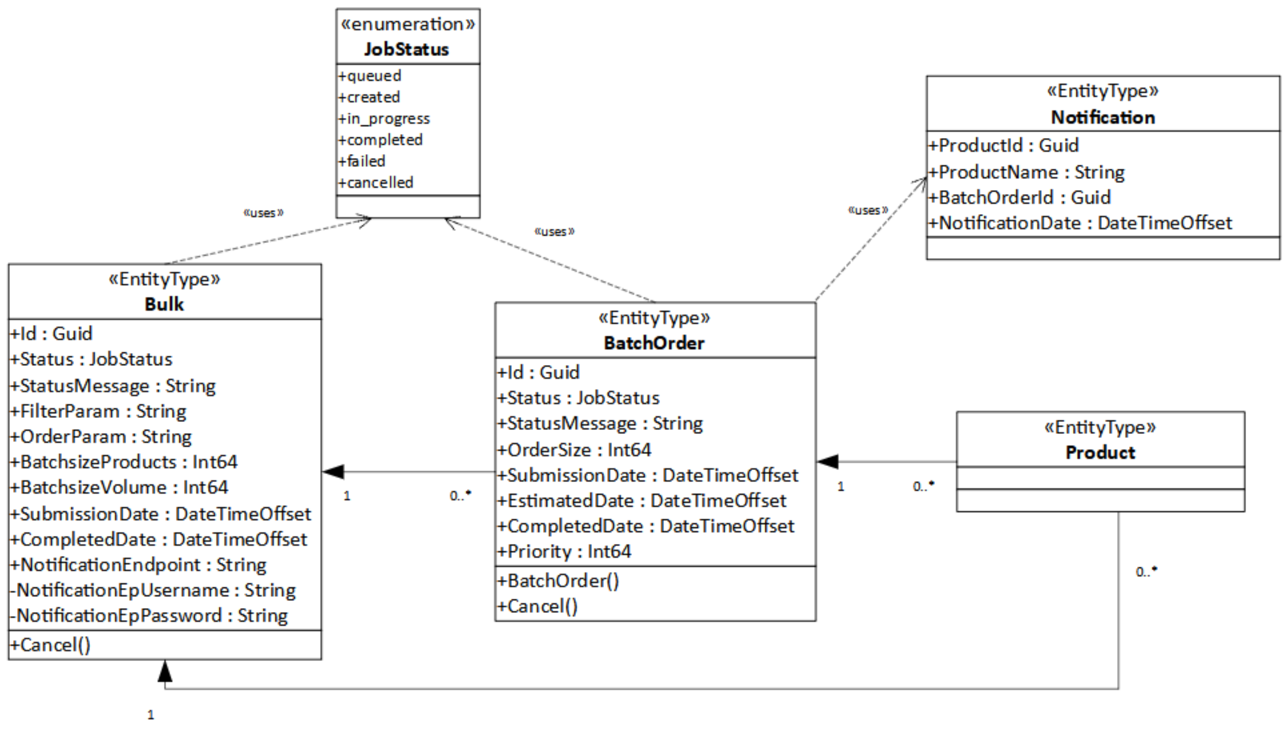


Figure 7: Bulk & BatchOrder Model

3.5.1 Bulk and BatchOrder Model and Properties

Bulk Properties	Type	M	Description	Example
Id	Guid	Y	It is a universally unique identifier (UUID). The Id is a local identifier for the Bulk instance within the LTA, assigned upon Bulk creation.	`2b17b57d-fff4-4645-b539-91f305c27c69`
Status	JobStatus enumeration	Y	JobStatus value: - *created* - *in_progress* - *completed* - *failed* - *cancelled*	`in_progress`

			Where: - *created* means that all child BatchOrders are still themselves in status *created* - *in_progress* means that one of more child BatchOrders are themselves in status *in_progress* - *completed* means that all child BatchOrders are in the status *completed* - *failed* means that one or more child BatchOrders in in status *failed* - *cancelled* means that the Bulk has been cancelled by the user (while in status *created* or *in_progress*). If *in_progress* then any BatchOrders that are *in_progress* are still completed but further BatchOrders are rejected.	
StatusMessage	String	Y	Text message providing additional information on the returned status, e.g. the reason for a failure. Example values are: *If status = 'created'* 'Bulk has been created but no BatchOrders have been triggered' *If status = 'in_progress'* 'One or more BatchOrders are under processing' *If status = 'completed':* 'requested products of all BatchOrders comprising the Bulk are in status 'completed'' *If status = 'failed':* 'at least one product retrieval has failed' 'at least one product currently unavailable' 'at least one product not found on LTA' *If status = cancelled:* 'Bulk cancelled by user'	`One or more BatchOrders are under processing`
FilterParam	String	Y	The filter parameters of the Bulk request (refers to the $filter= parameter of any Products? query)	`contains(Name,'_MSI_L0__GR_')and PublicationDate gt 2018-01-17T00:00:00.000Z and PublicationDate lt 2018-01-20T00:00:00.000Z`
OrderbyParam	String	N	Optional specification of the sorting order for the products within the batches. This would override any ordering from the LTA and could result in a less efficient Bulk extraction, so should be used only if critical for the organisation of the Bulk (refers to the	`PublicationDate desc`

			$orderby= parameter of any Products? query)	
BatchsizeProducts	Int64	N	The maximum number of products making up each child BatchOrder making up the Bulk. If this number causes the BatchsizeVolume to be exceeded then the number is limited to that which make up the BatchsizeVolume. The BatchsizeProducts serves as an optional information of the LTA client to the LTA service. The LTA service shall be eligible to overwrite the value specified by the client. If not provided, a configurable default LTA BatchsizeProducts value is used.	`30`
BatchsizeVolume	Int64	N	The maximum volume of each child BatchOrder making up the Bulk, in GiB. This value takes precedence of the BatchsizeProducts value. The BatchsizeVolume serves as an optional information of the LTA client to the LTA service. The LTA service shall be eligible to overwrite the value specified by the client. If the value exceeds a configurable default maximum LTA BatchsizeVolume value, then the BatchsizeVolume is limited to this maximum value. If not provided then the maximum default BatchsizeVolume value is used.	`50`
SubmissionDate	DateTimeOffset	Y	Date and time at which the bulk request was received by the LTA. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T14:46:03.788Z`
CompletedDate	DateTimeOffset	N	Date and time when all child BatchOrders were in status *completed*. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ Only provided when status is *completed*.	
NotificationEndpoint	String	N	URI used by the LTA for product download readiness notifications, should these be required. If not provided, no notifications will be sent.	`Any URI, e.g.` *`https://myservice/notification`*
NotificationEpUsername	String	N	The username associated with the Endpoint URI provided. Mandatory if Endpoint is provided, and requires authentication.	`myserviceuser`

NotificationEpPassword	String	N	The password associated with the Endpoint URI provided. Shall be suitably secured to not be disclosed in subsequent queries via the OData interfaces.	`***********`

Table 4: Bulk Entity Description

BatchOrder Properties	Type	M	Description	Example
Id	Guid	Y	It is a universally unique identifier (UUID). The Id is a local identifier for the BatchOrder instance within the LTA, assigned upon BatchOrder creation.	`2b17b57d-fff4-4645-b539-91f305c27c6982`
Status	JobStatus enumeration	Y	JobStatus value: - *queued* - *in_progress* - *completed* - *failed* - *cancelled* Where: - *queued* means the BatchOrder is queued and has not yet been triggered - *in_progress* means that the products making up the BatchOrder are being retrieved and disseminated to the AIP - *completed* means that all products making up the BatchOrder were successfully disseminated to the AIP and are available for download - *failed* means the retrieval and dissemination of one or more products making up the BatchOrder has failed - *cancelled* means that the BatchOrder has been cancelled by the user (while in status *queued* or *in_progress*)	`in_progress`
StatusMessage	String	Y	Text message providing additional information on the returned status, e.g. the reason for a failure. Example values are: *If status = 'queued':* 'BatchOrder request is queued' *If status = 'in_progress':* 'BatchOrder is under processing' *If status = 'completed':* 'requested products of BatchOrder are available'	`requested products of BatchOrder are available`

			If status = 'failed': 'at least one product retrieval has failed' 'at least one product currently unavailable' 'at least one product not found on LTA' *If status = cancelled:* 'BatchOrder cancelled by user'	
OrderSize	Int64	N	Sum of actual size in bytes (B) of the data of all products making up the BatchOrder. Only provided if the BatchOrder is in status *completed*	`4737286945`
SubmissionDate	DateTime Offset	N	Date and time at which the BatchOrder was triggered (i.e. changed from status *created* to status *in_progress*). No SubmissionDate is provide when the BatchOrder is in status *created*, or in status *cancelled* should the BatchOrder have been cancelled before it was triggered.Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T14:46:03.788Z`
EstimatedDate	DateTime Offset	N	Estimated date and time when the products making up the BatchOrder will be available for download from the Archive delivery Interface Point. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ The EstimatedDate is assigned when the BatchOrder is triggered (it is not provided for status *created*) and is not subsequently updated. The EstimatedDate is assigned on the basis of the retrieval queue when the BatchOrder is processed (the actual CompletedDate could be impacted by subsequent incoming higher priority orders)	`2018-01-22T18:00:00.000Z`
CompletedDate	DateTime Offset	N	Date and time when the products composing the BatchOrder were all available for download from the Archive delivery Interface Point. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ Only provided when Status is *completed*.	`2018-01-22T18:00:00.000Z`
Priority	Int64	N	Priority of the BatchOrder. It is an integer from 1-100, where 100 is the highest priority.	`10`

			There is no priority associated with a BatchOrder when it is in status created. In other statuses there is always a priority associated to a BatchOrder: if it is not set within the BatchOrder triggering request, then it is automatically set to the default priority assigned to the user. If the BatchOrder triggering request sets a priority higher than the highest priority assigned to the user then it is automatically set at the highest priority assigned.	

Table 5: BatchOrder Entity Description

3.5.2 Bulk and BatchOrder Examples

3.5.2.1 Step 1 – Bulk Create

i-i Bulk Create: Example Request:

```
POST \
http://<service-root-uri>/odata/v1/Bulks
{
  "@odata.context": "$metadata#Bulk/$entity",
  "FilterParam": "contains(Name,'_MSI_L0__GR_')and PublicationDate gt 2018-01-
17T00:00:00.000Z and PublicationDate lt 2018-01-20T00:00:00.000Z",
  "BatchsizeProducts": 30,
  "BatchsizeVolume": 50,
  "NotificationEndpoint": "https://myservice/notification",
  "NotificationEpUsername": "myserviceuser",
  "NotificationEpPassword": "*********"
}
```

NB: the example above will trigger Batch Staging Notifications as the optional NotificationEndpoint URL is provided. If not such NotificationEndpoint is provided with the initial Bulk Create request then no Notifications will later be sent.

i-ii Bulk Create: Example Response (JSON format):

```
HTTP/1.1 201 Created
{
  "@odata.context": "$metadata#OData.CSC.Bulk",
  "value":
  [
  {
  "Id": "2b17b57d-fff4-4645-b539-91f305c27c69",
  "Status": "created",
  "StatusMessage”: "Bulk has been created but no BatchOrders have been triggered",
  "FilterParam": "contains(Name,'_MSI_L0__GR_')and PublicationDate gt 2018-01-
17T00:00:00.000Z and PublicationDate lt 2018-01-20T00:00:00.000Z",
  "BatchsizeProducts": 30,
  "BatchsizeVolume": 50,
  "NotificationEndpoint": "https://myservice/notification",
  "SubmissionDate": "2019-05-27T18:25:30.000Z"
  }
  ]
}
```

3.5.2.2 Step 2 – BatchOrder Query and BatchOrder Triggering

i-i BatchOrders within a Bulk Query: Example Request:

```
http://<service-root-uri>/odata/v1/Bulks(2b17b57d-fff4-4645-b539-
91f305c27c69)/BatchOrders
```

i-ii BatchOrders within a Bulk Query: Example Response (JSON format):

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#BatchOrder/$entity",
  "value":
  [
  {
    "Id": "2b17b57d-fff4-4645-b539-91f305c83z91",
    "Status": "in_progress",
    "StatusMessage": "BatchOrder is under processing",
    "SubmissionDate": "2019-05-27T19:25:30.000Z",
    "EstimatedDate": "2019-05-27T19:35:30.000Z",
    "Priority": 50
  },
  {
    "Id": "2b17b57d-fff4-4645-b539-91f305c68f44",
    "Status": "queued",
    "StatusMessage": "BatchOrder request is queued"
  }
  ]
}
```

ii-i BatchOrder Query: Example Request

```
http://<service-root-uri>/odata/v1/BatchOrders(2b17b57d-fff4-4645-b539-
91f305c27d64)
```

ii-ii BatchOrder Query: Example Response

```
 HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#BatchOrder/$entity",
  "value":
  [
  {
  "Id": "2b17b57d-fff4-4645-b539-91f305c83z91",
  "Status": "in_progress",
  "StatusMessage": "BatchOrder is under processing",
  "SubmissionDate": "2019-05-27T19:25:30.000Z",
  "EstimatedDate": "2019-05-27T19:35:30.000Z",
  "Priority": 50
  }
  ]
}
```

iii-i BatchOrder Query – Constituent Products: Example Request

```
http://<service-root-uri>/odata/v1/BatchOrders(2b17b57d-fff4-4645-b539-
91f305c27d64)/Products
```

iii-ii BatchOrder Query – Constituent Products: Example Response (JSON format)

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Products",
  "value":
  [
  {Product1}
  {Product2}
  {Product3}
  {Product4}
  {Product5}
  {…}
  ]
}
```

See section 3.3.1.8 for the full set product of attributes returned in the response.

iv-i BatchOrder Triggering: Example Request:

BatchOrders are created upon creation of the parent Bulk, but require subsequent triggering to change the status from *created* to *in_progress* and trigger the product retrieval and dissemination to the AIP. Optionally, a priority is provided with the triggering request. BatchOrder triggering is achieved as follows:

```
POST \
http://<service-root-uri>/odata/v1/BatchOrders(2b17b57d-fff4-4645-b539-
91f305c27e65)/OData.CSC.BatchOrder
{
   "Priority": 10
}
```

iv-ii Batch Order: Example Response (JSON format):

```
HTTP/1.1 204 No Content
```

While the BatchOrder parameters are not provided in the triggering response, it should be noted that a BatchOrder entity will be updated as follows upon triggering:

- The **Status** (and corresponding **StatusMessage**) will be updated from *created* to *in_progress*
- A **Priority** will be included, either corresponding to the priority provided with the triggering request, or the default priority assigned to the user.
- A **SubmissionDate** will be provided, corresponding to the date/time of the triggering request receipt.
- An **EstimatedDate** will be provided.

3.5.2.3 Step 3 – Product Staging Notification and Product Retrieval

i-i BatchOrder Completion Query

If no Endpoint was provided in the Bulk (meaning no Product Staging Notifications will be send), all BatchOrders in completed status (i.e. all child products have been staged) can be retrieved as follows:

```
http://<service-root-uri>/odata/v1/BatchOrders?$filter=CompletedDate gt 2019-05-
28T20:00:00.000Z
```

i-ii BatchOrder Completion Query Response

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#BatchOrder/$entity",
  "value":
  [
  {
    "@odata.context": "$metadata#OData.CSC.Order",
    "Id": "2b17b57d-fff4-4645-b539-91f305c27c79",
    "Status": "completed",
    "StatusMessage": "requested products of BatchOrder are available",
    "OrderSize": 24737286691
    "SubmissionDate": "2019-05-28T18:12:00.220Z",
    "EstimatedDate": "2019-05-28T20:00:00.000Z",
    "CompletedDate": "2019-05-28T20:08:51.390Z",
    "Priority": 10
  }
  {
    "@odata.context": "$metadata#OData.CSC.Order",
    "Id": "2b17b57d-fff4-4645-b539-91f305c27c56",
    "Status": "completed",
    "StatusMessage": "requested products of BatchOrder are available",
    "OrderSize": 34737286945,
    "SubmissionDate": "2019-05-28T18:12:00.420Z",
    "EstimatedDate": "2019-05-28T20:00:00.000Z",
    "CompletedDate": "2019-05-28T20:12:18.055Z",
    "Priority": 10
  }
  ]
}
```

ii Example Product Download Readiness Notification:

