Data Hub Software (DHuS) OData and Open Search Interface Control Document

DHuS Open Source Framework

Role/Title	Name	Signature	Date
Authors	Adriana Grazia Castriotta		24/06/2015
	Calogera Tona		
	Stéphane Mbaye		
	Frédéric Pidancier		
Reviewed	Christophe Demange		
Approved	Andrea Tesseri –Contract Manager		

Contract: 40000113153

Change register

Version/Rev.	Date	Reason for Change	Pages modified
1.0 Draft	29/05/2015	First issue	
1.1 Final	24/06/2015	Updates after ESA review	

Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This Interface Control Document (ICD) documents and tracks the necessary information required to effectively define the DHuS system's interface as well as any rules for communicating with them. The document is intended in order to give both end users guidance on usage of the external interface of the system and to give developers guidance on architecture of the interfaces and how they should be maintained when evolving the system or adding new product types.

This document is a deliverable of the Open Source Framework Contract (ref.: 40000113153).

1.2 Document Structure

- Section 1: Introduction-this section
- Section 2: OData Interface
- Section 3: Responses
- Section 4: Open Search Interface

Mission specific details are provided as separated documents.

1.3 Definition and Acronyms

Definition	Description
User	Any user of the Data Access system having completed the registration loop or having the login credentials.
Authorized User	User having access to a specific sub-set of products
Collection	A logical grouping of data products defined at DHuS level. Groupings could be by one or a combination of products according the operator rules he/she sets.

	DHuS collections may or may not reflect the source collections or datasets of the ingested products.
Operator	A specific type of DHuS user, with access to the full set of DHuS functionality for operations and administration of the system.

Table 1 Definition

Acronym	Term
DHuS	Data Hub Software
DHuS OSF	Data Hub Software Open Source Framework
OS	Open Source
OSS	Open Source Software

Table 2 Acronym

1.4 Reference Documents

Id	Title	Reference	Issue
RD-1	OASIS OData	http://www.odata.org/documentation/odata-version-2-0/	
RD-2	Apache Solr Reference Guide Covering Apache Solr 4.7	https://www.apache.org/dyn/closer.cgi/lucene/solr/ref-guide/	
RD-3	DHuS OData Software Design Dcoument	GAEL-P286-SDD-003	
RD-4	Data Request Broker	http://www.gael.fr/drb/	
RD-5	Apache Olingo	https://olingo.apache.org/ https://olingo.apache.org/doc/odata2/index.html http://www.odata.org/libraries/	

Table 3 Reference Documents

2 OData Interface

This section specifies the DHuS OData Entity Data Model (EDM) that controls the Objects exposed and those that can be manipulated through the interface e.g. Products, Collections, Users, etc.

The OData interface is provided on the basis OASIS v2 implementation [RD-02].

2.1 Entity Data Model (EDM)

The DHuS OData Entity Data Model is depicted in by the following UML class diagram:

- the orange is for entities that are directly exposed through the interface;the yellow are only visible when accessed through orange ones;
- the white i.e. Item is abstract and never instantiated.

Finally, the Users/User in gray is prototyped but to be considered for future releases.

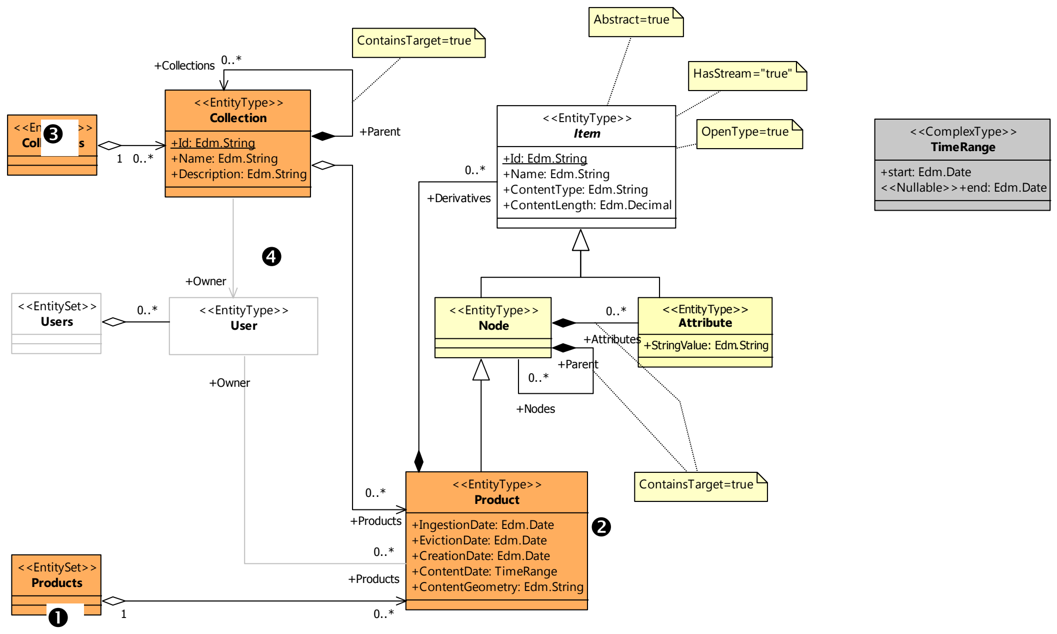

Figure 1DHuS OData Entity Data Model

The EntitySets, EntityTypes, ComplexTypes and Properties of the diagram above are described in the following sections. The order of description is mainly driven by the reader completion rather than any other based on typing or thematic order.

2.2 Products EntitySet

The primary container of the exposed OData EDM is the `Products EntitySet` that represents a list of `Products` entities cf. section 2.7 below. The `Products` are listed independently from their belonging `Collections`.

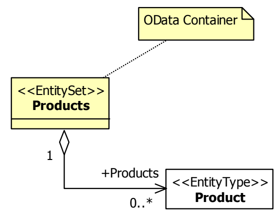

Figure 2-Products EntitySet

This list contains only the `Products` visible by the requesting users i.e. the owners or explicitly authorized users.

2.3 Product EntityType

The `Product EntityType` represents the primary payload of the DHuS server. This type derives from the `Node EntityType`, itself deriving from the Item abstract `EntityType` specified respectively in sections 2.7 and 2.6. Thus the Product inherits its properties from both the Node and the Item EntityTypes.

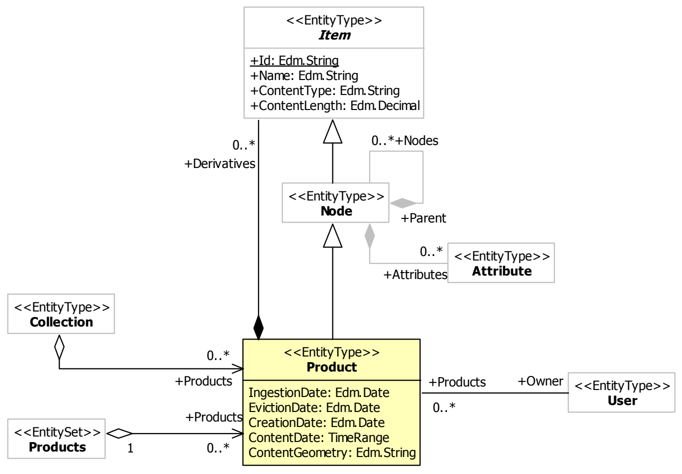

Figure 3-Product EntityType

The `Product EntityType` is readable by users explicitly authorized, or removable only by the operator. Products may be created by authorized users.