```
POST \
http://myservice/notification
{
  "@odata.context": "$metadata#Notification/$entity",
  "BatchOrderId": "9e17b57d-fff4-4645-b539-91f305c76g22",
  "ProductId": "e872683a-ea5b-455d-bfb2-fb304cbfbacb",
  "Name": "S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6",
  "NotificationDate": "2019-05-30T22:18:09.000Z"
}
```

See Table 7 for details on the Notification entity.

3.6 Subscription

The Subscription function allows for the notification of future products entering the LTA, or products being deleted from it, according to the set of filter parameters supplied within the subscription request. The request can either trigger the staging of all such products (with download readiness notification) or just provide a product availability notification for each product, which may then be optionally ordered.

The sequence diagram below illustrates the subscription process.

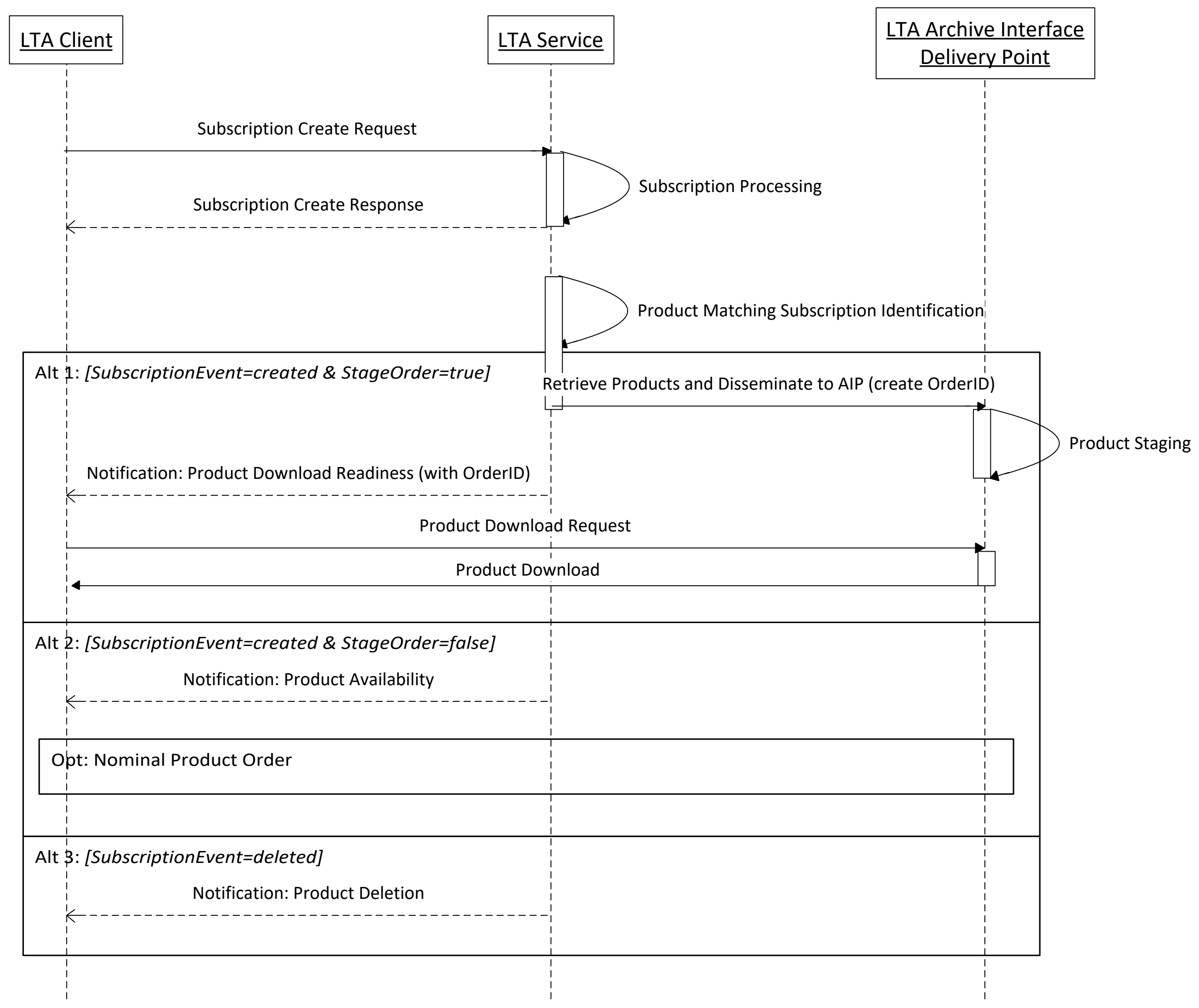

LTA Client
LTA Service
LTA Archive Interface Delivery Point
Subscription Create Request
Subscription Processing
Subscription Create Response
Product Matching Subscription Identification
Alt 1: [SubscriptionEvent=created & StageOrder=true]
Retrieve Products and Disseminate to AIP (create OrderID)
Product Staging
Notification: Product Download Readiness (with OrderID)
Product Download Request
Product Download
Alt 2: [SubscriptionEvent=created & StageOrder=false]
Notification: Product Availability
Opt: Nominal Product Order
Alt 3: [SubscriptionEvent=deleted]
Notification: Product Deletion

Figure 8: Subscription Nominal Sequence

The nominal scenario can be described in the following steps:

- **STEP 1 – Subscription Create**
  The scenario is initiated by an LTA Client submitting a create subscription request to the LTA service. The request contains the SubscriptionEvent choice (either created or deleted), filter parameters of the subscription (product types, time range, footprint, etc.), a selection to automatically stage newly

created products (true/false) and a NotificationEndpoint to which product availability/download/deletion notifications will be sent. The request is processed by the LTA Service, and a response returned which includes a unique identifier for the particular request, a status (initially *running*), the filter parameters, the submission date/time of the request and the stage order selection.

- **STEP 2 – Product Notification and Product Retrieval**
  Upon future identification of a new product creation or deletion matching the filter parameters of the subscription, the response of the LTA Service depends on whether or not it has been requested to automatically stage the products:

  - If *SubscriptionEvent=created & StageOrder=true*, the LTA Service triggers the staging of the identified product and dissemination to the AIP and, once this is complete, sends a product download readiness notification to the endpoint provided in the subscription request containing the unique OrderID of the staged product (also updating the Eviction Date within the Order entity). The Client is then free to submit a product Download Request to trigger the download of that product (see section 3.4.2).

  - If *SubscriptionEvent=created and StageOrder=false*, the LTA service sends a product availability notification to the endpoint provided in the subscription request. The Client is then free to order the retrieval of the product according to the nominal product retrieval scenario (see section 3.4).

  - If *SubscriptionEvent=deleted*, the LTA service sends a product deletion notification to the endpoint provided in the subscription request.

- A subscription continues to identify products matching the filter parameters until it is paused or cancelled.
- In case the client endpoint is unavailable and the subscription notification fails, there is no retry and the client may need to periodically check any local list of products maintained through a subscription.
- Clients shall be able to query the status of orders initiated from subscriptions.

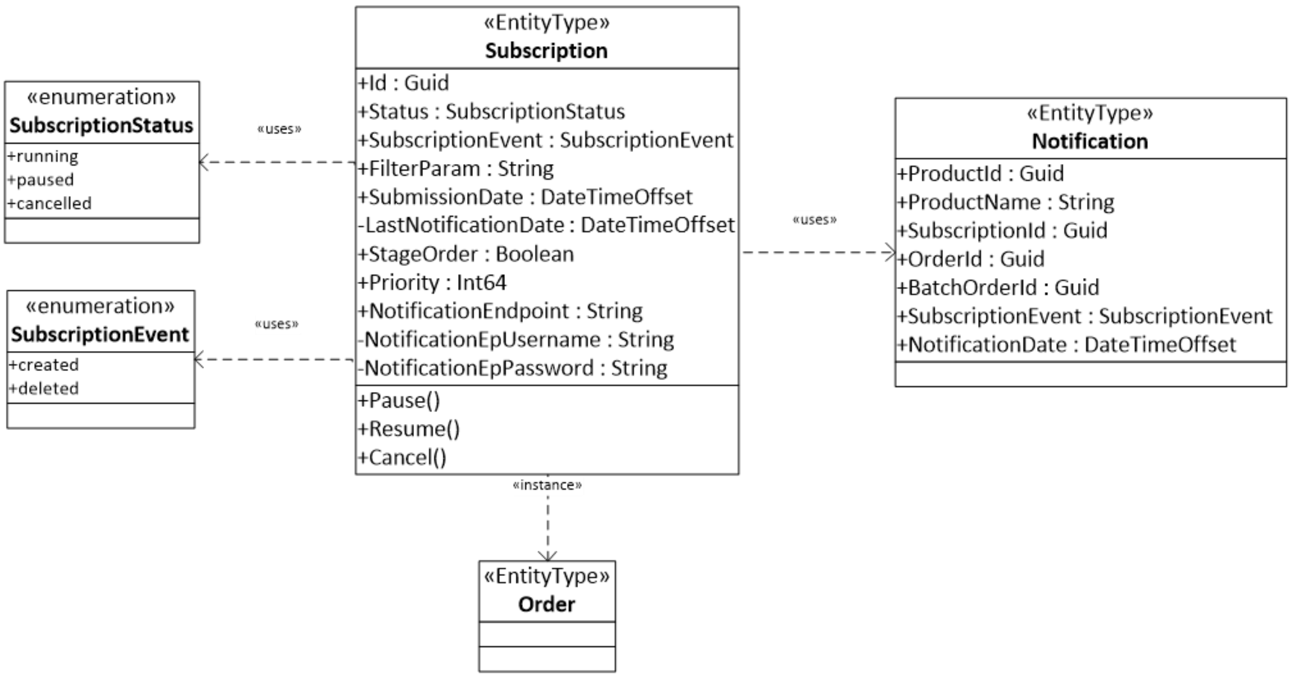


Figure 9: Subscription Model

Subscription Properties	Type	M	Description	Example
Id	Guid	Y	It is a universally unique identifier (UUID). The Id is a local identifier for the Subscription instance within the LTA, assigned upon Subscription creation.	`2b17b57d-fff4-4645-b539-91f305c27e11`
Status	Subscription Status enumeration	Y	SubscriptionStatus value: - *running* - *paused* - *cancelled*	`running`
SubscriptionEvent	Subscription Event enumeration	Y	The subscription event to be monitored and for which notification is provided, from: - *created* - *deleted*	`created`
FilterParam	String	Y	The filter parameters of the Subscription (refers to the $filter= parameter of any Products? query)	`contains(Name,'_MSI_L0__GR_')and PublicationDate gt 2019-07-01T00:00:00.000Z and PublicationDate lt 2018-09-01T00:00:00.000Z`
SubmissionDate	DateTimeOffset	Y	Date and time at which the Subscription was received by the LTA. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T14:46:03.788Z`
LastNotificationDate	DateTimeOffset	N	Date and time corresponding to the last time the Subscription was queried. Used by the LTA to limit	`2018-02-05T09:00:00.000Z`

			the next query from the LastNotificationDate to the current date. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ Only provided following the initial Subscription query.	
StageOrder	Boolean	Y	Automatically orders the staging of products fulfilling the subscription filter parameters Only used if SubscriptionEvent = created	`true`
Priority	Int64	N	Priority of the created orders resulting from the subscription. It is an integer from 1-100, where 100 is the highest priority. Only provided if: - SubscriptionEvent = created AND - the Subscription has stageOrder=true There is always a priority associated to an order: if it is not set within the Subscription Create Request, then it is automatically set to the default priority assigned to the user. If the Subscription Create Request sets a priority higher than the highest priority assigned to the user then it is automatically set at the highest priority assigned.	`10`
NotificationEndpoint	String	Y	URI used by the LTA for subscription notifications	`Any URI, e.g.` *`https://myservice/notification`*
NotificationEpUsername	String	N	The username associated with the Endpoint URI provided Mandatory if NotificationEndpoint requires authentication	`myserviceuser`
NotificationEpPassword	String	N	The password associated with the Endpoint URI provided Mandatory if NotificationEndpoint requires authentication	`***********`

Table 6: Subscription Entity Description

Notification Properties	Type	M	Description	Example
ProductId	Guid	Y	The universally unique identifier (UUID) of the product being notified. Maps to Product.Id	`2b17b57d-fff4-4645-b539-91f305c27c69`
ProductName	String	Y	The data file name of the product being notified. Maps to Product.Name	`S1A_IW_SLC__1SDV_20160117T103451_20160117T103518_009533_00DD94_D46A.SAFE.zip`
SubscriptionId	Guid	N	The UUID of the Subscription being notified. Maps to Subscription.Id Mandatory for Subscription Notifications; not provided for Batch Order Notifications	`2b17b57d-fff4-4645-b539-91f305c27e11`
OrderId	Guid	N	The UUID of the order being notified. Maps to Order.Id Only provided for Subscription notifications if: - SubscriptionEvent = created AND - the Subscription has StageOrder=true Not provided for BatchOrder notifications	`2b17b57d-fff4-4645-b539-91f305c54x22`
BatchOrderId	Guid	N	The UUID of the BatchOrder being notified. Maps to BatchOrder.Id Not provided for Subscription Notifications; mandatory for BatchOrder notifications	`5b17b57d-fff4-4645-b539-91f305c54x33`
SubscriptionEvent	Subscription Event enumeration	N	The subscription event set in the parent subscription to be monitored and for which notification is provided, from: - *created* - *deleted* Mandatory for Subscription Notifications; not provided for Batch Order Notifications	`created`
NotifictionDate	DateTimeOffset	Y	Date and time at which the notification was generated. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2019-02-05T09:00:00.000Z`

Table 7: Notification Entity Description

i-i Subscription Create: Example Request:

```
POST \
http://<service-root-uri>/odata/v1/Subscriptions
{
  "@odata.context": "$metadata#Subscription/$entity",
  "FilterParam": "contains(Name,'_MSI_L0__GR_')and PublicationDate gt 2019-07-
01T00:00:00.000Z and PublicationDate lt 2019-09-01T00:00:00.000Z",
  "StageOrder": true,
  "Priority": 10,
  "NotificationEndpoint": "https://myservice/notification",
  "NotificationEpUsername": "myserviceuser",
  "NotificationEpPassword": "**********"
}
```

i-ii Subscription Create: Example Response (JSON format):

```
HTTP/1.1 201 Created
{
  "@odata.context": "$metadata#OData.CSC.Subscription",
  "Id": "2b17b57d-fff4-4645-b539-91f305c27e11",
  "Status": "running",
  "FilterParam": "contains(Name,'_MSI_L0__GR_')and PublciationDate gt 2019-07-
01T00:00:00.000Z and PublicationDate lt 2019-09-01T00:00:00.000Z",
  "StageOrder": true,
  "Priority": 10,
  "SubmissionDate": "2019-05-27T18:25:30.000Z",
  "Endpoint": "https://myservice/notification"
}
```

ii-i Example Subscription Notification: Product Download Readiness:

```
POST \
http://myservice/notification
{
  "@odata.context": "$metadata#Notification/$entity",
  "SubscriptionEvent": "created",
  "ProductId": "e872683a-ea5b-455d-bfb2-fb304cbfbacb",
  "ProductName":
"S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6",
  "OrderId": "9e17b57d-fff4-4645-b539-91f305c76g22",
  "SubscriptionId": "2b17b57d-fff4-4645-b539-91f305c27e11",
  "NotificationDate": "2019-05-30T22:18:09.000Z"
}
```

ii-ii Example Subscription Notification: Product Availability:

```
POST \
http://myservice/notification
{
  "@odata.context": "$metadata#Notification/$entity",
  "SubscriptionEvent": "created",
  "ProductId": "e872683a-ea5b-455d-bfb2-fb304cbfbacb",
  "ProductName":
"S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6",
  "SubscriptionId": "2b17b57d-fff4-4645-b539-91f305c27e11",
  "NotificationDate": "2019-05-30T22:18:09.000Z"
}
```

ii-iii Example Subscription Notification: Product Deletion:

```
POST \
http://myservice/notification
{
  "@odata.context": "$metadata#Notification/$entity",
  "SubscriptionEvent": "deleted",
  "ProductId": "e872683a-ea5b-455d-bfb2-fb304cbfbacb",
  "ProductName":
"S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6",
  "SubscriptionId": "2b17b57d-fff4-4645-b539-91f305c27e11",
  "NotificationDate": "2019-05-30T22:18:09.000Z"
}
```

iii-i Subscription StageOrder Query: Example Request

```
http://<service-root-uri>/odata/v1/Subscription(‘2b17b57d-fff4-4645-b539-
91f305c27c69’)/Orders
```

iii-ii Subscription StageOrder Query: Example Response

```
{
  "@odata.context": "$metadata#Orders",
  "value":
  [
  {
    "Id": "2b17b57d-fff4-4645-b539-91f305c26x53",
    "Status": "completed",
    "StatusMessage”: ”requested product is available”,
    "OrderSize”: 4737286945,
    "SubmissionDate”: ”2019-03-27T13:55:12.390Z”,
    "CompletedDate”: ”2019-03-27T14:02:51.390Z”,
    "EvictionDate”: ”2019-03-30T14:02:51.390Z”,
    "Priority": 10
  }
  {
    "Id": "5e26g57d-fff4-4645-b539-91f305c72h49",
    "Status": "in_progress",
    "StatusMessage": "request is under processing",
    "OrderSize": 42562869174,
    "SubmissionDate": "2019-03-29T18:12:00.420Z",
    "EstimatedDate": "2019-03-29T20:12:18.055Z",
    "Priority": 10
  }
  ]
}
```

3.7 Traceability Interface

The LTA service interfaces with the Traceability System [AD-7] to verify the origin of any data before ingestion and provide its own traceability record once the data is ingested, retrieved or deleted.

3.8 User Administration

The LTA service enforces that only properly authenticated clients are accepted and that the access rights are applied. If a client calls an LTA service function without permission, it is informed with a corresponding return message.

3.8.1 Basic LTA User[6] Entity Model

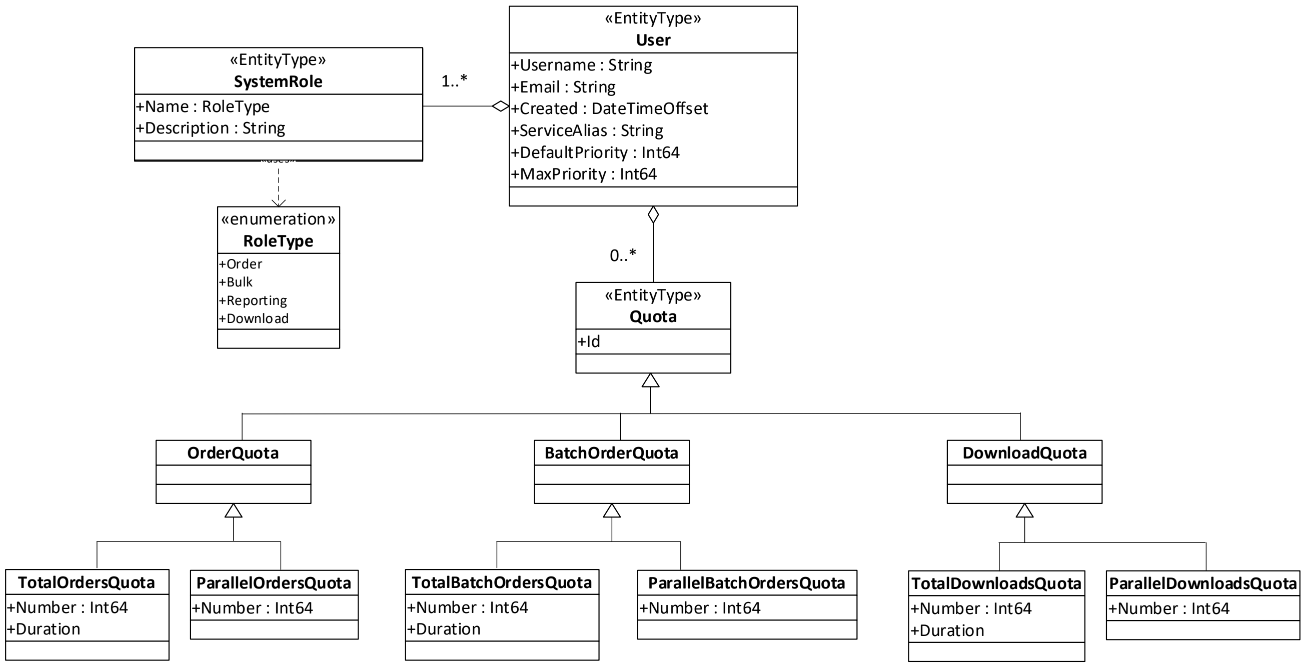


Figure 10: User Entity Model

The LTA User entity is a read-write entity that allows to define the system role of the user. It is identified by the key property Username. The properties assigned to each User entity are described in the table below:

User Properties	Type	M	Description	Example
Username (key)	String	Y	Username of the User	`DataDissem4CollGSHub`
Email	String	Y	Email of the User	`datahubops@xyz.com`
Created	DateTimeOffset	Y	Date and time of User creation	
ServiceAlias	String	Y	Generic string to describe the client	`DataAccessColhub`
DefaultPriority	Int64	Y	The default order priority assigned to the User. It is an integer from 1-100, where 100 is the highest priority.	`50`

[6] In this section, the User entity represents the Client

MaxPriority	Int64	Y	The maximum priority which can be assigned to an order by the User. It is an integer from 1-100, where 100 is the highest priority. It must be equal to or higher than the DefaultPriority	`100`

Table 8: User Entity Description

Usernames associated email are to be managed as generic indicators of client and not personal information.

The ServiceAlias value can be identical to the Username, unless indicated otherwise by the Agency.

The SystemRole sub-entity, which is defined for registered Users, is described in the sections below.

3.8.1.1 SystemRole

The SystemRole entity describes the rights and capabilities granted to a user. The properties of the SystemRole entity are described in the table below:

SystemRole Properties	Type	M	Description	Example
Name (key)	RoleType enumeration	Y	Assignment of the rights and capabilities granted to a user, including: - Order - Bulk - Reporting - Download	`Bulk`
Description	String	Y	Textual description of the role	`Allowing the user to make bulk requests`

Table 9: SystemRole Entity Description

All registered clients must have one or more roles assigned. The roles available for assignment to clients may be:

Client Services

- **Order –** The client may perform queries on the products stored within the LTA, place orders for product retrieval and create subscriptions. Users are only able to query the orders and subscriptions they themselves have created.
- **Bulk** – As above, and additionally able to place Bulk product retrieval requests, and query them.

Administration Services

- **(Monitoring &) Reporting –** Monitoring and Reporting functionality, including the permission to perform queries on all Orders, Subscriptions and Bulks which have been created by all users of the LTA.

3.8.2 Quota

User quota is assigned on a per user basis. Quota is assigned and applied to downloads for all users but may also be separately assigned applied to Orders or BatchOrders for users with the 'Order' or 'Bulk' SystemRole respectively. Therefore, users may be assigned the 'OrderQuota', 'BatchOrderQuota' and 'DownloadQuota', as shown below. 'OrderQuota'/'BatchOrderQuota' and 'DownloadQuota' are not linked: they are applied separately and independently at the order and download stages respectively. The quotas are not mutually exclusive and multiple or none may be assigned.

Order Quota *(applicable to users with the 'Order' SystemRole)*

- *TotalOrdersQuota*: Maximum number of authorized Orders which can be made in a defined period
- *ParallelOrdersQuota*: Maximum number of separate Orders with status '*in_progress*' which can be running in parallel.

BatchOrder Quota *(applicable to users with the 'Bulk' SystemRole)*

- *TotalBatchOrdersQuota*: Maximum number of authorized BatchOrders which can be triggered in a defined period
- *ParallelBatchOrdersQuota*: Maximum number of separate BatchOrders with status '*in_progress*' which can be running in parallel.

Download Quota *(only applicable to users with the 'Order' and/or 'Bulk' SystemRole)*

- *TotalDownloadsQuota*: Maximum volume (GB) transfer within a defined period
- *ParallelDownloadsQuota*: Maximum number of separate downloads which can be performed in parallel

Should the quota be exceeded the *429 Too Many Requests* code will be returned and the client must manage resubmission of the Order/BatchOrder or Download request at a later date.

NB: if an Order or BatchOrder is not yet in *in_progress* status, it may be cancelled without impact on the Order quota.

3.9 Monitoring and Reporting

The Monitoring and Reporting role allows for the collection of data, for analysis and reporting, of LTA system activity through API queries on all Products, Orders, Subscriptions and Bulks managed by the system. When used for the reporting function the API allows the collection of data concerning the overall use of the system by registered users; used for the monitoring function the API allows to regularly collect information concerning indicators which may be used to assess whether the routine operation of the system is running within the expected bounds, such as the ingestion rate of products.

The parameters obtained for Monitoring and Reporting through the API, with example requests, are detailed in the sections below.

3.9.1 Reporting

The table below provides a possible schema for routine collection of reporting information, example queries are developed further. The time period is indicative, depending on the strategy for end-to-end monitoring and reporting.

System Entity	Reporting Statistic	Time Period	Entity Parameters Collected
Products	Products ingested, including latency for ingestion (PublicationDate-OriginDate)	10 minutes	- Count - Name - ContentLength (Product size) - OriginDate - PublicationDate - ContentDate/Start
Orders	Orders submitted	10 minutes	- Count - Username - Id (Order) - OrderSize - SubmissionDate - Product.Name
	Orders completed and time for completion	10 minutes	- Count - Username - Id (Order) - SubmissionDate - CompletedDate - Product.Name
	Orders failed	10 minutes	- Count - Username - Id (Order) - SubmissionDate - Product.Name
	Orders cancelled	10 minutes	- Count - Username - Id (Order) - SubmissionDate - Product.Name
	Orders pending for over 1 day	10 minutes	- Count - Username - Id (Order) - SubmissionDate - Product.Name
Bulk	Bulks submitted	10 minutes	- Count - Username - Id (Bulk)
	Bulks completed	10 minutes	- Count - Username - Id (Bulk)
	Bulks failed	10 minutes	- Count - Username - Id (Bulk)
BatchOrder	BatchOrders submitted (triggered)	10 minutes	- Count - Username - Id (BatchOrder)
	BatchOrders completed	10 minutes	- Count - Username

			- Id (BatchOrder)
	BatchOrders failed	10 minutes	- Count - Username - Id (BatchOrder)
Subscriptions	Subscription in status *running* (total)	10 minutes	- Count
	Subscriptions submitted (daily)	10 minutes	- Count - Username - Id (Subscription)

Table 10: LTA Reporting Statistics

Examples of the queries used to retrieve the described statistics are presented below. Descriptions are limited to a single entity and can straightforwardly be modified to the other entities reported on where necessary.

i Products published per day

The $count notation is used to return the total products published in a particular day, and the $select notation to select the product properties to be returned.

i-i Request

```
https://<service-root-uri>/odata/v1/Products?$count=true&$filter=PublicationDate  gt
2019-01-17T00:00:00.000Z and PublicationDate lt 2019-01-
18T00:00:00.000Z&$select=Name,ContentLength,OriginDate,PublicationDate,ContentDate/Start
```

i-ii Response (JSON format)

In the example, the count of products returned has been limited to two.

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Products/$entity",
  "@odata.count": 2,
  "value":
  [
  {
    "@odata.mediaContentType": "application/octet-stream",
    "Name": "S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6",
    "ContentType": "application/octet-stream",
    "ContentLength": 4063869137,
    "OriginDate": "2019-01-17T07:22:00.165Z",
    "PublicationDate": "2019-01-17T09:18:00.048Z",
    "ContentDate":
    {
      "Start": "2019-01-17T05:30:00.060Z"
    }
  }
  {
    "@odata.mediaContentType": "application/octet-stream",
    "Name": "S1B_IW_SLC__1SDV_20191224T235516_20191224T235892_003545_006104_0DP9",
    "ContentType": "application/octet-stream",
    "ContentLength": 4423569265,
    "OriginDate": "2019-01-17T08:01:02.306Z",
    "PublicationDate": "2019-01-17T10:25:00.066Z",
    "ContentDate":
    {
      "Start": "2019-01-17T08:34:51.922Z"
    }
  }
  ]
}
```

ii Orders Submitted per day

The $expand notation is used to return the required properties of the Product and User entities related to each Order matching the filter criteria.

ii-i Request

```
https://<service-root-uri>/odata/v1/Orders?$count=true&$filter=SubmissionDate gt 2019-01-
17T00:00:00.000Z and SubmissionDate lt 2019-01-
18T00:00:00.000Z&$select=Id,OrderSize,SubmissionDate&$expand=Products($select=Name),Users
($select=Username)
```

i--ii Response (JSON format)

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Orders/$entity",
  "@odata.count": 1,
  "value":
  [
  {
    "@odata.context": "$metadata#OData.CSC.Order",
    "Id": "2b17b57d-fff4-4645-b539-91f305c27c43",
    "OrderSize": 4737286945,
    "SubmissionDate": "2019-01-17T13:55:12.390Z",
    "Products":
    [
    {
    "@odata.mediaContentType": "application/octet-stream",
    "Name": "S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6"
    }
    ]
    "Users":
    [
    {
    "@odata.mediaContentType": "application/octet-stream",
    "Username": "jroberts12"
    }
    ]
  }
  ]
}
```

iii Orders in status 'completed' per day and time of completion

iii-i Request

```
https://<service-root-uri>/odata/v1/Orders?$count=true&$filter=CompletedDate gt 2019-01-
17T00:00:00.000Z and CompletedDate lt 2019-01-18T00:00:00.000Z
&$select=Id,SubmissionDate,CompletedDate$expand=Products($select=Name),Users($select=User
name)
```

iii-ii Response (JSON format)

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Orders/$entity",
  "@odata.count": 1,
  "value":
  [
  {
    "@odata.context": "$metadata#OData.CSC.Order",
    "Id": "2b17b57d-f ff4-4645-b539-91f305c27c12",
    "SubmissionDate": "2019-01-17T13:55:12.390Z",
    "CompletedDate": "2019-01-17T15:02:22.561Z",
    "Products":
    [
    {
    "@odata.mediaContentType": "application/octet-stream",
    "Name": "S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6"
    }
    ]
    "Users":
    [
    {
    "@odata.mediaContentType": "application/octet-stream",
    "Username": "jroberts12"
    }
    ]
  }
  ]
}
```

iv Orders Pending for over 1 Day

Checks for all Orders which have a Submission date prior to the start of the reporting query day and a status still *in_progress*.

iv-i Request

```
https://<service-root-uri>/odata/v1/Orders?$count=true&$filter=SubmissionDate lt 2019-01-
17T00:00:00.000Z and Status eq
OData.CSC.Order.JobStatus'in_progress'&$select=Id,SubmissionDate
$expand=Products($select=Name),Users($select=Username)
```

iv-ii Response

```
HTTP/1.1 200 OK
{
  "@odata.context": "$metadata#Orders/$entity",
  "@odata.count": 1,
  "value":
  [
  {
    "@odata.context": "$metadata#OData.CSC.Order",
    "Id": "2b17b57d-fff4-4645-b539-91f305c27c99",
    "OrderSize": 4737286945,
    "SubmissionDate": "2019-01-15T22:25:33.491Z",
    "Products":
    [
    {
    "@odata.mediaContentType": "application/octet-stream",
    "Name": "S1B_IW_SLC__1SDV_20161224T235308_20161224T235335_003545_006104_0DD6"
    }
    ]
    "Users":
    [
    {
    "@odata.mediaContentType": "application/octet-stream",
    "Username": "jroberts12"
    }
    ]
  }
  ]
}
```

Download Reporting

Due to the flexibility inherent with the download via byte ranges the download reporting may be managed through the summary reporting outlined below.

3.9.2 LTA Metrics

The high-level statistics listed below are those which shall be made available by the LTA towards a dashboard, for example, summarizing the current status of the system and its operations. Since such statistics would be relatively intensive to calculate on demand it is assumed that they will be pre-calculated on a routine basis (e.g. daily, hourly, every 10 minutes) and cached, rather than dynamically retrieved via standard API queries. The following OData Metrics Entity provides the metrics at a reporting frequency outlined in Table 11.

Metrics Property	Type	Description
Name	String	Name of metric according to a standard naming convention
Timestamp	DateTimeOffset	Date/time of metric reporting. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ
MetricType	Enum	Gauge or Counter
Gauge	String	Value of Gauge at reporting timestamp (mandatory if MetricType=Gauge)
Counter	Int64	Value of Counter at reporting timestamp (mandatory if MetricType=Counter)

Metric	Description	Type	Frequency
Archived.<productType>.<platformShortName>.<platformSerialIdentifier>.size	Cumulative volume of <productType> of mission <platformShortName>.<platformSerialIdentifier> in archive in Bytes	Counter	10 minute update
Archived.<productType>.<platformShortName>.<platformSerialIdentifier>.count	Cumulative number of <productType> of mission <platformShortName>.<platformSerialIdentifier> in archive	Counter	10 minute update
Archived.<platformShortName>.<platformSerialIdentifier>.size	Total cumulative volume of data in archive for <platformShortName> <platformSerialIdentifier>.	Counter	Hourly update
Archived.<platformShortName>.<platformSerialIdentifier>.count	Total cumulative number of products in archive for <platformShortName> <platformSerialIdentifier>.	Counter	Hourly update
Retrieved.<productType>.<platformShortName>.<platformSerialIdentifier>.size	Cumulative volume of <productType> of mission <platformShortName>.<platformSerialIdentifier> ordered and delivered to archive interface point in Bytes	Counter	10 minute update
Retrieved.<productType>.<platformShortName>.<platformSerialIdentifier>.completed	Cumulative number of <productType> of mission <platformShortName>.<platformSerialIdentifier> ordered and delivered to archive interface point	Counter	10 minute update
Retrieved.<productType>.<platformShortName>.<platformSerialIdentifier>.failed	Cumulative number of <productType> of mission <platformShortName>.<platformSerialIdentifier> ordered and failed to be delivered to archive interface point	Counter	10 minute update
Download.<productType>.<platformShortName>.<platformSerialIdentifier>.<ServiceAlias>.completed	Cumulative number of <productType> of mission <platformShortName>.<platformSerialIdentifier> downloads completed (by <ServiceAlias>)	Counter	10 minute update
Download.<productType>.<platformShortName>.<platformSerialIdentifier>.<ServiceAlias>.failed	Cumulative number of <productType> of mission <platformShortName>.<platformSerialIdentifier> downloads failed (by <ServiceAlias>)	Counter	10 minute update
Download.<productType>.<platformShortName>.<platformSerialIdentifier>.<ServivceAlias>.size	Cumulative size of <productType> of mission <platformShortName>.<platformSerialIdentifier> downloaded (by <ServiceAlias>) in Bytes	Counter	10 minute update
OriginToPublication.Daily.min.time.<productType>.<platformShortName>.<platformSerialIdentifier>	Daily minimum time difference in seconds between PRIP PublicationDate and LTA PublicationDate of <productType> of mission <platformShortName>.<platformSerialIdentifier> (sliding window of 24 hours)	Gauge	10 minute update
OriginToPublication.Daily.max.time.<productType>.<platformShortName>.<platformSerialIdentifier>	Daily maximum time difference in seconds between PRIP PublicationDate and LTA PublicationDate of <productType> of mission	Gauge	10 minute update

	<platformShortName>.<platformSerialIdentifier> (sliding window of 24 hours)		
OriginToPublication.Daily.avg.time.<productType>.<platformShortName>.<platformSerialIdentifier>	Daily average time difference in seconds between PRIP PublicationDate and LTA PublicationDate of <productType> of mission <platformShortName>.<platformSerialIdentifier> (sliding window of 24 hours)	Gauge	10 minute update
OriginToPublication.Monthly.min.time.<productType>.<platformShortName>.<platformSerialIdentifier>	Monthly minimum time difference in seconds between PRIP PublicationDate and LTA PublicationDate of <productType> of mission <platformShortName>.<platformSerialIdentifier> (sliding window of last month)	Gauge	Hourly update
OriginToPublication.Monthly.max.time.<productType>.<platformShortName>.<platformSerialIdentifier>	Monthly maximum time difference in seconds between PRIP PublicationDate and LTA PublicationDate of <productType> of mission <platformShortName>.<platformSerialIdentifier> (sliding window of last month)	Gauge	Hourly update
OriginToPublication.Monthly.avg.time.<productType>.<platformShortName>.<platformSerialIdentifier>	Monthly average time difference in seconds between PRIP PublicationDate and LTA PublicationDate of <productType> of mission <platformShortName>.<platformSerialIdentifier> (sliding window of last month)	Gauge	Hourly update
SubmissionToCompletion.Daily.min.time	Minimum time from order submission to delivery on archive interface point (sliding window of 24 hours)	Gauge	10 minute update
SubmissionToCompletion.Daily.max.time	Maximum time from order submission to delivery on archive interface point (sliding window of 24 hours)	Gauge	10 minute update
SubmissionToCompletion.Daily.avg.time	Average time from order submission to delivery on archive interface point (sliding window of 24 hours)	Gauge	10 minute update
SubmissionToCompletion.Monthly.min.time	Minimum time from order submission to delivery on archive interface point (sliding window of last month)	Gauge	Hourly update
SubmissionToCompletion.Monthly.max.time	Maximum time from order submission to delivery on archive interface point (sliding window of last month)	Gauge	Hourly update
SubmissionToCompletion.Monthly.avg.time	Average time from order submission to delivery on archive interface point (sliding window of last month)	Gauge	Hourly update
Service.<KPI>.value	KPI value	Gauge	

Table 11: LTA Summary Reporting Statistics

N.B. Service Alias is based on the user information.