NOTE – Because the Product entity type is an Item, it has an HasStream attribute set to "true". As such, the Product entities have an attached stream through which the actual product content can be downloaded and uploaded.

2.3.1 Products Properties

The Product EntityType accepts the following properties.

Id

The `Id` is a non nullable Edm.String (sequence of numbers and characters) representing the Universally Unique Identifier (UUID) string conforming to the IETF RFC/4122. The `Id` uniquely identifies the `Product` in its container. The UUDI form is enforced to minimize the number of collisions between all DHuS instances and in particular without necessary centralized coordination. As UUDI has many variants, the exact format of this identifier is not tied to a particular lexical space. If necessary, this `Id` property could even be not strictly conformant to the UUDI variants but users shall be conscious that a DHuS instance may reject an `Id` value because of internal inconsistencies or may behave improperly after product ingestion, for example, in the case of federation between two DHuS instances that may have conflicting `Id`s.

UUID is not a user-friendly string because of its length and sequence of interlaced digits and characters. However, the `Id`s are generated by the DHuS and users will generally copy the values exposed without editing it manually.

Selecting a Product from its Id (from the Products EntitySet)
```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')
```

Name

The `Name` of a `Product` is a nullable Edm.String (sequence of characters). It is not a recommended practice to have a null `Name`. For an EO product, the `Name` should usually be similar to the file or archive file name of the product that identifies the acquiring sensor, its primary mode, the acquisition time frame, etc., without extension. This file name couldn't be considered as a unique identifier because the same product may be ingested multiple times or because multiple products may be processed with the same name.

Getting a Product Name (XML response)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/Name`
Getting a Product Name (raw value)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/Name/$value`
Selecting a Product via its Name property
```
/odata/v1/Products?$filter=Name eq
'S1A_EW_GRDH_1SDH_20140710T131138_20140710T131205_001426_0015EB_52EB''
```

ContentType

The `ContentType` has a same definition as the abstract `Item/ContentType` cf. section 2.6 below.

The `ContentType` is a nullable Edm.String (sequence of characters) representing the `MIME` type of data payload downloadable from this `Product` i.e. the format of the stream resulting from a request with an OData `$value` option. The `ContentType` shall be null if and only if the `Product` has no content. It shall be equal to the `Content-Type` header of the HTTP response returned by the DHuS when downloading this `Product`.

This property is read-only. Priority is given to the HTTP request `Content-Type` HTTP headers provided during uploads.

For EO products, the `ContentType` is usually `application/octet-stream`, `application/zip` or `application/x-gtar`.

Getting a Product ContentType (XML response)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/ContentType`
Getting a Product ContentType (raw value)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/ContentType/$value`

NOTE – This property is primarily specified to provide clients with content type information along with other `Product`'s properties in a single HTTP request.

NOTE – The HTTP `Content-Type` header should not be confused with the `Content-Encoding`. The `Content-Type` is the MIME type of the actual payload that has to be preserved as is, while the `Content-Encoding` regards a modification applied for the transfer and that has to be reversed immediately after reception to restore the genuine payload.

<u>ContentLength</u>

The `ContentLength` has a same definition as the abstract `Item/ContentLength` cf. section 2.6 below.

The `ContentLength` is a Edm.Decimal representing the size in bytes of the payload downloadable from the stream denoted by this `Product`. It is a nullable non-negative integer with an unspecified Precision and a zero Scale i.e. unlimited number of digits and no digits to the right of the decimal point according to OData specifications. The `ContentLength` shall be null or equal to zero only and only if the `Product` has no content i.e. no byte to feed the stream. It shall be equal to the `Content-Length` header of the HTTP response returned by the DHuS when downloading this `Product` if no HTTP `Content-Encoding` has been applied. Otherwise the `ContentLength` is the size of the payload once the HTTP `Content-Encoding` is reversed, and may therefore be different from the `Content-Length` HTTP header.

This property is read-only. Priority is given to the HTTP request `Content-Length` HTTP headers provided during uploads.

Getting a Product ContentLength (XML response)

```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/ContentLength
```

Getting a Product ContentLength (raw value)

```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-
0800200c9a66')/ContentLength/$value
```

Filter Products having ContentLength grater or equal to 40000000 bytes

```
/odata/v1/Products?$filter=ContentLength ge 400000000
```

NOTE – This property is primarily specified to provide clients with content type information along with other `Product`'s properties and in a single HTTP request.

<u>CreationDate</u>

`CreationDate` is a non-nullable Edm.Date property indicating the date of creation of this `Product` instance in the contacted DHuS. The `CreationDate` is expressed in UTC. A Product shall always have a `CreationDate`. This property has nothing to do with the `Product` content and in particular with the date on which the content has been processed i.e. also called processing date in EO domain.

Getting a Product CreationDate (XML response)

```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/CreationDate
```

Getting a Product CreationDate (raw value)

```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-
0800200c9a66')/CreationDate/$value
```

<u>IngestionDate</u>

`IngestionDate` is a non-nullable Edm.Date property denoting the date on which the `Product` was ingested by the contacted DHuS instance. The date referential is UTC.

Getting a Product IngestionDate (JSON response)

```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/IngestionDate
```

Getting a Product IngestionDate (raw value)

```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-
0800200c9a66')/IngestionDate/$value
```

<u>ContentDate</u>

`ContentDate` is a nullable DHuS.TimeRange property denoting the time period associated to the Product payload. This date regards the period of the subject of the content and not the content itself. For example, for an EO product, the `ContentDate` deals with the observation date and not to the downlink or processing dates. For a document, the `ContentDate` is the period addressed by the document and not the date on which it was written or issued.

The `ContentDate` is an interval based on a `TimeRange ComplexType` created for the DHuS. Thus the `ContentDate` accepts two nullable properties named `Start` and `End`, both of type `Edm.Date` and expressed in the UTC time reference system. It is an error if anyone of these two properties is null. If no `ContentDate` is known at all, the overall `ContentDate` entity shall be set to null. For instantaneous content or for any case where `Start` and `End` are undifferentiated, both properties shall have the same value, so that no OData client would fail while querying only one property.

Getting a Product ContentDate (XML response including star and end)
```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/ContentDate
```
Getting a Product ContentDate Start date (raw value)
```
/odata/v1/Products('e91ac9a0-…66')/ContentDate/Start/$value
```
Querying Products acquired before 2009 (XML response)
```
/odata/v1/Products?$filter=year(ContentDate/End) le 2008
```

Checksum

`Checksum` is a DHuS.Checksum property denoting the MD5 of the product. Since the Checksum is a value calculated on the product after its compression, this value helps users detecting errors which may have been introduced during its transmission of products from DHuS to the users' storage.