3.10 LTA Event Recording

The LTA uses the Event Entity for recording information on events which have taken place or are planned to take place in the future. The precise scope will include, for example, planned unavailabilities of the LTA due to service or infrastructure maintenance and unplanned availabilities due to e.g. network issues.

The Event entity model and properties are described in the subsection below.

3.10.1 Event Entity Model

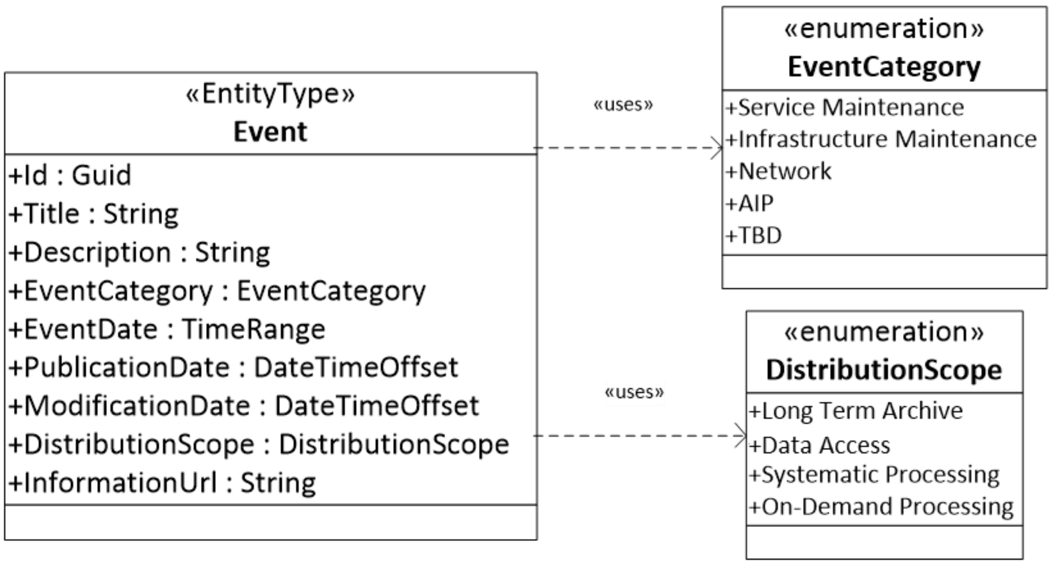


Figure 11: Event Entity Model

Event Properties	Type	M	Description	Example
Id	Guid	Y	It is a universally unique identifier (UUID). The Id is a local identifier for the Event instance within the LTA, assigned upon creation of the Event.	`2b17b57d-fff4-4645-b539-91f305c27c69`
Title	String	Y	Title of the event (concise summary)	`Network Incident on the LTA Infrastructure (27 - 29 May 2019)`
Description	String	Y	Description of the Event	`The network incident that occurred on the LTA Service on 27 May 2019, causing a degradation of the product staging and retrieval services, was resolved on 29 May 2019, at 02:00 UTC.`

EventCategory	EventCategory Enumeration	Y	Chosen from a predefined list of categories (single selection available): - *Service Maintenance* - *Infrastructure Maintenance* - *Network* - *AIP* - *TBD*	`Service Maintenance`

EventDate	Time Range	Y	The Event range period. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`"EventDate":`
				`{`
				`"Start":"2019-05-29T10:34:51.922Z",`
				`"End":"2019-05-29T12:35:18.872Z"`
				`}`

PublicationDate	DateTimeOffset	Y	Publication date and time of the Event entity. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-17T14:46:03.788Z`
ModificationDate	DateTimeOffset	N	Last modification date and time of the Event entity. Time is in UTC in the format YYYY-MM-DDThh:mm:ss.sssZ	`2018-01-19T18:00:00.000Z`
DistributionScope	DistributionScope Enumeration	N	This field is relevant only for events concerning the Service availability. Chosen from a predefined list of Data Access entities (multiple selection available): - *Long Term Archive* - *Data Access* - *Systematic Processing* - *On-Demand Processing*	`Long Term Archive`
InformationUrl	String	N	External link to the information within the anomaly tracking	`Any URI`

Table 13: Event Entity Description

4 ANNEX 1: ODATA METADATA DESCRIPTION

```
<?xml version="1.0"?>
<edmx:Edmx Version="4.0">
  <edmx:DataServices>
    <Schema Namespace="OData.CSC">
      <EnumType Name="SubscriptionEvent" IsFlags="false" UnderlyingType="Edm.Int32">
        <Member Name="created" Value="0"/>
        <Member Name="deleted" Value="1"/>
      </EnumType>
      <EnumType Name="JobStatus" IsFlags="false" UnderlyingType="Edm.Int32">
        <Member Name="created" Value="0"/>
        <Member Name="queued" Value="1"/>
        <Member Name="in_progress" Value="2"/>
        <Member Name="completed" Value="3"/>
        <Member Name="failed" Value="4"/>
        <Member Name="cancelled" Value="5"/>
      </EnumType>
      <EnumType Name="EventCategory" IsFlags="false" UnderlyingType="Edm.Int32">
        <Member Name="Service Maintenance" Value="0"/>
        <Member Name="Infrastructure Maintenance" Value="1"/>
        <Member Name="Network" Value="2"/>
        <Member Name="AIP" Value="3"/>
      </EnumType>
      <EnumType Name="DistributionScope" IsFlags="false" UnderlyingType="Edm.Int32">
        <Member Name="Long Term Archive" Value="0"/>
        <Member Name="Data Access" Value="1"/>
        <Member Name="Systematic Processing" Value="2"/>
        <Member Name="On-Demand Processing" Value="3"/>
      </EnumType>
      <EnumType Name="SubscriptionStatus" IsFlags="false" UnderlyingType="Edm.Int32">
        <Member Name="running" Value="0"/>
        <Member Name="paused" Value="1"/>
        <Member Name="cancelled" Value="2"/>
      </EnumType>
      <EnumType Name="EnumType" IsFlags="false" UnderlyingType="Edm.Int32">
        <Member Name="Gauge" Value="0"/>
        <Member Name="Counter" Value="1"/>
      </EnumType>
      <Function Name="Intersects" IsBound="true">
        <Parameter Name="product" Type="OData.CSC.Product"/>
        <Parameter Name="area" Type="Edm.Geography"/>
        <ReturnType Type="Collection(OData.CSC.Product)"/>
     </Function>
     <EntityType Name="Bulk">
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.Guid"/>
        <Property Name="Status" Type="OData.CSC.JobStatus"/>
        <Property Name="StatusMessage" Type="Edm.String"/>
        <Property Name="FilterParam" Type="Edm.String"/>
        <Property Name="OrderbyParam" Type="Edm.String"/>
        <Property Name="BatchsizeProducts" Type="Edm.Int64"/>
        <Property Name="BatchsizeVolume" Type="Edm.Int64"/>
        <Property Name="SubmissionDate" Type="Edm.DateTimeOffset" Precision="3"/>
        <Property Name="CompletedDate" Type="Edm.DateTimeOffset" Precision="3"/>
        <NavigationProperty Name="BatchOrders" Type="Collection(OData.CSC.BatchOrder)"/>
        <NavigationProperty Name="Products" Type="Collection(OData.CSC.Product)"/>
      </EntityType>
      <EntityType Name="Event">
```

```
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.Guid"/>
        <Property Name="Title" Type="Edm.String"/>
        <Property Name="Description" Type="Edm.String"/>
        <Property Name="EventCategory" Type="OData.CSC.EventCategory"/>
        <Property Name="DistributionScope" Type="OData.CSC.DistributionScope"/>
        <Property Name="EventDate" Type="OData.CSC.TimeRange"/>
        <Property Name="PublicationDate" Type="Edm.DateTimeOffset" Precision="3"/>
        <Property Name="ModificationDate" Type="Edm.DateTimeOffset" Precision="3"/>
        <Property Name="InformationUrl" Type="Edm.String"/>
        <NavigationProperty Name="Products" Type="Collection(OData.CSC.Product)"/>
      </EntityType>
      <EntityType Name="StringAttribute" BaseType="OData.CSC.Attribute">
        <Property Name="Value" Type="Edm.String"/>
      </EntityType>
      <EntityType Name="IntegerAttribute" BaseType="OData.CSC.Attribute">
        <Property Name="Value" Type="Edm.Int64"/>
      </EntityType>
      <EntityType Name="DateTimeOffsetAttribute" BaseType="OData.CSC.Attribute">
        <Property Name="Value" Type="Edm.DateTimeOffset"/>
      </EntityType>
      <EntityType Name="DoubleAttribute" BaseType="OData.CSC.Attribute">
        <Property Name="Value" Type="Edm.Double"/>
      </EntityType>
      </EntityType>
      <EntityType Name="BooleanAttribute" BaseType="OData.CSC.Attribute">
        <Property Name="Value" Type="Edm.Boolean"/>
      </EntityType>
      <EntityType Name="Order">
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.Guid"/>
        <Property Name="Status" Type="OData.CSC.JobStatus"/>
        <Property Name="StatusMessage" Type="Edm.String"/>
        <Property Name="OrderSize" Type="Edm.Int64"/>
        <Property Name="SubmissionDate" Type="Edm.DateTimeOffset"/>
        <Property Name="EstimatedDate" Type="Edm.DateTimeOffset"/>
        <Property Name="CompletedDate" Type="Edm.DateTimeOffset"/>
        <Property Name="EvictionDate" Type="Edm.DateTimeOffset"/>
        <Property Name="Priority" Type="Edm.Int64"/>
        <NavigationProperty Name="Product" Type="OData.CSC.Product"/>
      </EntityType>
      <EntityType Name="Subscription">
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.Guid"/>
        <Property Name="Status" Type="OData.CSC.SubscriptionStatus"/>
        <Property Name="SubscriptionEvent" Type="OData.CSC.SubscriptionEvent"/>
        <Property Name="FilterParam" Type="Edm.String"/>
        <Property Name="Priority" Type="Edm.Int64"/>
        <Property Name="SubmissionDate" Type="Edm.DateTimeOffset" Precision="3"/>
        <Property Name="LastNotificationDate" Type="Edm.DateTimeOffset" Precision="3"/>
        <Property Name="StageOrder" Type="Edm.Boolean"/>
      </EntityType>
      <EntityType Name="Product" HasStream="true">
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.Guid"/>
```

```
    <Property Name="Name" Type="Edm.String"/>
    <Property Name="ContentType" Type="Edm.String"/>
    <Property Name="ContentLength" Type="Edm.Int64"/>
    <Property Name="OriginDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <Property Name="PublicationDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <Property Name="ModificationDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <Property Name="Online" Type="Edm.Boolean"/>
    <Property Name="EvictionDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <Property Name="Checksum" Type="Collection(OData.CSC.Checksum)"/>
    <Property Name="ContentDate" Type="OData.CSC.TimeRange"/>
    <Property Name="Footprint" Type="Edm.Geography"/>
    <Property Name="GeoFootprint" Type="Edm.Geography"/>
    <NavigationProperty Name="Attributes" Type="Collection(OData.CSC.Attribute)"/>
  </EntityType>
  <EntityType Name="BatchOrder">
    <Key>
      <PropertyRef Name="Id"/>
    </Key>
    <Property Name="Id" Type="Edm.Guid"/>
    <Property Name="Status" Type="OData.CSC.JobStatus"/>
    <Property Name="StatusMessage" Type="Edm.String"/>
    <Property Name="OrderSize" Type="Edm.Int64"/>
    <Property Name="Priority" Type="Edm.Int64"/>
    <Property Name="SubmissionDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <Property Name="CompletedDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <Property Name="EstimatedDate" Type="Edm.DateTimeOffset" Precision="3"/>
    <NavigationProperty Name="Products" Type="Collection(OData.CSC.Product)"/>
  </EntityType>
  <EntityType Name="Attribute" Abstract="true">
    <Key>
      <PropertyRef Name="Name"/>
    </Key>
    <Property Name="Name" Type="Edm.String"/>
    <Property Name="ValueType" Type="Edm.String"/>
  </EntityType>
  <EntityType Name="Metric">
     <Key>
       <PropertyRef Name="Name"/>
     </Key>
     <Property Name="Name" Type="Edm.String"/>
     <Property Name="Timestamp" Type="Edm.DateTimeOffset" Precision="3"/>
     <Property Name="MetricType" Type="OData.CSC.EnumType"/>
     <Property Name="Gauge" Type="Edm.String"/>
     <Property Name="Counter" Type="Edm.Int64"/>
  </EntityType>
  <ComplexType Name="TimeRange">
    <Property Name="Start" Type="Edm.DateTimeOffset" Nullable="false" Precision="6"/>
    <Property Name="End" Type="Edm.DateTimeOffset" Nullable="false" Precision="6"/>
  </ComplexType>
  <ComplexType Name="Checksum">
    <Property Name="Algorithm" Type="Edm.String" Nullable="false"/>
    <Property Name="Value" Type="Edm.String" Nullable="false"/>
    <Property Name="ChecksumDate" Type="Edm.DateTimeOffset"/>
  </ComplexType>
  <ComplexType Name="Property">
    <Property Name="Name" Type="Edm.String" Nullable="false"/>
    <Property Name="Value" Type="Edm.String" Nullable="false"/>
  </ComplexType>
  <Action Name="Pause" IsBound="true">
    <Parameter Name="Subscription" Type="OData.CSC.Subscription" Nullable="false"/>
  </Action>
  <Action Name="Order" IsBound="true">
    <Parameter Name="Product" Type="OData.CSC.Product" Nullable="false"/>
```

```
        <Parameter Name="Priority" Type="Edm.Int32"/>
        <Parameter Name="NotificationEndpoint" Type="Edm.String"/>
        <Parameter Name="NotificationEpUsername" Type="Edm.String"/>
        <Parameter Name="NotificationEpPassword" Type="Edm.String"/>
        <ReturnType Type="OData.CSC.Order"/>
      </Action>
      <Action Name="BatchOrder" IsBound="true">
        <Parameter Name="BatchOrder" Type="OData.CSC.BatchOrder" Nullable="false"/>
        <Parameter Name="Priority" Type="Edm.Int32"/>
        <ReturnType Type="OData.CSC.BatchOrder"/>
      </Action>
      <Action Name="FilterList" IsBound="true">
        <Parameter Name="FilterProducts" Type="Collection(OData.CSC.Product)"
Nullable="false"/>
        <ReturnType Type="Collection(OData.CSC.Product)"/>
      </Action>
      <Action Name="Resume" IsBound="true">
        <Parameter Name="Subscription" Type="OData.CSC.Subscription" Nullable="false"/>
        <ReturnType Type="OData.CSC.Subscription"/>
      </Action>
      <Action Name="Cancel" IsBound="true">
        <Parameter Name="Order" Type="OData.CSC.Order" Nullable="false"/>
        <ReturnType Type="OData.CSC.Order"/>
      </Action>
      <Action Name="Cancel" IsBound="true">
        <Parameter Name="Bulk" Type="OData.CSC.Bulk" Nullable="false"/>
        <ReturnType Type="OData.CSC.Bulk"/>
      </Action>
      <Action Name="Cancel" IsBound="true">
        <Parameter Name="BatchOrder" Type="OData.CSC.BatchOrder" Nullable="false"/>
        <ReturnType Type="OData.CSC.BatchOrder"/>
      </Action>
      <Action Name="Cancel" IsBound="true">
        <Parameter Name="Subscription" Type="OData.CSC.Subscription" Nullable="false"/>
        <ReturnType Type="OData.CSC.Subscription"/>
      </Action>
      <EntityContainer Name="Container">
        <EntitySet Name="Orders" EntityType="OData.CSC.Order">
          <NavigationPropertyBinding Path="Product" Target="Products"/>
          <NavigationPropertyBinding Path="User" Target="Users"/>
        </EntitySet>
        <EntitySet Name="Products" EntityType="OData.CSC.Product">
          <NavigationPropertyBinding Path="Attributes" Target="Attributes"/>
        </EntitySet>
        <EntitySet Name="Subscriptions" EntityType="OData.CSC.Subscription">
          <NavigationPropertyBinding Path="User" Target="Users"/>
        </EntitySet>
        <EntitySet Name="Attributes" EntityType="OData.CSC.Attribute"
IncludeInServiceDocument="false"/>
        <EntitySet Name="Events" EntityType="OData.CSC.Event"/>
        <EntitySet Name="Bulks" EntityType="OData.CSC.Bulk">
          <NavigationPropertyBinding Path="BatchOrders" Target="BatchOrders"/>
          <NavigationPropertyBinding Path="Products" Target="Products"/>
          <NavigationPropertyBinding Path="User" Target="Users"/>
        </EntitySet>
        <EntitySet Name="BatchOrders" EntityType="OData.CSC.BatchOrder">
          <NavigationPropertyBinding Path="Products" Target="Products"/>
          <NavigationPropertyBinding Path="User" Target="Users"/>
        </EntitySet>
        <EntitySet Name="StringAttributes" EntityType="OData.CSC.StringAttribute"
IncludeInServiceDocument="false"/>
        <EntitySet Name="IntegerAttributes" EntityType="OData.CSC.IntegerAttribute"
IncludeInServiceDocument="false"/>
```

```
        <EntitySet Name="DoubleAttributes" EntityType="OData.CSC.DoubleAttribute"
IncludeInServiceDocument="false"/>
        <EntitySet Name="BooleanAttributes" EntityType="OData.CSC.BooleanAttribute"
IncludeInServiceDocument="false"/>
        <EntitySet Name="DateTimeOffsetAttributes"
EntityType="OData.CSC.DateTimeOffsetAttribute" IncludeInServiceDocument="false"/>
        <EntitySet Name="Metrics" EntityType="OData.CSC.Metric"/>
      </EntityContainer>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

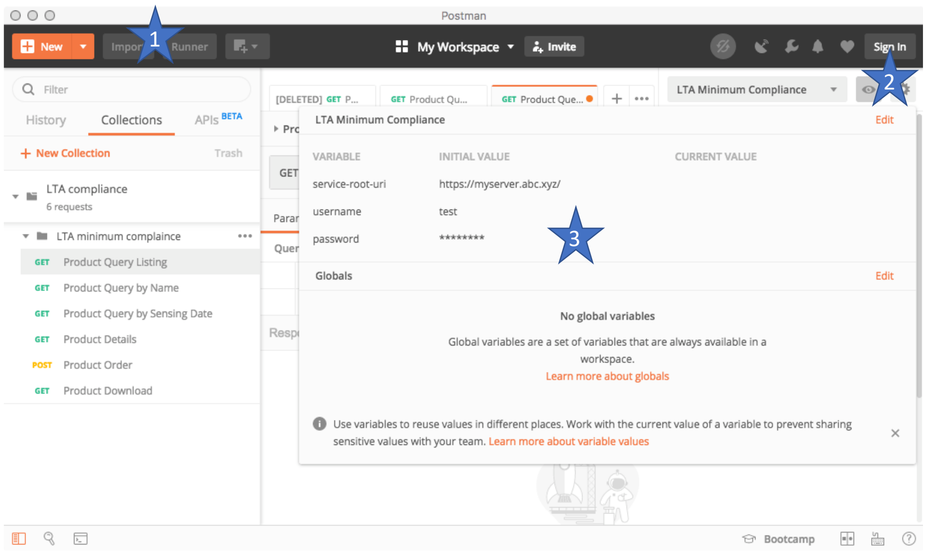
5 ANNEX 2: COMPLIANCE TEST SUITES[7]

The present Annex provides details of the compliance test suites for the ICD. The test suites are provided as a set of OData requests to be sent to the AIP.

The current implementation of the test suites is designed to test the minimum and extended capabilities of the AIP, and are implemented as Postman (https://www.getpostman.com/) Collections JSON files (format v2.1). These are provided as accompanying external files: LTA-ICD_MinimumCompliance.json and LTA-ICD_ExtendedCompliance.json.

5.1 LTA Minimum Compliance Test Suite Instructions

1) Import the Postman Collection
2) Add an LTA Minimum Compliance environment
3) Add service-root-uri, username and password variables according to AIP configuration, as shown below.



LTA Minimum Compliance

VARIABLE	INITIAL VALUE	CURRENT VALUE
service-root-uri	https://myserver.abc.xyz/	
username	test	
password	********	

Globals

No global variables

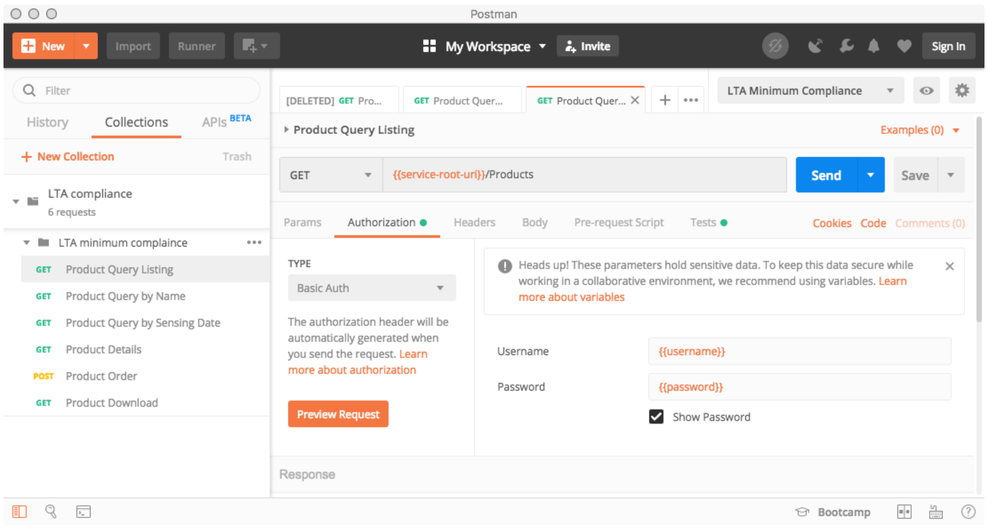
Global variables are a set of variables that are always available in a workspace.
Learn more about globals

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

5.1.1 *Product Listing Query*

Inside the LTA minimum compliance folder the first test query issues a basic product listing to the AIP. As with all tests Basic Authentication is assumed, and the username/password are set according to the variables established in the environment. The test is initiated by sending the request to the AIP and success is indicated in the Test tab as pass or fail.

[7] The AIP requests and tests presented in this chapter are for information and illustration purposes only. The final test suites may vary and are to be provided as external files.


GET {{service-root-uri}}/Products

TYPE Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

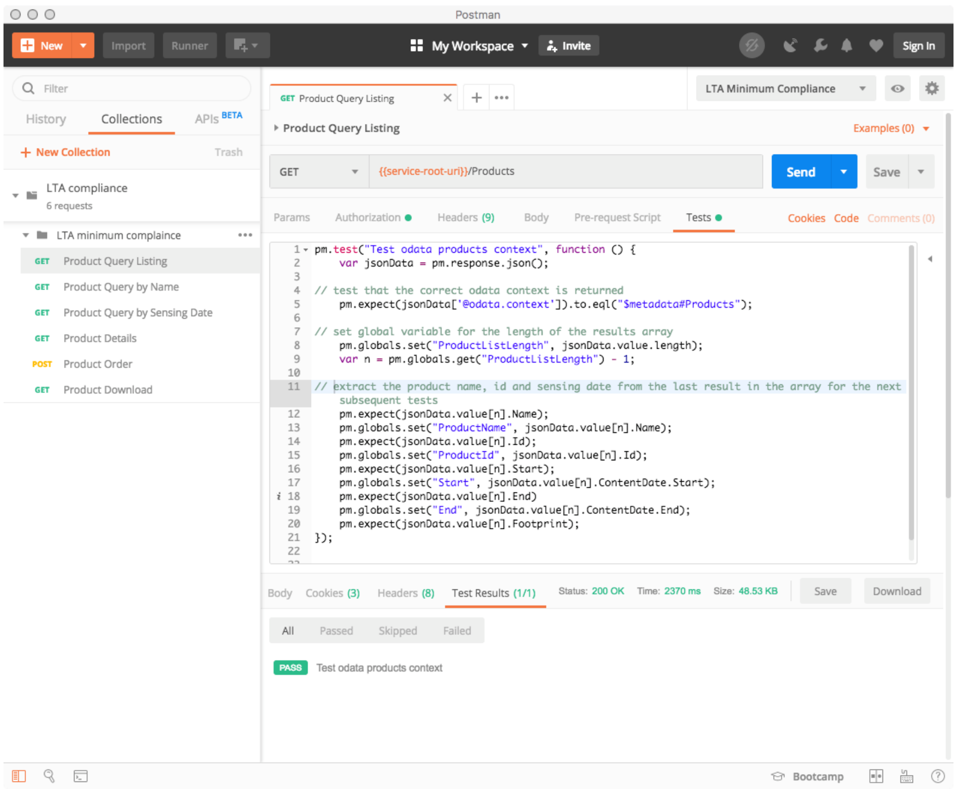
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username {{username}}

Password {{password}}

Show Password ✓

The tests performed on the Product Listing Query are formulated to verify the OData context of the response, to verify that minimum set of metadata have been catalogued for the archived product and to set the variables for one of the products returned to be used in subsequent tests.


GET {{service-root-uri}}/Products

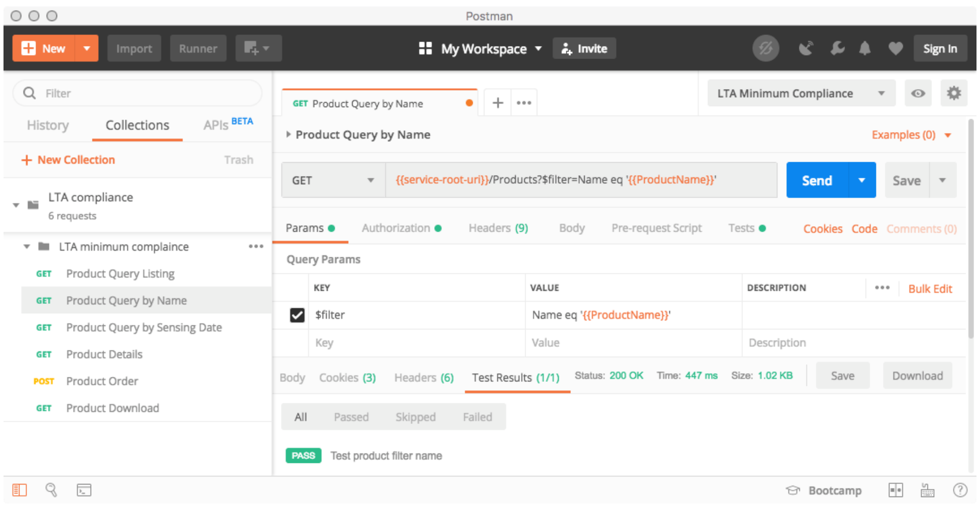
```
pm.test("Test odata products context", function () {
    var jsonData = pm.response.json();

// test that the correct odata context is returned
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#Products");

// set global variable for the length of the results array
    pm.globals.set("ProductListLength", jsonData.value.length);
    var n = pm.globals.get("ProductListLength") - 1;

// extract the product name, id and sensing date from the last result in the array for the next
    subsequent tests
    pm.expect(jsonData.value[n].Name);
    pm.globals.set("ProductName", jsonData.value[n].Name);
    pm.expect(jsonData.value[n].Id);
    pm.globals.set("ProductId", jsonData.value[n].Id);
    pm.expect(jsonData.value[n].Start);
    pm.globals.set("Start", jsonData.value[n].ContentDate.Start);
    pm.expect(jsonData.value[n].End)
    pm.globals.set("End", jsonData.value[n].ContentDate.End);
    pm.expect(jsonData.value[n].Footprint);
});
```

Test Results (1/1) Status: 200 OK Time: 2370 ms Size: 48.53 KB

PASS Test odata products context

5.1.2 *Product Query By Name*

The second test query introduces the filter syntax, and queries for a product by name according to the variables set in the previous test.



```
GET {{service-root-uri}}/Products?$filter=Name eq '{{ProductName}}'
```

	KEY	VALUE	DESCRIPTION
✓	$filter	Name eq '{{ProductName}}'	
	Key	Value	Description

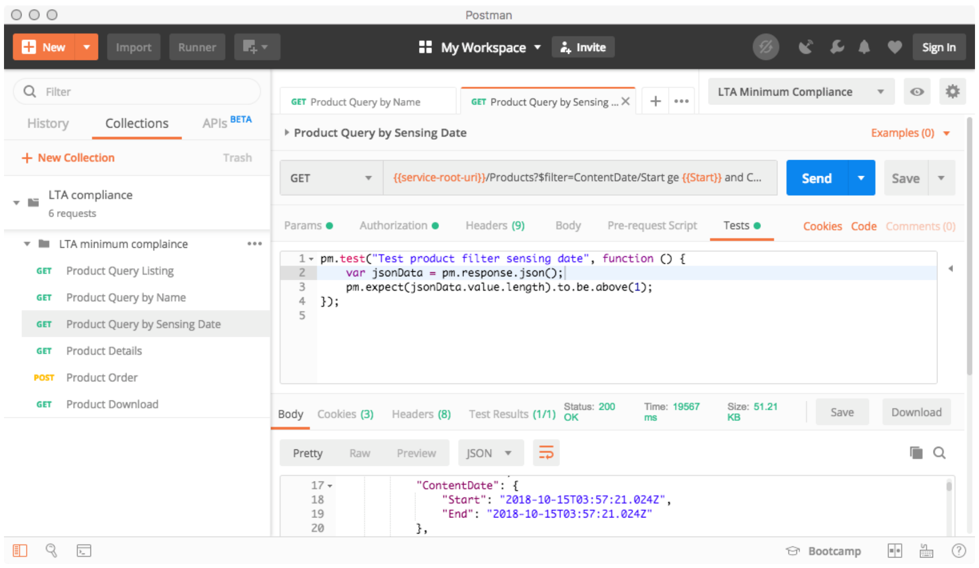
Status: 200 OK Time: 447 ms Size: 1.02 KB

PASS Test product filter name

The test verifies that a single result is returned with the correct name.

5.1.3 *Product Query By Sensing Date*

The third test introduces the query by sensing date.



```
GET {{service-root-uri}}/Products?$filter=ContentDate/Start ge {{Start}} and C...
```

```
pm.test("Test product filter sensing date", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value.length).to.be.above(1);
});
```

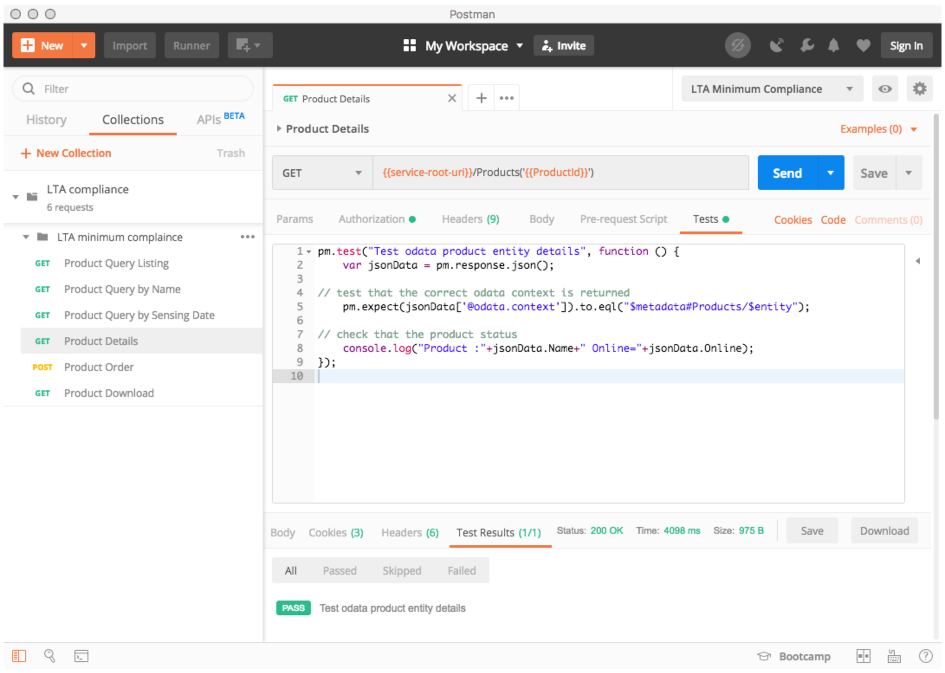
Status: 200 OK Time: 19567 ms Size: 51.21 KB

```
        "ContentDate": {
            "Start": "2018-10-15T03:57:21.024Z",
            "End": "2018-10-15T03:57:21.024Z"
        },
```

Whereas the test checks that at least one Product is returned, manual inspection should be used to confirm that results outside the time period submitted in the query are not returned.

5.1.4 *Product Details*

The fourth test checks that the product can be referenced via its ID and outputs the current status offline/online to the console



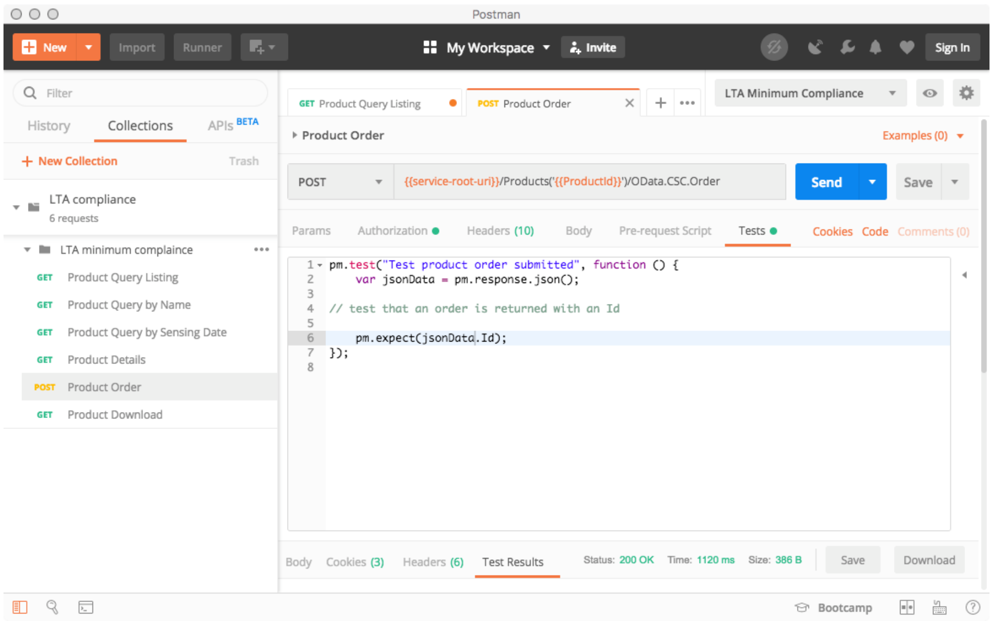
```
GET {{service-root-uri}}/Products('{{ProductId}}')
```