Getting a Product Checksum (XML response)
```
/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/Checksum
```
Getting a Product Checcksum (raw value)
```
/odata/v1/Products('e91ac9a0-…66')/Checksum/Value/$value
```

ContentGeometry

`ContentGeometry` is a nullable Edm.String property denoting the geometry of the product. At the current state of the analysis, the string content should be expressed as an XML document using GML language.

As for the `ContentDate` above, the `ContentGeometry` does not deal with the content itself but with the one of the content subject. For EO product, the `ContentGeometry` could be the geographical boundaries of the observed area. For a document, the `ContentGeometry` would deal with the area addressed by the text rather than the location where it has been written or printed.

Getting a Product ContentDate (XML response including star and end)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/ContentGeometry`
Getting a Product ContentDate Start date (raw value)
`/odata/v1/Products('e91ac9a0-…66')/ContentGeometry/$value`

Metalink

`Metalink` is a not-nullable Edm.String property denoting the file name and the URI for download the product.

Getting a Product Metalink (XML response)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/Metalink`

2.3.2 Products Navigation Properties

The Product EntityType accepts the following navigation properties.

Nodes

The `Nodes` is a nullable, read-only, containment navigation property representing the roots of a forest of `Nodes` modeling the content of the `Product`. This property is inherited from the `Node` entity type defined in section 2.7.

The availability of `Nodes` depends of the DHuS configuration and in particular to the level of support or description provided to the underlying Data Request Broker (DRB) API [RD-3] It may vary from a simple file representation to a deep tree with leaves representing small binary fields or even single bits.

Getting all Nodes of the Product (XML response)

```
/odata/v1/Products('2573986b-f66e-46a4-90e8-
00598c3b6475')/Nodes('S1A_S5_GRDH_1SSV_20141003T182910_20141003T182928_002669
_002F8D_E968.SAFE')/Nodes
```

Getting all children of the Node 'annotation' of the Product (XML response)

```
/odata/v1/Products('2573986b-f66e-46a4-90e8-
00598c3b6475')/Nodes('S1A_S5_GRDH_1SSV_20141003T182910_20141003T182928_002669
_002F8D_E968.SAFE')/Nodes('annotation')/Nodes
```

Browsing nodes, the user will arrive to the last level and to get it content the user may add the following string to its URI, as shown in the following example: `/Value/$value`

Browsing a content of the 'antenna-pattern[18]/swath' Product Node (XML response)

```
/odata/v1/Products('2573986b-f66e-46a4-90e8-
00598c3b6475')/Nodes('S1A_S5_GRDH_1SSV_20141003T182910_20141003T182928_002669
_002F8D_E968.SAFE')/Nodes('annotation')/Nodes('s1a-s5-grd-vv-20141003t182910-
20141003t182928-002669-002f8d-
001.xml')/Nodes('product')/Nodes('antennaPattern')/Nodes('antennaPatternList'
)/Nodes('antennaPattern%5B18%5D')/Nodes('swath')/Value/$value
```

The Nodes exploration allow the download of part of a products, e.g the download of the manifest file as shown in the example below:

Download just the manifest of the product (XML response)

```
/odata/v1/Products('2573986b-f66e-46a4-90e8-
00598c3b6475')/Nodes('S1A_S5_GRDH_1SSV_20141003T182910_20141003T182928_002669
_002F8D_E968.SAFE')/Nodes('manifest.safe')/$value
```

Attributes

The `Attributes` is a nullable, read-only, containment navigation property that represents the named metadata of the `Product`. This property is inherited from the `Node` entity type defined in section 2.7. The list of available Attributes for a given Product depends on the product type and the configuration of the DHuS. Typical examples of metadata for EO products provide the sensor name, mode, cycle, orbit number, etc.

Getting all Attributes of a Product (XML response including star and end)
`/odata/v1/Products('e91ac9a0-ea35-11e3-ac10-0800200c9a66')/Attributes`
Getting 'Pass direction' Attribute of a Product (XML response)
`/odata/v1/Products('e91ac9a0-…')/Attributes('Pass direction')`
Getting 'Pass direction' value of a Product (raw value)
`/odata/v1/Products('e91ac9a0-…')/ Attributes('Pass direction')/Value/$value`

2.4 Collections EntitySet

Another container is the `Collections` that represents the list of the root `Collections` entities cf. section 2.5 below.

This list contains only the `Collections` visible by the requesting users i.e. the owners or explicitly authorized users. Similarly the `Collections EntitySet` is modifiable only by authorized users.

2.5 Collection EntityType

The `Collection EntityType` represents a logical group of `Products`. The `Collections` may have nested `Collections` to for a logical tree, but a single `Collection` can only have one parent `Collection`.

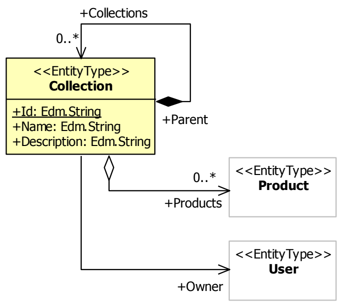

Figure 4-Collection EntityType

The `Collection EntityType` is readable, modifiable or removable only by their owners and users explicitly authorized. `Collections` may be created by authorized users.

The `Collection EntityType` accepts the following properties.

Id

The Id is a non-nullable Edm.String identifying the `Collection` URI. The Id shall be unique in its container and thus unique in the `Collections` container or unique in the parent `Collection`.

Getting the list of Collections id (XML response)
```
/odata/v1/Collections
```
Getting the list of products in the 'collection_ID' Collection (XML response)

```
/odata/v1/Collections('collection_ID')/Products
```
Getting the list of subcollections of the 'collection_ID' Collection (XML response)
```
/odata/v1/Collections('collection_ID')/Collections
```
Selecting the first Products of a 'Sentinel' Collection (XML response)
```
/odata/v1/Collections('Sentinel')/Products?$top=1
```
Selecting the 'EW' products under the 'collection_ID' Collection (XML response)
```
/odata/v1/Collections('collection')/Products?$filter=substringof(%27EW%27,Nam
e)
```
Name

The `Name` is a non-nullable Edm.String providing a human readable identifier of the `Collection`. The lexical space of the `Name` is left unconstrained but OData clients should avoid zero-length string that are meaningless and special characters that may require specific encoding when used in an URL or as a folder or tree node name.

Description The `Description` is a nullable EDM.String describing the `Collection`.

Getting the description of the 'collection_ID' Collection (raw value)
```
/odata/v1/Collections('collection_ID')/Description/$value
```

2.6 Item EntityType

The `Item EntityType` is an "abstract" type that intends to represent any piece of data handled by a DHuS server. For example, an Item may represent a `Product`, a `Node` i.e. a part of a `Product`, an `Attribute` or any of their derivatives.