```
pm.test("Test odata product entity details", function () {
    var jsonData = pm.response.json();

// test that the correct odata context is returned
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#Products/$entity");

// check that the product status
    console.log("Product :"+jsonData.Name+" Online="+jsonData.Online);
});
```

```
PASS Test odata product entity details
```

5.1.5 *Product Order*

Assuming the product is offline (Online=false) the fifth test orders the product, triggering it's delivery to the AIP for download. The test confirms that an order is made and provided an ID.



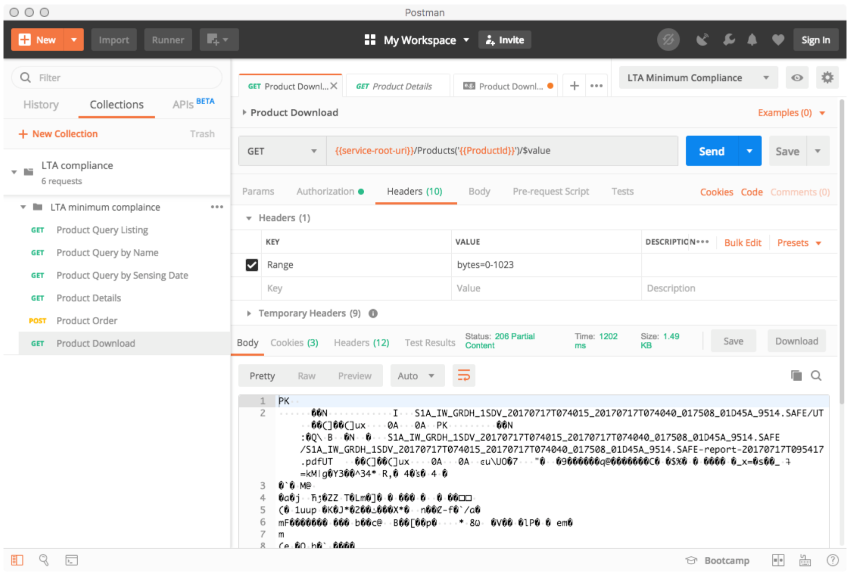
```
POST {{service-root-uri}}/Products('{{ProductId}}')/OData.CSC.Order
```

```
pm.test("Test product order submitted", function () {
    var jsonData = pm.response.json();

// test that an order is returned with an Id

    pm.expect(jsonData.Id);
});
```

The test in step 5.1.4 may be repeated to verify when the data is online (Online=true) and ready for download.

5.1.6 *Product Download*

The sixth step tests the product download. Due to limitations in managing large file download in Postman the download request is made for only the first 1024 bytes of data. Therefore a 206 Partial Content HTTP response code is expected.



GET {{service-root-uri}}/Products('{{ProductId}}')/$value

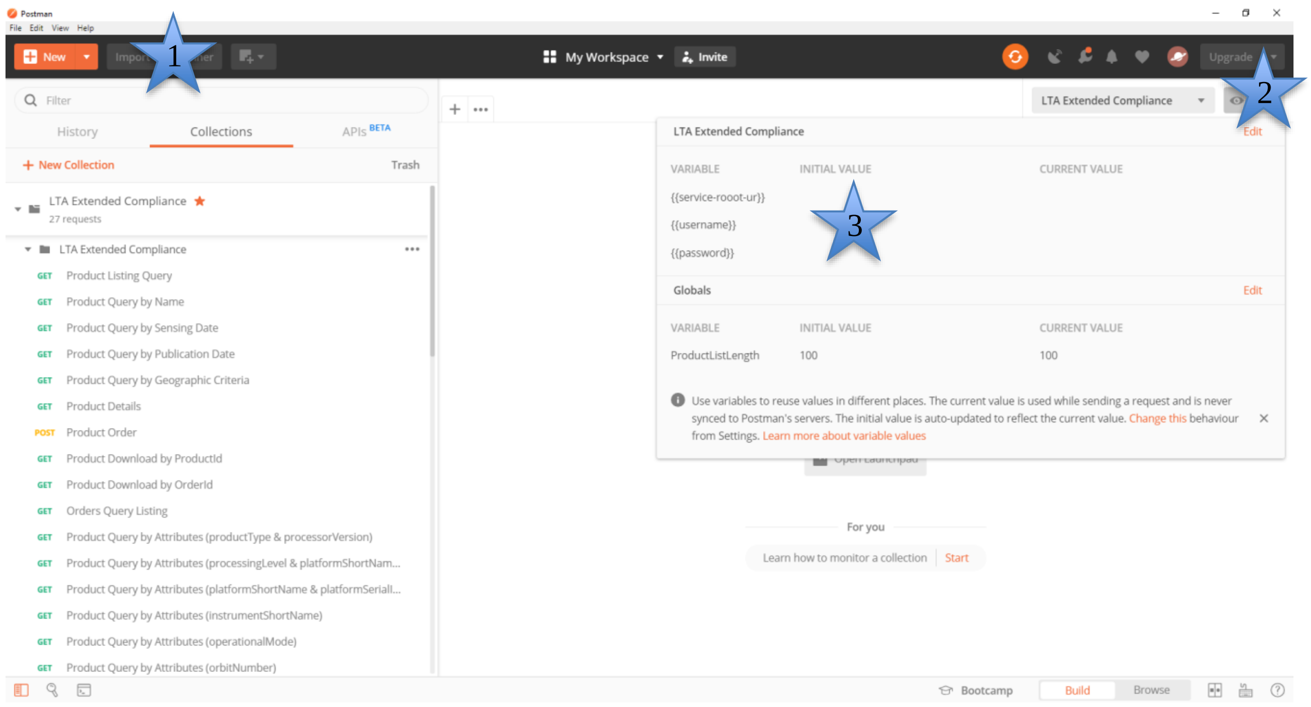
	KEY	VALUE	DESCRIPTION
✓	Range	bytes=0-1023	
	Key	Value	Description

Status: 206 Partial Content   Time: 1202 ms   Size: 1.49 KB

```
PK
      ��N            I   S1A_IW_GRDH_1SDV_20170717T074015_20170717T074040_017508_01D45A_9514.SAFE/UT
      ��(]��(]ux    0A   0A  PK         ��N
    :�Q\ B  �N  �   S1A_IW_GRDH_1SDV_20170717T074015_20170717T074040_017508_01D45A_9514.SAFE
    /S1A_IW_GRDH_1SDV_20170717T074015_20170717T074040_017508_01D45A_9514.SAFE-report-20170717T095417
    .pdfUT   ��(]��(]ux    0A   0A  cu\UO�7   "�  �9������q@�������C� �$%� � ���� �_x=�s��_ ˥
    =kM|g�Y3��^34* R,� 4�s� 4 �
�`� M@
�a�j  ħ;�ZZ T�Lm�]� � ��� �  � ��□□
(� 1uup �K�J*�2��.���X*�  n��Ȼ-f�`/a�
mF������� ��� b��c@  B��[��p�    * 8Ω  �V�� �lP� � em�
m
Ce �O h�` ����
```

5.2 LTA Extended Compliance Test Suite Instructions

1) Import the Postman Collection
2) Add an LTA Extended Compliance environment
3) Add service-root-uri, username and password variables according to AIP configuration, as shown below.



GET Product Listing Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders Query Listing
GET Product Query by Attributes (productType & processorVersion)
GET Product Query by Attributes (processingLevel & platformShortNam...
GET Product Query by Attributes (platformShortName & platformSeriall...
GET Product Query by Attributes (instrumentShortName)
GET Product Query by Attributes (operationalMode)
GET Product Query by Attributes (orbitNumber)

LTA Extended Compliance — Edit

VARIABLE	INITIAL VALUE	CURRENT VALUE
{{service-rooot-uri}}		
{{username}}		
{{password}}		

Globals — Edit

VARIABLE	INITIAL VALUE	CURRENT VALUE
ProductListLength	100	100

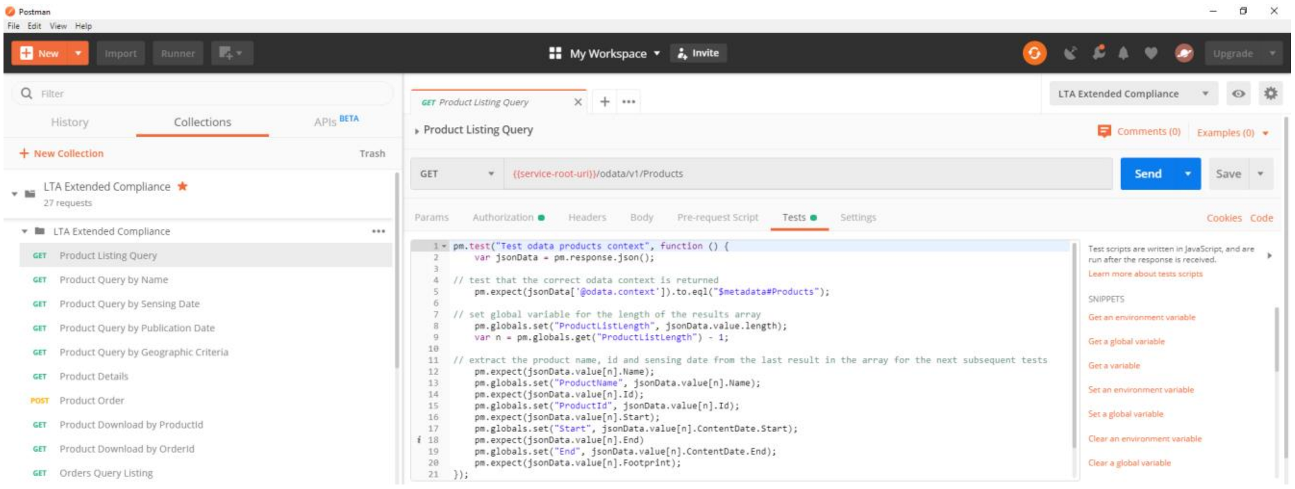
Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values

The LTA Extended Compliance Test Suite, includes the Minimum Compliance Test Suite and comes with additional queries for testing the AIP.

5.2.1 *Product Listing Query*

The first test query issues a basic product listing to the AIP. The test is initiated by sending the request to the AIP, using the environmental variables established, and success is indicated in the Test tab as pass or fail.

The tests performed on the Product Listing Query are formulated to verify the OData context of the response, to verify that minimum set of metadata have been catalogued for the archived product and to set the variables for one of the products returned to be used in subsequent tests.



LTA Extended Compliance
- GET Product Listing Query
- GET Product Query by Name
- GET Product Query by Sensing Date
- GET Product Query by Publication Date
- GET Product Query by Geographic Criteria
- GET Product Details
- POST Product Order
- GET Product Download by ProductId
- GET Product Download by OrderId
- GET Orders Query Listing

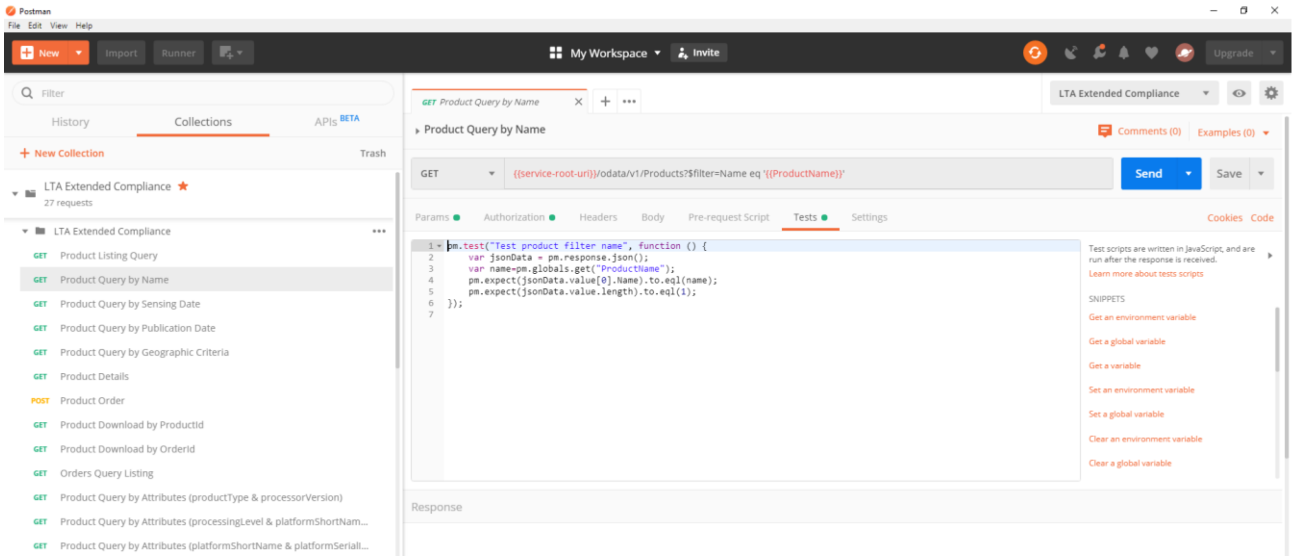
Product Listing Query

GET {{service-root-uri}}/odata/v1/Products

```
pm.test("Test odata products context", function () {
    var jsonData = pm.response.json();

// test that the correct odata context is returned
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#Products");

// set global variable for the length of the results array
    pm.globals.set("ProductListLength", jsonData.value.length);
    var n = pm.globals.get("ProductListLength") - 1;

// extract the product name, id and sensing date from the last result in the array for the next subsequent tests
    pm.expect(jsonData.value[n].Name);
    pm.globals.set("ProductName", jsonData.value[n].Name);
    pm.expect(jsonData.value[n].Id);
    pm.globals.set("ProductId", jsonData.value[n].Id);
    pm.expect(jsonData.value[n].Start);
    pm.globals.set("Start", jsonData.value[n].ContentDate.Start);
    pm.expect(jsonData.value[n].End)
    pm.globals.set("End", jsonData.value[n].ContentDate.End);
    pm.expect(jsonData.value[n].Footprint);
});
```

5.2.2 *Product Query By Name*

The second test query introduces the filter syntax, and queries for a product by name according to the variables set in the previous test.



LTA Extended Compliance
- GET Product Listing Query
- GET Product Query by Name
- GET Product Query by Sensing Date
- GET Product Query by Publication Date
- GET Product Query by Geographic Criteria
- GET Product Details
- POST Product Order
- GET Product Download by ProductId
- GET Product Download by OrderId
- GET Orders Query Listing
- GET Product Query by Attributes (productType & processorVersion)
- GET Product Query by Attributes (processingLevel & platformShortNam...
- GET Product Query by Attributes (platformShortName & platformSeriali...

Product Query by Name

GET {{service-root-uri}}/odata/v1/Products?$filter=Name eq '{{ProductName}}'

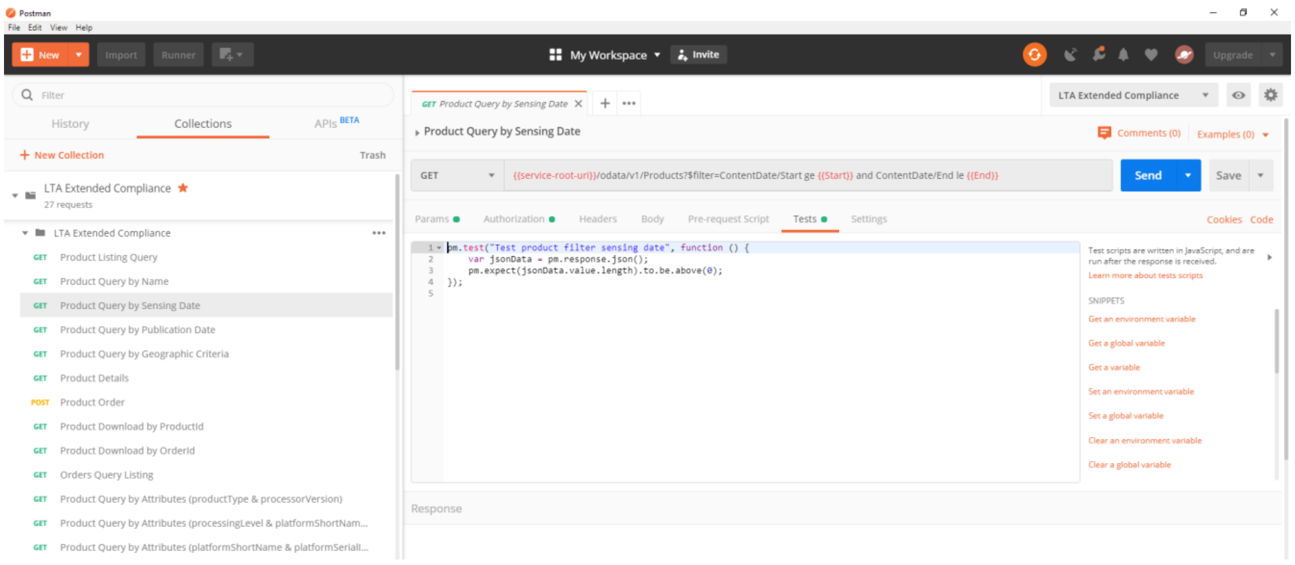
```
pm.test("Test product filter name", function () {
    var jsonData = pm.response.json();
    var name=pm.globals.get("ProductName");
    pm.expect(jsonData.value[0].Name).to.eql(name);
    pm.expect(jsonData.value.length).to.eql(1);
});
```

The test verifies that a single result is returned with the correct name.

European Space Agency
Agence spatiale européenne

5.2.3 Product Query By Sensing Date

The third test introduces the query by sensing date.



GET Product Query by Sensing Date

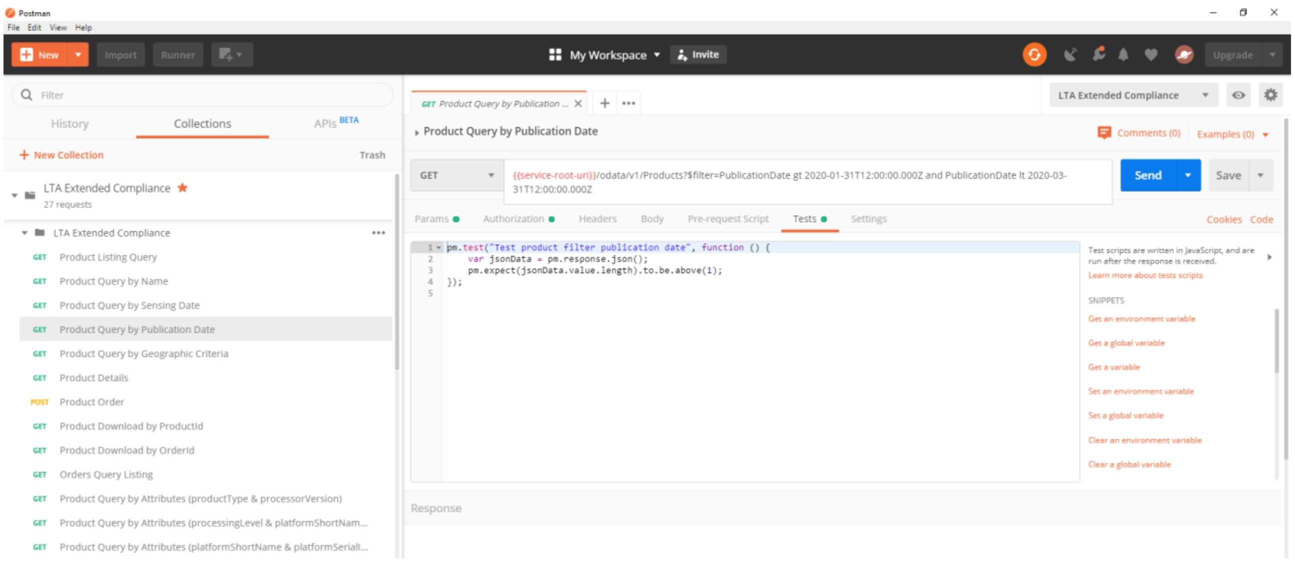
GET {{service-root-uri}}/odata/v1/Products?$filter=ContentDate/Start ge {{Start}} and ContentDate/End le {{End}}

```
pm.test("Test product filter sensing date", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value.length).to.be.above(0);
});
```

Whereas the test checks that at least one Product is returned, manual inspection should be used to confirm that results outside the time period submitted in the query are not returned.

5.2.4 Product Query By Publication Date

The fourth test concerns the query by publication date.



GET Product Query by Publication Date

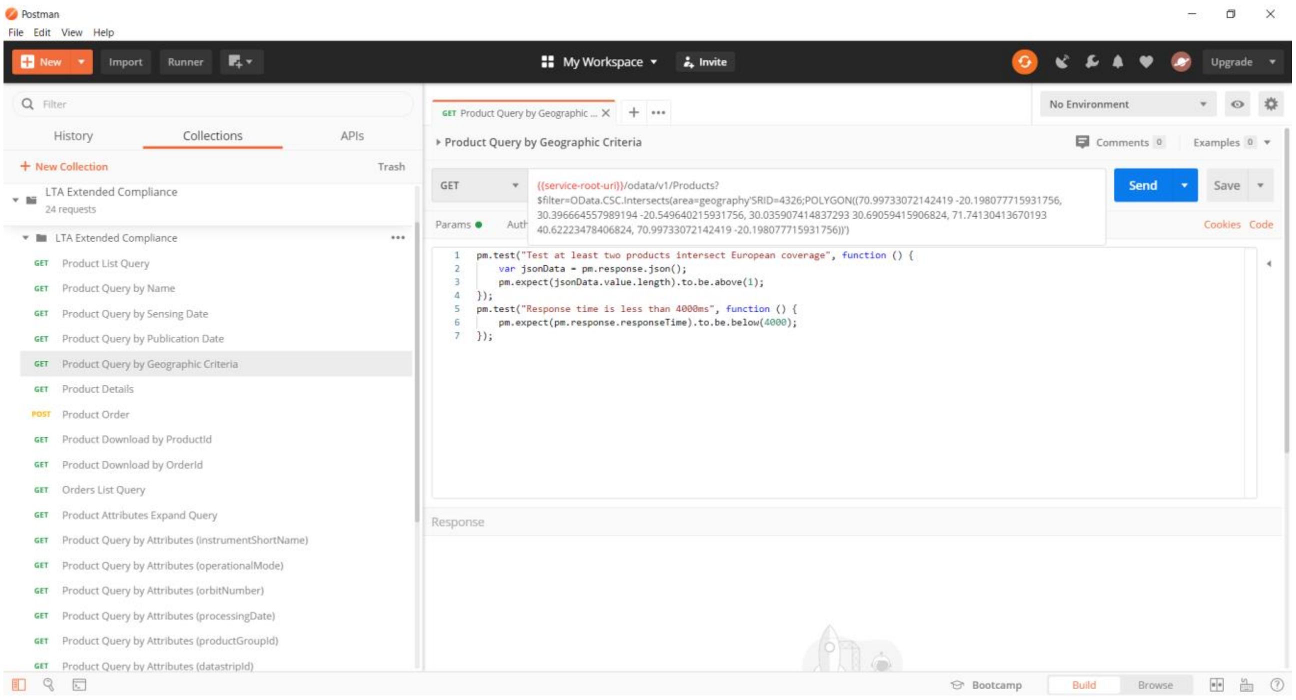
GET {{service-root-uri}}/odata/v1/Products?$filter=PublicationDate gt 2020-01-31T12:00:00.000Z and PublicationDate lt 2020-03-31T12:00:00.000Z

```
pm.test("Test product filter publication date", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value.length).to.be.above(1);
});
```

The test checks that at least one Product is returned. However, the Publication Date period shall cover at least part of, or the full ingestion period in the LTA. This will result in the query returning multiple products. Manual inspection should be used to confirm that results outside the time period submitted in the query are not returned.

5.2.5 *Product Query By Geographic Criteria*

The fifth test introduces the query by geographic criteria and queries for products intersecting a polygon consisting of several geographical points.


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
New Collection
Trash
LTA Extended Compliance
24 requests
LTA Extended Compliance
GET Product List Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders List Query
GET Product Attributes Expand Query
GET Product Query by Attributes (instrumentShortName)
GET Product Query by Attributes (operationalMode)
GET Product Query by Attributes (orbitNumber)
GET Product Query by Attributes (processingDate)
GET Product Query by Attributes (productGroupId)
GET Product Query by Attributes (datastripId)
GET Product Query by Geographic ...
No Environment
Product Query by Geographic Criteria
Comments 0
Examples 0
Send
Save
Params
Auth
Cookies
Code
Response
Bootcamp
Build
Browse

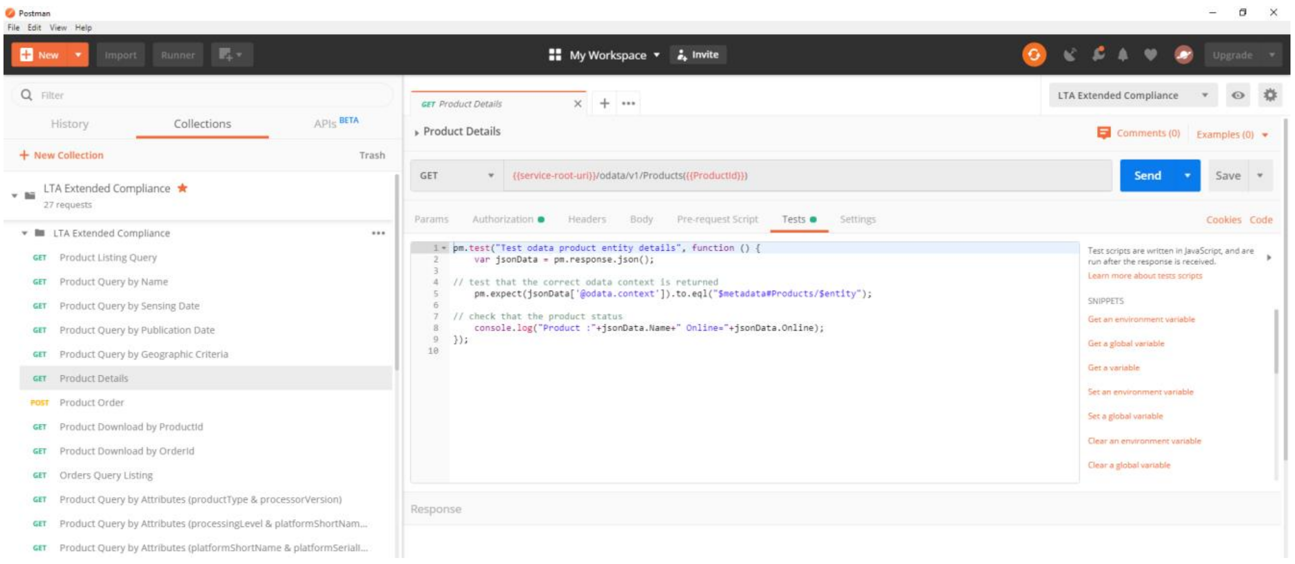
```
GET {{service-root-uri}}/odata/v1/Products?$filter=OData.CSC.Intersects(area=geography'SRID=4326;POLYGON((70.99733072142419 -20.198077715931756, 30.396664557989194 -20.549640215931756, 30.035907414837293 30.69059415906824, 71.74130413670193 40.62223478406824, 70.99733072142419 -20.198077715931756))')
```

```
pm.test("Test at least two products intersect European coverage", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value.length).to.be.above(1);
});
pm.test("Response time is less than 4000ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(4000);
});
```

The test checks that at least one Product is returned. However, since the polygon chosen for this test represents almost the entire geographical coverage of Europe, multiple products are expected to be returned.

5.2.6 *Product Details*

The sixth test checks that the product can be referenced via its ID and outputs the current status offline/online to the console.

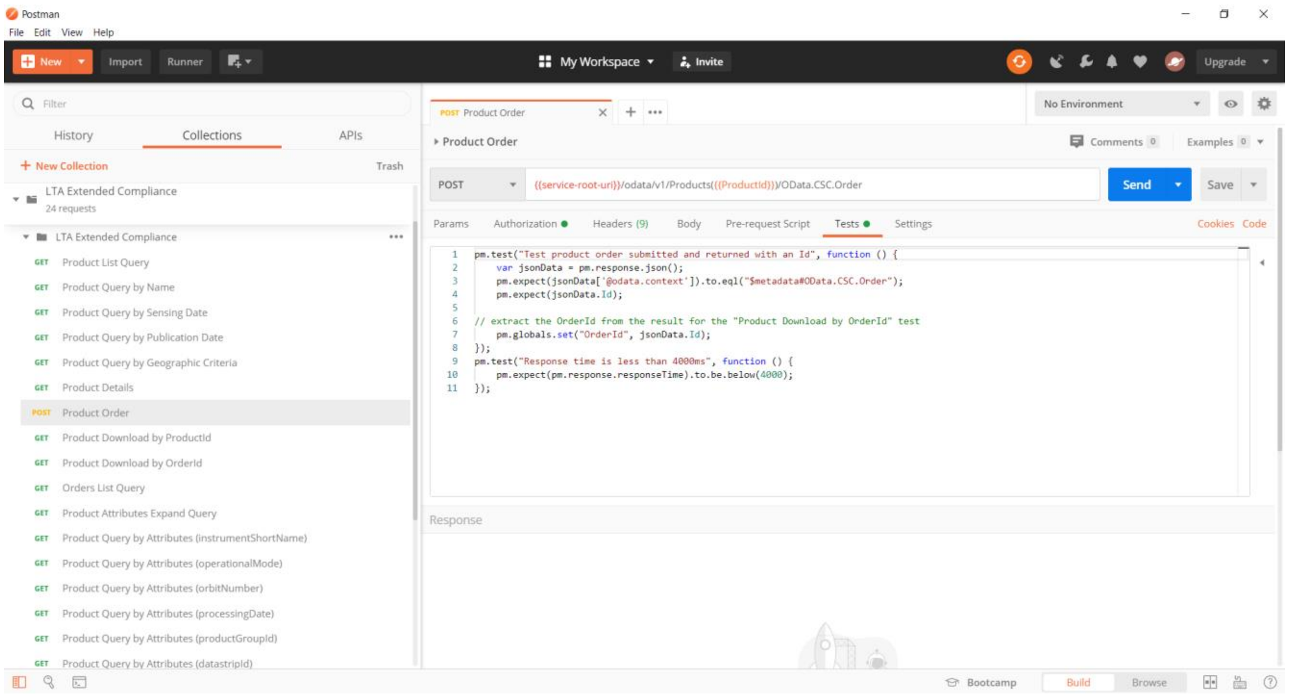


GET {{service-root-url}}/odata/v1/Products({{ProductId}})

```
pm.test("Test odata product entity details", function () {
    var jsonData = pm.response.json();

// test that the correct odata context is returned
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#Products/$entity");

// check that the product status
    console.log("Product :"+jsonData.Name+" Online="+jsonData.Online);
});
```

5.2.7 *Product Order*

Assuming the product is offline (Online=false) the seventh test orders the product, triggering its delivery to the AIP for download. The test confirms that an order is made and provided with an ID.



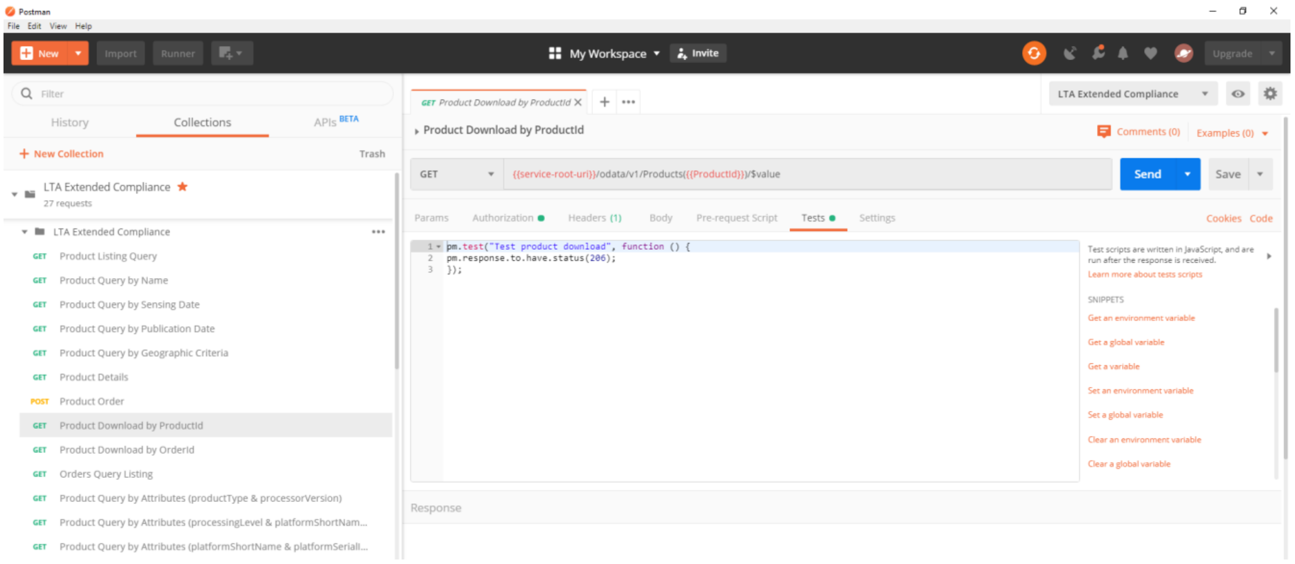
POST {{service-root-url}}/odata/v1/Products({{ProductId}})/OData.CSC.Order

```
pm.test("Test product order submitted and returned with an Id", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#OData.CSC.Order");
    pm.expect(jsonData.Id);