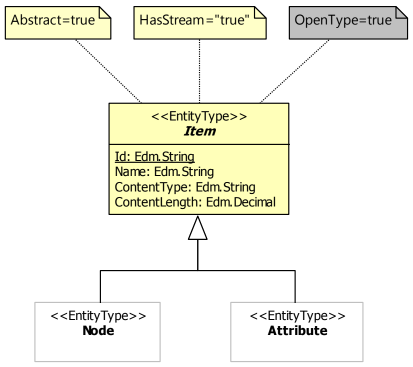

Figure 5-Item EntityType

An `Item` may be logical element or may have a data payload and as such has a `HasStream` attribute set to "`true`" in the OData EDM.

The `Item` has also an `OpenType` attribute set to `true`, specifying that properties not declared in the EDM may appear in the responses of the service.

NOTE – The `OpenType` attribute value is experimental in these specifications and may change in future versions. Minor consequences are to be expected for clients that use the properties explicitly specified in the EDM.

2.6.1 Item Properties

The `Item EntityType` accepts the following properties.

Id

The `Id` is a non nullable Edm.String including a sequence of at least one character. The `Id` uniquely identifies the `Item` in its container. The uniqueness property depends on the nature of the container as specified by OData.

The specifications of this property are completed by the extending `EntityType`'s e.g. `Product`, `Node`, and `Attribute`.

Name

The `Name` of an `Item` is a nullable Edm.String including a sequence of characters. The `Name` is to be considered as a human readable label helping the identification of the `Item`. Uniqueness of the `Name` among its `Item` sibling is preferable but is not enforced as for the `Id`. This property is supposed to be equal to the `Id` in many situations where the `Id` remains readable, which is not the case for `UUID`'s as the following `e91ac9a0-ea35-11e3-ac10-0800200c9a66`.

ContentType

The `ContentType` is a nullable Edm.String including a sequence of characters representing the `MIME` type of data payload downloadable from this `Item` i.e. the format of the stream resulting from a request with an OData `$value` option. The `ContentType` shall be null if and only if the `Item` has no content i.e. no byte to feed the stream. It shall be equal to the `Content-Type` header of the HTTP response returned by the DHuS when downloading this `Item`.

This property is read-only. Priority is given to the HTTP request `Content-Type` HTTP headers provided during uploads.

NOTE – This property is primarily specified to provide clients with content type information along with other `Item`'s properties and in a single HTTP request.

NOTE – The HTTP `Content-Type` header should not be confused with the `Content-Encoding`. The `Content-Type` is the MIME type of the actual payload that has to be preserved as is, while the `Content-Encoding` regards a modification applied for the transfer and that has to be reversed immediately after reception to restore the genuine payload.

ContentLength

The `ContentLength` is a Edm.Decimal indicating the size in bytes of the payload downloadable from the stream denoted by this `Item`. It is a nullable non-negative integer with an unspecified Precision and a zero Scale i.e. unlimited number of digits and no digits to the right of the decimal point according to OData specifications. The `ContentLength` shall be null or equal to zero only and only if the `Item` has no content i.e. no byte to feed the stream. It shall be equal to the `Content-Length` header of the HTTP response returned by the DHuS when downloading this `Item` if no HTTP `Content-Encoding` has been applied. Otherwise the `ContentLength` is the size of the payload once the HTTP `Content-Encoding` reversed, and may therefore be different from the `Content-Length` HTTP header.

This property is read-only. Priority is given to the HTTP request `Content-Length` HTTP headers provided during uploads.

NOTE – This property is primarily specified to provide clients with content type information along with other `Item`'s properties and in a single HTTP request.

2.7 Node EntityType

The `Node EntityType` inherits its definition from the `Item EntityType`. A `Node` is an `Item` that is a container of other `Nodes`, all forming a tree representation of the modeled data. The exact semantic of a `Node` depends of its usage and in particular its level. It can represent an archive file at `Product` level, a directory, a file, an element in an XML document, a record in a database or a field in a binary file at the leaves level.

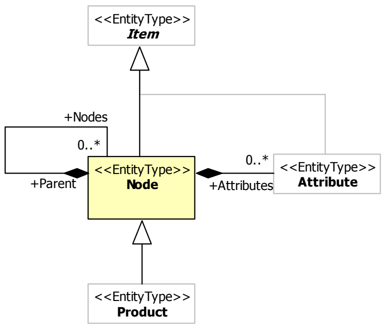

Figure 6-Node EntityType

NOTE - Because the Node entity type is an Item, it has an HasStream attribute set to "true". As such, a `Node` can have a value that can be downloaded depending on the payload it represents. Typically, a file represented by a `Node` can be downloaded, even nested in a compressed archive. Similarly, a record of a binary file could be downloaded as a binary segment or as an XML fragment once converted. The download capability does not guarantee to download any `Product` sub-`EntityType` .

2.7.1 Node Properties

The `Node EntityType` accepts the following properties.

Id

The `Id` is an Edm.String having a same definition as the abstract `Item/Id` cf. section 2.6 above. It shall be noticed that `Nodes` are only contained by a `Node` or by a single entity. Therefore, the uniqueness if the `Id` property is in the scope of the parent container e.g. a `Node` denoting a file may have an `Id` equal to the filename which is unique with respect to its siblings sharing the same parent `Node` denoting a folder.

Other properties

Other properties accepted by the Node EntityType are the Item properties (Name, ContentType, ContenLenght; see description in 2.6.1).

2.7.2 Node Navigation Properties

The Node EntityType accepts the following navigation properties: **Nodes** and **Attributes** and they have been described in 2.3.2.

2.8 Attributes EntityType

The `Attributes EntityType` inherits its definition from the `Item EntityType`. An `Attribute` is an `Item` that is a container of other `Attributes`, all forming a tree representation of the modeled data. The exact semantic of a `Attribute` depends of its usage and in particular its level. It represents a metadata of the `Product`.

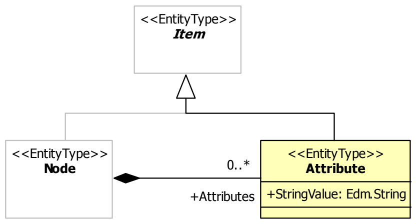

Figure 7-Attribute EntityType

2.8.1 Attributes Properties

The `Attributes EntityType` accepts the following properties.

Id
The `id` is a non-nullable Edm.String having a same definition as the abstract Item/Id cf. section 2.6 above.

Other properties

Other properties accepted by the Node EntityType are the Item properties (Name, ContentType, ContenLenght; see description in 2.6.1)

2.9 Quicklook visualization

As mentioned before the Quicklooks can be extracted through the Nodes exploration, knowing the 'UUID' and the 'Name' of the product, e.g.

Getting the Quicklook of a products using the Nodes exploration method
```
/odata/v1/Products('[UUID]')/Nodes('[NAME].SAFE')/Nodes('preview')/Nodes
```

The OData additionally exposes to users the possibility, knowing the 'UUID' of the products, to quickly get the quicklook of it, e.g.

Getting the Quicklook of a products using the fast link provided by DHuS
```
/odata/v1/Products('[UUID]')/Products('Quicklook')/$value
```

2.10 Query String Options

The OData protocol provides easy access to the Data Hub and can be used for building URI for performing search queries and product downloads.