// extract the OrderId from the result for the "Product Download by OrderId" test
    pm.globals.set("OrderId", jsonData.Id);
});
pm.test("Response time is less than 4000ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(4000);
});
```

The test in step 5.1.4 may be repeated to verify when the data is online (Online=true) and ready for download.

5.2.8 Product Download by ProductId

The eighth step tests the product download using the ProductId. Due to limitations in managing large file download in Postman the download request is made for only the first 1024 bytes of data. Therefore a 206 Partial Content HTTP response code is expected.


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs BETA
New Collection
Trash
LTA Extended Compliance
27 requests
LTA Extended Compliance
GET Product Listing Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders Query Listing
GET Product Query by Attributes (productType & processorVersion)
GET Product Query by Attributes (processingLevel & platformShortNam...
GET Product Query by Attributes (platformShortName & platformSeriall...
GET Product Download by ProductId
LTA Extended Compliance
Comments (0)
Examples (0)
Send
Save
Params
Authorization
Headers (1)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts
SNIPPETS
Get an environment variable
Get a global variable
Get a variable
Set an environment variable
Set a global variable
Clear an environment variable
Clear a global variable
Response

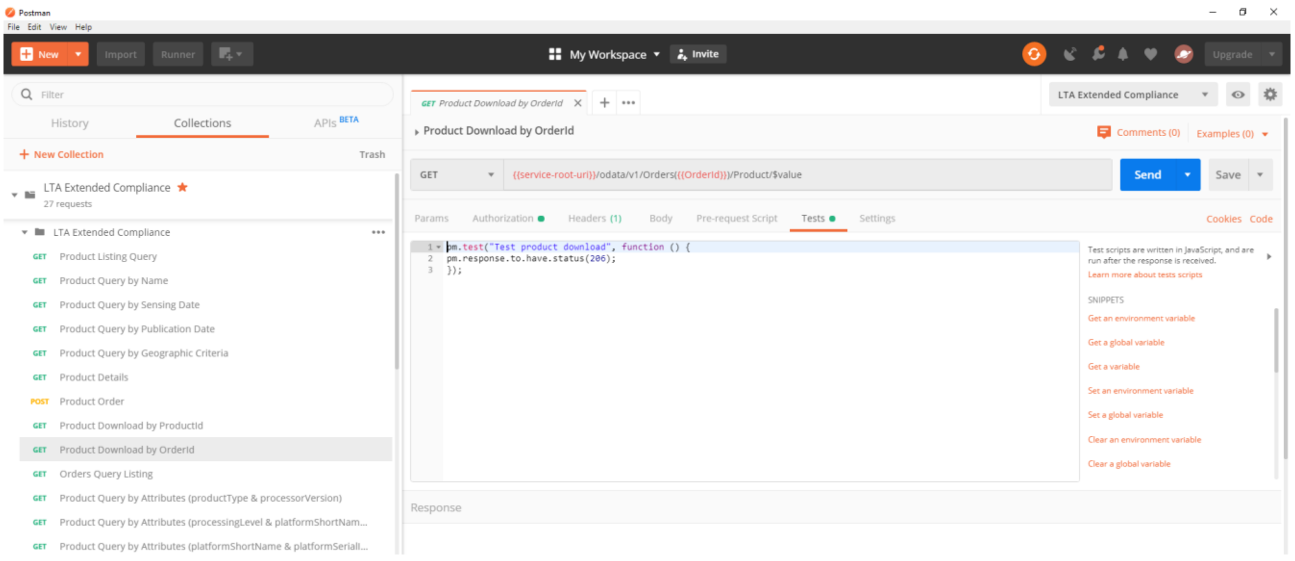
Product Download by ProductId

GET {{service-root-uri}}/odata/v1/Products({{ProductId}})/$value

```
pm.test("Test product download", function () {
pm.response.to.have.status(206);
});
```

5.2.9 Product Download by OrderId

The ninth step tests the product download using the OrderId, returned in the Product Order Query (Test #7). Due to limitations in managing large file download in Postman the download request is made for only the first 1024 bytes of data. Therefore a 206 Partial Content HTTP response code is expected.


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs BETA
New Collection
Trash
LTA Extended Compliance
27 requests
LTA Extended Compliance
GET Product Listing Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders Query Listing
GET Product Query by Attributes (productType & processorVersion)
GET Product Query by Attributes (processingLevel & platformShortNam...
GET Product Query by Attributes (platformShortName & platformSeriall...
GET Product Download by OrderId
LTA Extended Compliance
Comments (0)
Examples (0)
Send
Save
Params
Authorization
Headers (1)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts
SNIPPETS
Get an environment variable
Get a global variable
Get a variable
Set an environment variable
Set a global variable
Clear an environment variable
Clear a global variable
Response

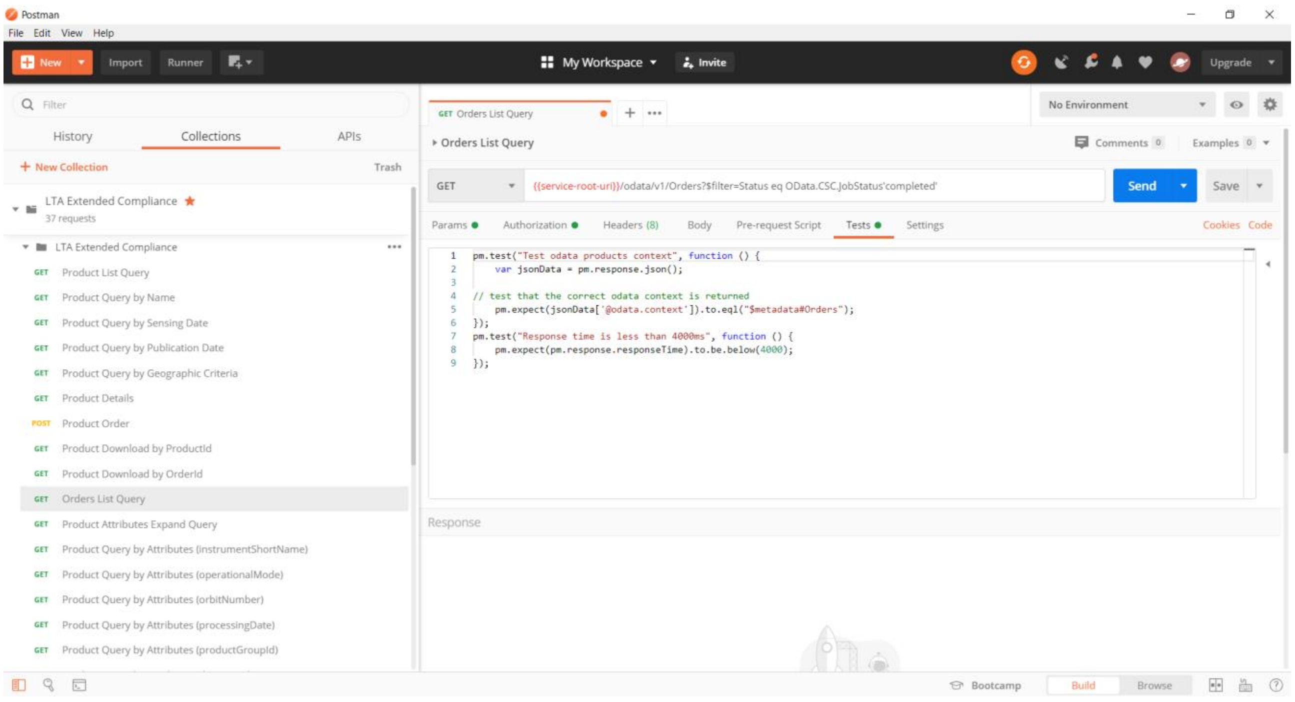
Product Download by OrderId

GET {{service-root-uri}}/odata/v1/Orders({{OrderId}})/Product/$value

```
pm.test("Test product download", function () {
pm.response.to.have.status(206);
});
```

5.2.10 Order List Query

This test introduces the Orders listing query and should return the list of completed Orders. It is also intended to verify the OData context of the response.

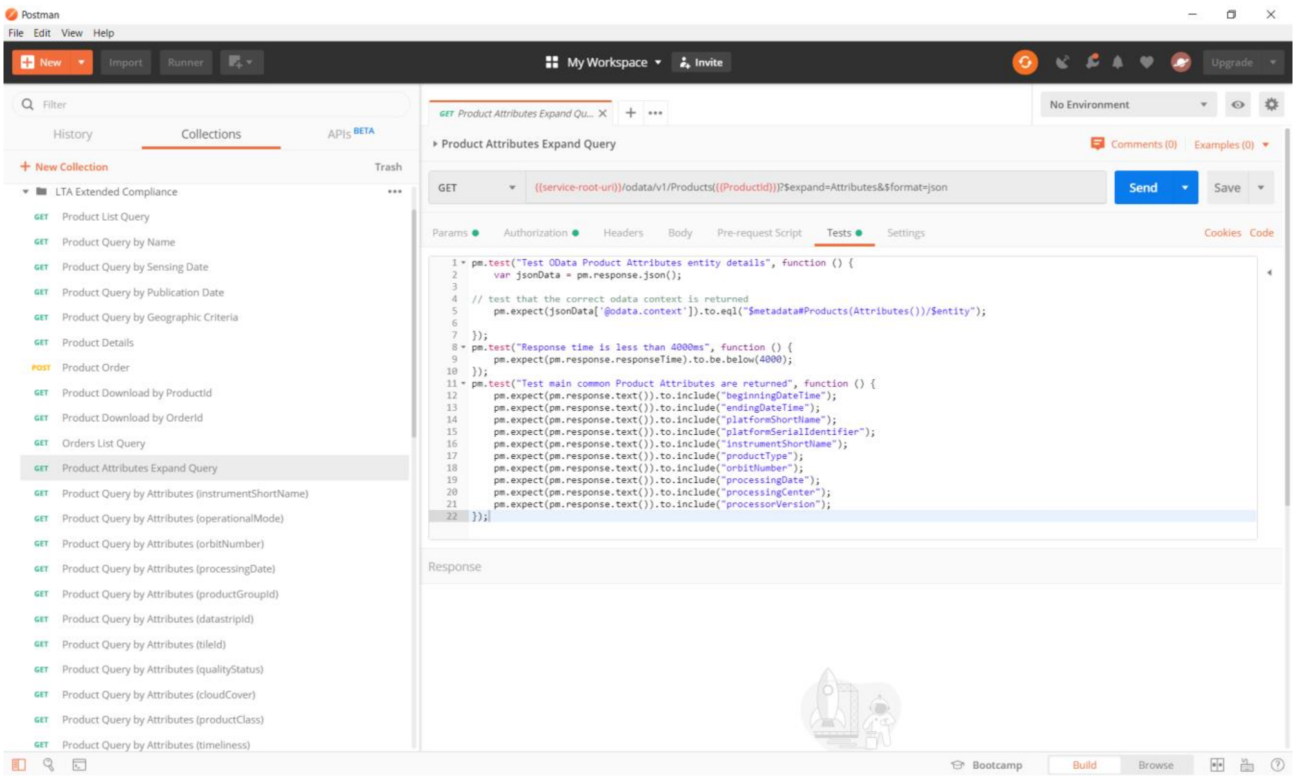

Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
New Collection
Trash
LTA Extended Compliance
37 requests
LTA Extended Compliance
GET Product List Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders List Query
GET Product Attributes Expand Query
GET Product Query by Attributes (instrumentShortName)
GET Product Query by Attributes (operationalMode)
GET Product Query by Attributes (orbitNumber)
GET Product Query by Attributes (processingDate)
GET Product Query by Attributes (productGroupId)
GET Orders List Query
No Environment
Orders List Query
Comments 0
Examples 0
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Response
Bootcamp
Build
Browse

```
GET {{service-root-uri}}/odata/v1/Orders?$filter=Status eq OData.CSC.JobStatus'completed'
```

```
pm.test("Test odata products context", function () {
    var jsonData = pm.response.json();

// test that the correct odata context is returned
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#Orders");
});
pm.test("Response time is less than 4000ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(4000);
});
```

5.2.11 *Product Attributes Expand Query*

This query is aimed at listing a Product’s Attributes. The test verifies the OData context of the response as well as testing that the main common Product Attributes are returned.


LTA Extended Compliance

GET Product List Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders List Query
GET Product Attributes Expand Query
GET Product Query by Attributes (instrumentShortName)
GET Product Query by Attributes (operationalMode)
GET Product Query by Attributes (orbitNumber)
GET Product Query by Attributes (processingDate)
GET Product Query by Attributes (productGroupId)
GET Product Query by Attributes (datastripId)
GET Product Query by Attributes (tileId)
GET Product Query by Attributes (qualityStatus)
GET Product Query by Attributes (cloudCover)
GET Product Query by Attributes (productClass)
GET Product Query by Attributes (timeliness)

Product Attributes Expand Query

GET {{service-root-uri}}/odata/v1/Products({{ProductId}})?$expand=Attributes&$format=json

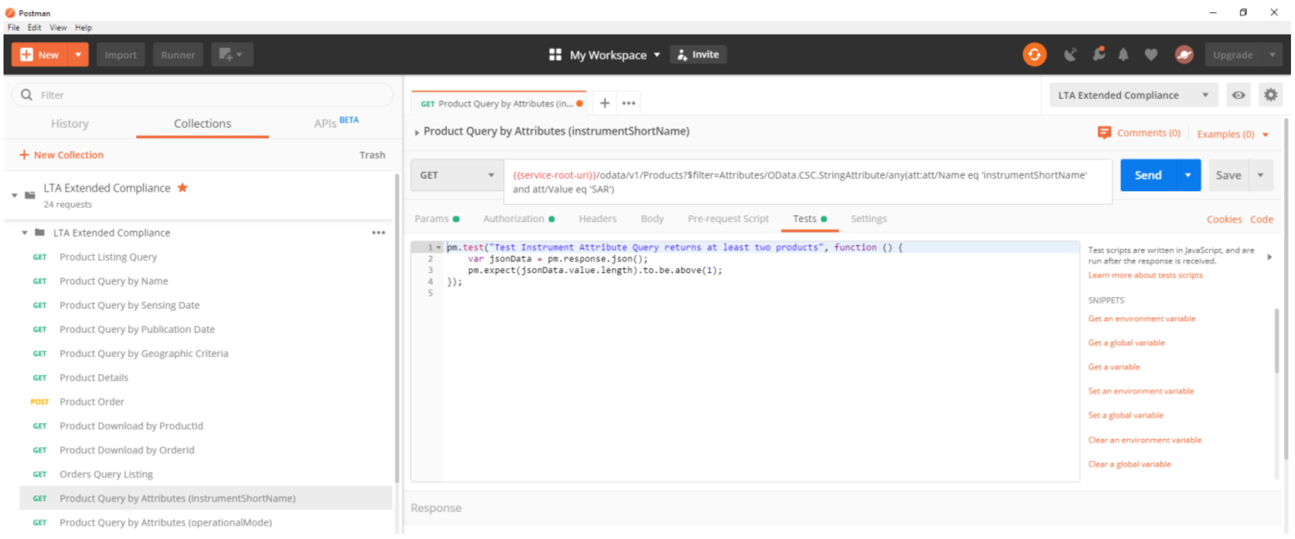
```
pm.test("Test OData Product Attributes entity details", function () {
    var jsonData = pm.response.json();

// test that the correct odata context is returned
    pm.expect(jsonData['@odata.context']).to.eql("$metadata#Products(Attributes())/$entity");

});
pm.test("Response time is less than 4000ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(4000);
});
pm.test("Test main common Product Attributes are returned", function () {
    pm.expect(pm.response.text()).to.include("beginningDateTime");
    pm.expect(pm.response.text()).to.include("endingDateTime");
    pm.expect(pm.response.text()).to.include("platformShortName");
    pm.expect(pm.response.text()).to.include("platformSerialIdentifier");
    pm.expect(pm.response.text()).to.include("instrumentShortName");
    pm.expect(pm.response.text()).to.include("productType");
    pm.expect(pm.response.text()).to.include("orbitNumber");
    pm.expect(pm.response.text()).to.include("processingDate");
    pm.expect(pm.response.text()).to.include("processingCenter");
    pm.expect(pm.response.text()).to.include("processorVersion");
});
```

5.2.12 *Product Query By Attributes*

The last tests introduce the query by Attributes. Attributes are mapped to metadata as described in [AD-1], [AD-2], [AD-3], [AD-4], [AD-5]. Queries can be performed on several individual attributes or combinations of attributes. The test suite only contains a subset of the attributes described in [AD-1], [AD-2], [AD-3], [AD-4], [AD-5], in particular:

- Query by instrumentShortName
- Query by operationalMode
- Query by orbitNumber
- Query by processingDate
- Query by productGroupId
- Query by datastripId
- Query by tileId
- Query by qualityStatus
- Query by cloudCover
- Query by productClass
- Query by timeliness
- Query by combining productType and processorVersion Attributes
- Query by combining platformShortName & platformSerialIdentifier Attributes

These tests verify that at least one (in several cases, at least 2) products are returned. In addition to this, some Attributes Queries tests verify that the products returned conform to the request made, through checking the content of the response body (e.g. the returned filename). Where the information requested is not a part of the filename, manual inspection should be performed in order to confirm that the returned products conform to the input attributes.


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs BETA
New Collection
Trash
LTA Extended Compliance
24 requests
LTA Extended Compliance
GET Product Listing Query
GET Product Query by Name
GET Product Query by Sensing Date
GET Product Query by Publication Date
GET Product Query by Geographic Criteria
GET Product Details
POST Product Order
GET Product Download by ProductId
GET Product Download by OrderId
GET Orders Query Listing
GET Product Query by Attributes (instrumentShortName)
GET Product Query by Attributes (operationalMode)
GET Product Query by Attributes (in...
Product Query by Attributes (instrumentShortName)
Comments (0)
Examples (0)
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Settings
Cookies
Code
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts
SNIPPETS
Get an environment variable
Get a global variable
Get a variable
Set an environment variable
Set a global variable
Clear an environment variable
Clear a global variable
Response

GET {{service-root-uri}}/odata/v1/Products?$filter=Attributes/OData.CSC.StringAttribute/any(att:att/Name eq 'instrumentShortName' and att/Value eq 'SAR')

```
pm.test("Test Instrument Attribute Query returns at least two products", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value.length).to.be.above(1);
});
```

Note: In addition to the individual tests performed for each query, there are tests that verify the response time for all queries, except for the Product Download Queries (5.2.8 and 5.2.9), where the response time may vary significantly depending on certain factors (e.g. product size, retrieval time, etc.).