A URI used by an OData service has up to three significant parts: the service root URI, resource path and query string options

```
odata/v1/Products?$filter=[query]&$[option]
```

where:

- `$filter=[query]&$[option]` is the **query string options** part

Query String Options admitted by the Data Hub service:

- `$format` Specifies the HTTP response format e.g. XML or JSON
- `$filter` Specifies an expression or function that must evaluate to true for a record to be returned in the collection
- `$orderby` Determines what values are used to order a collection of records
- `$select` Specifies a subset of properties to return
- `$skip` Sets the number of records to skip before it retrieves records in a collection
- `$top` Determines the maximum number of records to return

OData supports a set of built-in filter operations as shown in the table below.

Comparison Operators	Logical Operators
eq	and
ne	or
gt	not
ge	

lt	
le	

Examples:

Select the products that have been published in the Data Hub after 06 Dec 2014
```
/odata/v1/Products?$filter=IngestionDate gt datetime'2014-12-06T00:00:00'
```

List the first 100 products skipping the first 10
```
/odata/v1/Products?$orderby=IngestionDate desc&$top=100&$skip=100
```

List the Name of all products in the archive and display it in Json format
```
/odata/v1/Products?$format=json&$select=Id
```

2.10.1 Functions

OData supports a set of built-in functions that can be used within $filter operations. The following table lists the available functions and examples.

Function	Example
String Functions	
endswith	/odata/v1/Products?$filter=endswith(Name,‘E968’’)
startswith	/odata/v1/Products?$filter=startswith(Name,’S1A_EW’)
substring	/odata/v1/Products?$filter=substringof('SLC',Name)
Date Functions	
year	/odata/v1/Products?$filter=year(IngestionDate) eq 2014
month	/odata/v1/Products?$filter=mont(IngestionDate) eq 12

`day`	…
`minute`	…
`datetime`	`/odata/v1/Products?$filter=IngestionDate gt datetime'2014-12-06T00:00:00'`

2.11 Download string Option

The Download String Options admitted by the Data Hub service is the following:

- `$value` Specifies the HTTP GET request

Download the product having a uuid= 'f9114e65-76af-43db-8599-35c674e6aacb'
```
/odata/v1/Products('f9114e65-76af-43db-8599-35c674e6aacb')/$value
```

The DHuS supports also the HTTP partial GET

The request partial GET must include a Range header field indicating the desired range, and may include an If-Range header field to make the request conditional.

2.11.1 HTTP GET response

The response of a GET or partial GET will include the following header fields:

- Either a **Content-Range header** field indicating the range included with this response, or a multipart/byteranges Content-Type including Content-Range fields for each part. If a Content-Length header field is present in the response, its value must match the actual number of OCTETs transmitted in the message-body.
- **Date** indicating the Date of the GET request
- **ETag** indicating MD5 value
- **Expires** indicating the expiring date of the GET request

Example of HTTP partial GET using curl (byte range = ' 0-999 ')

```
curl -u userid:password -k -o name_file.zip -r "0-999"
"[dhus_hostname]/odata/v1/Products('UUID')/\$value"
```

The option `-r/--range <range>` retrieves a byte range (i.e a partial document) from a HTTP/1.1, FTP or SFTP server or a local FILE.

N.B.: Ranges can be specified in a number of ways, e.g:

- *0-499 specifies the first 500 bytes*
- *500-999 specifies the second 500 bytes*
- *-500 specifies the last 500 bytes*
- *9500- specifies the bytes from offset 9500 and forward*
- *0-0,-1 specifies the first and last byte only*
- *500-700,600-799 specifies 300 bytes from offset 500*
- *100-199,500-599 specifies two separate 100-byte ranges*

2.12 OData Responses

This section specifies the response formats that are expected in output of the Service API. The formats available include XML/Atom, JSON.

As recommended by the OData specifications, the DHuS OData service supports responses in XML/ATOM or JSON. Default format is XML.

When explicitly mentioned in these specifications, the output may be formatted as Metalink, XML or other specific formats.

This section describes the responses returned from the requests sent to a DHuS OData service. The description is focused on the response body and does not cover the HTTP response envelope. A priority is given to actual examples of DHuS responses rather than to a full coverage of all flavours provided by OData protocol. .

The default response format is Atom [RFC4287], an XML-based document format that describes Collections of related information known as "feeds". The response format can however be controlled from the requests through the `$format` query option already introduced in section 2.9.1 above. The formats currently implemented by DHuS service are Atom (or XML), JSON and Metalink which is a non-standard output of OData but is an experimental behaviour useful for EO context.

The following sub-sections describe the response types for Atom/XML, JSON and Metalink output formats. The very last sub-section deals with the specific responses returned in case of error.

2.12.1 XML Responses

The Atom/XML format is the default response format, though it can be forced by setting the `$format` query option to "`atom`" or "`xml`" indifferently as in the following examples.

```
{svc-root}?$format= atom
```

```
{svc-root}?$format= xml
```

As defined in the OData Atom/XML response format specifications, the responses structure and construction may vary according to the objects returned. The following sub-sections provide typical examples of Atom/XML responses that can be returned by a DHuS OData service, and in particular upon requests of the Service Document, the Service Matadata Document or request of Products and Collections, two entities very specific to the DHuS.

2.12.1.1 Service Document Responses

The structure and construction rules of the DHuS OData Service Document are depicted by the following example. This example has been reformatted for the sake of readability and the actual responses could vary in term of spaces and carriage returns.

```
<?xml version="1.0" encoding="utf-8"?>

<service xmlns="http://www.w3.org/2007/app"
         xmlns:atom="http://www.w3.org/2005/Atom"
         xml:base="{svc-root}">

  <workspace>
    <atom:title>Default</atom:title>

    <collection href="Products">
      <atom:title>Products</atom:title>
    </collection>

    <collection href="Collections">
      <atom:title>Collections</atom:title>
    </collection>

  </workspace>

</service>
```

2.12.1.2 Service Metadata Document Responses

The OData Service Metadata Document is not compliant to the Atom specifications but has been filed here because it is an XML document. Its structure and construction rules derive from an OData Entity Data Model (EDM) XML language. As a consequence of this fixed format, the request of this Service Metadata Document will not accept any `$format` parameter, even with the however compatible “`xml`” value.

As a reminder, the DHuS OData Service Metadata Document exposes the Entity Data Model of the service including among others, the Entities and Properties that can be queried. This document can be queried as with the following URL:

```
{svc-root}/$metadata
```

The following snippet is extracted from the complete Service Metadata Document exposed by a DHuS OData service. Some sections or XML Namespace declarations have been removed for brevity.

```
<edmx:Edmx Version="1.0">
  <edmx:DataServices>
    <Schema Namespace="DHuS">

      … removed for brevity …

      <EntityType Name="Products" m:HasStream="true">
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.String" Nullable="false"/>
        <Property Name="Name" Type="Edm.String" … />
        <Property Name="ContentType" Type="Edm.String"/>
        <Property Name="ContentLength" Type="Edm.Int64"/>
        <Property Name="ChildrenNumber" Type="Edm.Int64"/>
        <Property Name="Value" Type="Edm.String"/>
        <Property Name="CreationDate" Type="Edm.DateTime" Nullable="false"/>
        <Property Name="IngestionDate" Type="Edm.DateTime" … />
        <Property Name="EvictionDate" Type="Edm.DateTime"/>
        <Property Name="ContentDate" Type="DHuS.TimeRange"/>
        <Property Name="ContentGeometry" Type="Edm.String"/>

        … removed for brevity …

      </EntityType>

      <EntityType Name="Collection">

        … removed for brevity …

      </EntityType>

      <EntityContainer Name="DHuSData" m:IsDefaultEntityContainer="true">

        … removed for brevity …

        <EntitySet Name="Products" EntityType="DHuS.Product"/>
        <EntitySet Name="Collections" EntityType="DHuS.Collection"/>

        … removed for brevity …

      </EntityContainer>

    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

2.12.1.3 Products Responses

The responses containing a single Product entity differ from those containing a collection of Product entities. Generally speaking, the single Product entity is returned as a bare Atom entry element, while for a collection the same Atom entries denoting the Product entities are wrapped into an Atom feed element.

The following snippet is an actual response of a DHuS OData service that returned a single Product entity. Again, this snippet has been reformatted for readability purpose but nothing has been removed from the actual response.

```
<?xml version="1.0" encoding="utf-8"?>

<entry xmlns="http://www.w3.org/2005/Atom" ❶
       xmlns:m="http://schemas.microsoft.com/ado/2007/08/↵
                  dataservices/metadata"
       xmlns:d="http://schemas.microsoft.com/ado/2007/08/↵
                  dataservices"
       xml:base="{svc-root}">

  <id>{svc-root}/Products('6184bfee-37f7-403b-82c0-90b43579192f')</id> ❷

  <title type="text">S1A_IW_GRDM_1SDH_20140820T063911_↵ ❸
                      20140820T063936_002020_001F56_9DAB</title>

  <updated>2014-10-28T15:14:16.909Z</updated>

  <category term="DHuS.Product"
            scheme="http://schemas.microsoft.com/ado/2007/08/↵
                      dataservices/scheme"/>

  <link href="Products('6184bfee-37f7-403b-82c0-90b43579192f')" ❹
        rel="edit" title="Product"/>
  <link href="Products('6184bfee-37f7-403b-82c0-90b43579192f')/$value"
        rel="edit-media" type="application/octet-stream"/>
  <link href="Products('6184bfee-37f7-403b-82c0-90b43579192f')/Nodes"
        rel="http://schemas.microsoft.com/ado/2007/08/↵
               dataservices/related/Nodes" title="Nodes"
        type="application/atom+xml;type=feed"/>
  <link href="Products('6184bfee-37f7-403b-82c0-90b43579192f')/Attributes"
        rel="http://schemas.microsoft.com/ado/2007/08/↵
               dataservices/related/Attributes" title="Attributes"
        type="application/atom+xml;type=feed"/>
```

```
<link href="Products('6184bfee-37f7-403b-82c0-90b43579192f')/Products"
      rel="http://schemas.microsoft.com/ado/2007/08/↵
             dataservices/related/Products" title="Products"
      type="application/atom+xml;type=feed"/>

<content type="application/octet-stream" ❺
         src="Products('6184bfee-37f7-403b-82c0-90b43579192f')/$value"/>

<m:properties> ❻
  <d:Id>6184bfee-37f7-403b-82c0-90b43579192f</d:Id>
  <d:Name>S1A_IW_GRDM_1SDH_20140820T063911↵
          _20140820T063936_002020_001F56_9DAB</d:Name>
  <d:ContentType>application/octet-stream</d:ContentType>
  <d:ContentLength>46379571</d:ContentLength>
  <d:ChildrenNumber>2</d:ChildrenNumber>
  <d:Value m:null="true"/>
  <d:CreationDate>2014-10-23T16:29:51.591</d:CreationDate>
  <d:IngestionDate>2014-10-28T15:14:16.909</d:IngestionDate>
  <d:EvictionDate m:null="true"/>
  <d:ContentDate m:type="DHuS.TimeRange">
    <d:Start>2014-08-20T06:39:11.366</d:Start>
    <d:End>2014-08-20T06:39:36.36</d:End>
  </d:ContentDate>
  <d:ContentGeometry>↵ ❼
    &lt;gml:Polygon srsName="http://www.opengis.net/gml/srs/epsg.xml#4326"
                  xmlns:gml="http://www.opengis.net/gml"&gt;
      &lt;gml:outerBoundaryIs&gt;
        &lt;gml:LinearRing&gt;
          &lt;gml:coordinates&gt;16.869593,79.331299 5.036688,80.201675↵
            7.996001,81.628258 21.289911,80.635994 16.869593,79.331299
          &lt;/gml:coordinates&gt;
        &lt;/gml:LinearRing&gt;
      &lt;/gml:outerBoundaryIs&gt;
    &lt;/gml:Polygon&gt;↵
  </d:ContentGeometry>
  <d:Metalink>↵ ❽
    &lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt;
    &lt;metalink xmlns="urn:ietf:params:xml:ns:metalink"&gt;
      &lt;file name="S1A_IW_GRDM_1SDH_20140820T063911_↵
                  20140820T063936_002020_001F56_9DAB.zip"&gt;
        &lt;url&gt;http://dhus.gael.fr/dhus/odata/v1/↵
          Products('6184bfee-37f7-403b-82c0-90b43579192f')/$value↵
        &lt;/url&gt;
      &lt;/file&gt;
    &lt;/metalink&gt;
  </d:Metalink>
  </m:properties>
</entry>
```

❶ The Atom `entry` element represents the returned Product entity and contains all the definitions and properties of this latter.

❷ The `id` element contains the unique identification of the Product entity. Actually, although the DHuS Products are identified through a UUID supposedly unique, the true key that univocally identifies a Product is the full URI/URL including the address of requested DHuS.

❸ The Atom `title` of the `entry` is actually the `Name` property of the Product entity.

❹ The series of `link` Atom element helps discovering and the traversing of the tree of entities and properties from this Product entity. As an example, one of these `link` elements specifies the paths to the `$value` of the Product, which in turn represents the means for downloading the Product content. Similarly, these `link` elements show the path to the Product Attributes, Nodes and derivative Products entities associated to this Product entity.

❺ The `content` element explicitly states that this Product entity has a content that can be downloaded. This may not be the case for all entities returned by the OData API. For example, the Collection entities are logical and are not supposed to have content, and even more, a downloadable content.

❻ The `properties` element contains all properties associated with this Product entity. The list depends on the Entity Data Model (EDM) exposed by the DHuS OData service. This is not a reference document, but the following document provides high level definitions of the properties that are expected as properties of a Product entity:

Property Name	Type	Description
Id	UUID	A unique identifier of the Product entity. This identifier is supposedly unique by construction but the actual uniqueness can only be verified and guaranteed for a

		single instance of DHuS OData service.

2.12.1.4 Collections Responses

```
<?xml version="1.0" encoding="utf-8"?>

<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:m="http://schemas.microsoft.com/↲
               ado/2007/08/dataservices/metadata"
      xmlns:d="http://schemas.microsoft.com/↲
               ado/2007/08/dataservices"
      xml:base="{svc-root}/">

  <id>{svc-root}/Collections</id>
  <title type="text">Collections</title>
  <updated>2014-11-06T19:28:59.224Z</updated>
  <author><name/></author>
  <link href="Collections" rel="self" title="Collections"/>

  <entry> ❶

    <id>{svc-root}/Collections('my_collection')</id>

    <title type="text">my_collection</title> ❷

    <updated>2014-11-06T19:28:59.224Z</updated>

    <category term="DHuS.Collection" ❸
              scheme="http://schemas.microsoft.com/↲
                      ado/2007/08/dataservices/scheme"/>

    <link href="Collections('my_collection')"
          rel="edit" title="Collection"/>

    <link href="Collections('my_collection')/Products" ❹
          rel="http://schemas.microsoft.com/↲
               ado/2007/08/dataservices/related/Products"
          title="Products"
          type="application/atom+xml;type=feed"/>

    <link href="Collections('my_collection')/Collections" ❺
          rel="http://schemas.microsoft.com/↲
               ado/2007/08/dataservices/related/Collections"
          title="Collections"
          type="application/atom+xml;type=feed"/>

    <content type="application/xml">
      <m:properties>
        <d:Name>my_collection</d:Name>
        <d:Description/>
      </m:properties>
    </content>

  </entry>

  <entry> ❻
    … other entries removed for brevity …
  </entry>

</feed>
```

2.12.1.5 Error Responses

In case of invalid requests or null results including a XML `$format` option or by default, a DHuS may generate an error response with the following form:

```
<error xmlns="http://schemas.microsoft.com/ado/2007/08/↲
                dataservices/metadata">
  <code/>
  <message xml:lang="en-US">↲
    Could not find an entity set or function import for 'Produts'.↲
  </message>
</error>
```

2.12.1.6 Service Document Responses

When the URL equals the service root with no resource path

```
{

  "d": {
    "EntitySets": [
      "Products",
      "Collections"
    ]
  }
}
```

2.12.1.7 Service Metadata Document Responses

The Metadata Document is never returned in JSON format. Forcing the JSON via the `$format` option will result to an error response similar to the following:

```
{
  "error": {
    "code": null,
    "message": {
      "lang": "en-US",
      "value": "System query option '$format' is not compatible↵
        with the return type."
    }
  }
}
```

2.12.1.8 Products Responses

2.12.1.9 Collections Responses

```
{
  "d": {
    "results": [ ❶
      {
        "Collections": {
          "__deferred": {
            "uri": "{svc-root}/Collections('my_collection')/Collections" ❷
          }
        },
        "Products": {
          "__deferred": {
            "uri": "{svc-root}/Collections('my_collection')/Products" ❸
          }
        },
        "Description": "",
        "Name": "my collection", ❹
        "__metadata": {
          "type": "DHuS.Collection", ❺
          "uri": "{svc-root}/Collections('my_collection')",
          "id": "{svc-root}/Collections('my_collection')"
        }
      },
      {
        "Collections": { ❻
          … other collections and results removed for brevity …
      }
    ]
  }
}
```

2.12.1.10 Error Responses-JSON

In case of invalid requests or null results including a JSON `$format` option, a DHuS may generate an error response with the following form:

```
{
  "error": {
    "code": null,
    "message": {
      "lang": "en",
      "value": "Navigation failed (result is null)."
    }
  }
}
```

2.12.2 Error Responses

In case of invalid requests or null results including a `$filter` option, a DHuS may generate an error response with the following form:

```
<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
>
<code/>
<message xml:lang="en-US">
Invalid filter expression: '[filter]'.
</message>
</error>
```

3 Open Search Interface

OpenSearch (Solr) is a set of technologies that allow publishing of search results in a standard and accessible format. OpenSearch is RESTful technology and complementary to the OData. In fact, OpenSearch can be used to complementary serve as the query aspect of OData, which provides a way to access identified or located results and download them. The Data Hub implementation uses the Apache Solr search engine [RD-2].

In OpenSearch, there are four basic parts to the standard:

- A Description Document that is used to describe the search engine
- Search client applications
- OpenSearch response elements
- A result set

In Data Hub the OpenSearch query can be triggered by the following URI:

`<dhus_hostname>:<port>/<path>/search?q=<query>`

where:

`<dhus_hostname>:<port>/<path>` is the Service Root

`search?q=<query>` is the Query

Some functions are available to build the query. Here below the table with the main functions:

Function	Description	Example
Rows	Customize the maximum number of results	q=*&rows=1
Start	Results paging	q=*&start=10

Footprint	**Return results having footprint covering the geographic area (polygon or point)**	**q=footprint:"Intersects(<geographic type>)”**

The Data Hub provides the Open Search interface to perform searches by indexed metadata. For the complete list of indexed metadata see Annex.

In this section, we’ll look at the OpenSearch 1.1 standard, focusing on the description file and some of the response elements.

3.1 OpenSearch Description Document

The Description Document is used to describe the Web interface of the OpenSearch-compliant search engine. The document is written in XML. The document consists of description elements that form the overall Description Document. The description elements are laid out in the OpenSearch 1.1 standard and are summarized here.

Note All XML files start with an XML declaration that specifies, among other things, which version of XML is being used. This is the first line in the XML document and normally looks like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The grammar of the URL template used by OpenSearch 1.1 is defined by the Augmented Backus-Naur Form (ABNF) rules from RFC 2234. A parameter as a variable that will need a value supplied in order for the URL template to work. Anytime there is a parameter in the template, a value must be entered by the search client before the search request can be performed. Parameter names consist of an optional parameter name prefix followed by the local parameter name. If the parameter name prefix is present, then it will be separated from the local parameter name with the “:” character. All parameter names are associated with a parameter namespace. In the case of unqualified parameter names, the local parameter name is implicitly associated with the OpenSearch 1.1 namespace. In the case of fully qualified parameter names, the local parameter name is explicitly associated with an external namespace via the parameter name prefix.

3.1.1 OpenSearchDescription Element

The OpenSearchDescription element is the first entry in the Description Document after the XML declaration statement. Hence, this element is the overall opening and closing tag for the Description Document. The following is what the opening and closing tags would look like:

```
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">

</OpenSearchDescription>
```

The OpenSearchDescription opening and closing tags must appear once in the Description Document and cannot appear more than once. In addition, this tag is a root tag because it has no parents inside which it must operate.

3.1.2 Title Element

The title element allows to assign a name to the search engine the user triggered. It must appear inside the OpenSearchDescription tags and may contain no more than 16 plain-text characters. The name may not include HTML or other markup language. This element must also appear in the Description Document. The entry would look like the following:

```
<title>Sentinel Data Hub search results for: [query] </title>
```

3.1.3 Subtitle Element

The subtitle element describes the result of the search query the user performed. The XML will look like the following:

```
<subtitle>Displaying 1 results. Request done in 0.001 seconds.</subtitle>
```

3.1.4 Updated Element

The `updated element` indicates the UTC date in which the result of the query has been performed. It will look something like the following:

```
<updated>2015-05-27T13:04:41.357Z</updated>
```

3.1.5 Author Element

The `Author` element points to the `name` of the Data Hub instance and it is the same for every product that have been ingested in the same instance. It will look something like the following:

```
<author>
<name>Sentinel-1 Scientific Data Hub</name>
</author>
```

3.1.6 Id Element

The `id` element points to the URL address that clients normally use to enter search queries for the remote index that was generated by the remote search engine. The URL element must appear within the OpenSearchDescription tags and will have several required and optional attributes.

Examples of the URL element query template are as follows:

```
<id>http://example.com/search?q={searchTerms}</id>
```

3.1.7 Query Element

The Query element is the template or actual query that will be performed against the remote index. The usercan specify a specific search request or define a variable to host user-defined keyword query terms.

The Query element should also provide at least one element of role="example" in each OpenSearch Description Document so that search clients can test the search engine. Search engines should include a Query element of role="request" in each search response so that search clients can re-create the current search.

An example of the XML would be the following:

```
<opensearch:Query role="request" searchTerms="*" startPage="1"/>
```

3.1.8 AutoDiscovery of RSS/Atom

RSS and Atom are the only supported formats in Search Server 2008 for result sets. If the remote search index isn’t capable of returning the results in either RSS or Atom, then the user’ ll need to write a custom connector page that will convert the HTML/XHTML results into RSS or Atom in order to display the results in the Federated Results Web part. Some restrictions will apply when the user uses this element in the Description Document.

First, the "type" attribute must contain the value "application/opensearchdescription+xml." Second, the "rel" attribute must contain the value "search." Third, the "href" attribute must contain a URI that resolves to an OpenSearch Description Document.

An example of the XML would look like the following:

```
<link rel="self" type="application/atom+xml"
href="http://192.168.1.149:8080/api/search?q=*&amp;start=0&amp;rows=10"/>
```

3.2 OpenSearch Response Elements

In addition to the description elements, the OpenSearch standard also specifies some response elements, which we will briefly discuss. There are fewer response elements than description elements. In addition, the response elements are really just that: responses to the query elements found in the Description Document. Here an example of some response XML directly from the OpenSearch query **"S1A_S1_GRDM_1SDV_20140607T172812_20140607T172836_000947_000EBD_7543"**

```
<?xml version="1.0" encoding="utf-8"?><feed xmlns:opensearch="http://a9.com/-
/spec/opensearch/1.1/" xmlns="http://www.w3.org/2005/Atom">
<title>Sentinel Data Hub search results for:
S1A_S1_GRDM_1SDV_20140607T172812_20140607T172836_000947_000EBD_7543</title>
<subtitle>Displaying 1 results. Request done in 0.002 seconds.</subtitle>
<updated>2015-05-27T12:06:56.852Z</updated>
<author>
<name>Sentinel Data Hub</name>
</author>
<id>http://192.168.1.149:8080/api/search?q=S1A_S1_GRDM_1SDV_20140607T172812_20140607T1
72836_000947_000EBD_7543</id>
<opensearch:totalResults>1</opensearch:totalResults>
<opensearch:startIndex>0</opensearch:startIndex>
<opensearch:itemsPerPage>10</opensearch:itemsPerPage>
<opensearch:Query role="request"
searchTerms="S1A_S1_GRDM_1SDV_20140607T172812_20140607T172836_000947_000EBD_7543"
startPage="1"/>
<link rel="self" type="application/atom+xml"
href="http://192.168.1.149:8080/api/search?q=S1A_S1_GRDM_1SDV_20140607T172812_20140607
T172836_000947_000EBD_7543&amp;start=0&amp;rows=10"/>
<link rel="first" type="application/atom+xml"
href="http://192.168.1.149:8080/api/search?q=S1A_S1_GRDM_1SDV_20140607T172812_20140607
T172836_000947_000EBD_7543&amp;start=0&amp;rows=10"/>
<link rel="last" type="application/atom+xml"
href="http://192.168.1.149:8080/api/search?q=S1A_S1_GRDM_1SDV_20140607T172812_20140607
T172836_000947_000EBD_7543&amp;start=0&amp;rows=10"/>
<link rel="search" type="application/opensearchdescription+xml"
href="opensearch_description.xml"/>
<entry>
<title>S1A_S1_GRDM_1SDV_20140607T172812_20140607T172836_000947_000EBD_7543</title>
<link href="http://192.168.1.149:8080/odata/v1/Products('877b0c26-9ffd-4046-b260-
34e079afe1b6')/$value"/>
<link rel="alternative" href="http://192.168.1.149:8080/odata/v1/Products('877b0c26-
9ffd-4046-b260-34e079afe1b6')/"/>
<link rel="icon" href="http://192.168.1.149:8080/odata/v1/Products('877b0c26-9ffd-
4046-b260-34e079afe1b6')/Products('Quicklook')/$value"/>
<id>877b0c26-9ffd-4046-b260-34e079afe1b6</id>
<summary>Date: 2014-06-07T17:28:12.508Z, Instrument: SAR-C SAR, Mode: VV VH,
Satellite: Sentinel-1, Size: 32 MB</summary>
```

```
<str name="…
…
 /str>
<bool name="processed">true</bool>
<arr name="collection">
<str>collection</str>
</arr>
</entry>
</feed>
```

The user can see the responses to query elements that were entered into the Description Document (not illustrated here), such as `totalResults, startIndex, Query role, searchTerms, itemsPerPage, startPage and title.`

Other response elements are part of the OpenSearch standard. The following sections outline them briefly.

3.2.1 TotalResults Element

The `totalResults` element indicates the total number of results that will come back from the search engine. Interestingly, the standard indicates that if this element doesn't appear on the result page, then the user should consider that page to be the last page in the result set. The value returned must be a non-negative integer. The default number will equal the offset index number of the last content item on the current page. This element is not required on the result set page, but based on the standard itself. The XML for this element would look like the following:

```
<totalResults>492420</totalResults>
```

3.2.2 StartIndex Element

The `startIndex` element is the number of the first content item in the result set. If this element doesn't appear, then the OpenSearch standard indicates that the current page should be considered as the first page in the result set. The value must be an integer, and its default value equals the indexOffset value in the Description Document. This element is not required in the result set. The XML would look like the following:

```
<startIndex>1</startIndex>
```

3.2.3 ItemsPerPage Element

The `ItemsPerPage` element indicates the number of content items returned on each page of the result set. If this value is not set as one of the response elements, then the number of items that are returned on the first page of the result set will be considered the default number. If the value is set, the number must be a non-negative integer. The XML would look like the following:

```
<itemsPerPage>10</itemsPerPage>
```