Integrating DDF

Table of Contents

License

This document last updated: 2019-12-16.

WARNING	If integrating with a Highly Available Cluster of DDF, see High Availability Guidance.

DDF is structured to enable flexible integration with external clients and into larger component systems.

If integrating with an existing installation of DDF, continue to the following sections on endpoints and data/metadata management.

If a new installation of DDF is required, first see the Managing section for installation and configuration instructions, then return to this section for guidance on connecting external clients.

If you would like to set up a test or demo installation to use while developing an external client, see the Quick Start Tutorial for demo instructions.

For troubleshooting and known issues, see the Release Notes.

1. Endpoints

Federation with DDF is primarily accomplished through Endpoints accessible through http(s) requests and responses.

NOTE	Not all installations will expose all available endpoints. Check with DDF administrator to confirm availability of these endpoints.

1.1. Ingest Endpoints

Ingest is the process of getting data and/or metadata into the DDF catalog framework.

These endpoints are provided by DDF to be used by integrators to ingest content or metadata.

Catalog REST Endpoint

Uses REST to interact with the catalog.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

FTP Endpoint

Ingests files directly into the DDF catalog using the FTP protocol.

1.2. CRUD Endpoints

To perform CRUD (Create, Read, Update, Delete) operations on data or metadata in the catalog, work with one of these endpoints.

Catalog REST Endpoint

Uses REST to interact with the catalog.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

Queries Endpoint

To perform CRUD (Create, Read, Update, Delete) operations on query metacards in the catalog, work with one of these endpoints.

1.3. Query Endpoints

Query data or metadata stored within an instance of DDF using one of these endpoints.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

OpenSearch Endpoint

Sends query parameters and receives search results.

1.4. Content Retrieval Endpoints

To retrieve content from an instance of DDF, use one of these endpoints.

Catalog REST Endpoint

Uses REST to interact with the catalog.

1.5. Pub-Sub Endpoints

These endpoints provide publication and subscription services to allow notifications when certain events happen within DDF.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

1.6. Endpoint Details

1.6.1. Catalog REST Endpoint

The Catalog REST Endpoint allows clients to perform operations on the Catalog using REST, a simple architectural style that performs communication using HTTP.

Bulk operations are not supported: for all RESTful CRUD commands, only one metacard ID is supported

in the URL.

The Catalog REST Endpoint can be used for one or more of these operations on an instance of DDF:

- Ingest metacards or resources into the DDF catalog.
- Retrieve metacards or resources from the catalog.
- Update metacards or resources in the catalog.
- Delete resources and metadata from the catalog.
- Get information about configured sources.

This example metacard can be used to test the integration with DDF.

Example Metacard

```
<?xml version="1.0" encoding="UTF-8"?>
<metacard xmlns="urn:catalog:metacard" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil="http://www.w3.org/2001/SMIL20/"
xmlns:smillang="http://www.w3.org/2001/SMIL20/Language" gml:id=
"3a59483ba44e403a9f0044580343007e">
  <type>ddf.metacard</type>
  <string name="title">
    <value>Test REST Metacard</value>
  </string>
  <string name="description">
    <value>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla scelerisque
cursus mi.</value>
  </string>
</metacard>
```

1.6.1.1. Catalog REST Create Operation Examples

The REST endpoint can be used to upload resources as attachments.

Send a `POST` request with the input to be ingested contained in the HTTP request body to the endpoint.

Create Request URL

```
https://<FQDN>:<PORT>/services/catalog/
```

Example Create Request

```
POST /services/catalog?transform=xml HTTP/1.1
Host: <FQDN>:<PORT>
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
Cache-Control: no-cache

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="parse.resource"; filename=""
Content-Type:


------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="parse.metadata"; filename=""
Content-Type:


------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

The `create` and `update` methods both support the multipart mime format. If only a single attachment exists, it will be interpreted as a resource to be parsed, which will result in a metacard and resource being stored in the system.

If multiple attachments exist, then the REST endpoint will assume that one attachment is the actual resource (attachment should be named `parse.resource`) and the other attachments are overrides of metacard attributes (attachment names should follow metacard attribute names). In the case of the metadata attribute, it is possible to also have the system transform that metadata and use the results of that to override the metacard that would be generated from the resource (attachment should be named `parse.metadata`).

Create Success

If the ingest is successful, a status of `201 Created` will be returned, along with the Metacard ID in the header of the response.

NOTE

Request with Non-XML Data

If a request with non-XML data is sent to the Catalog REST endpoint, the metacard will be created but the resource will be stored in the `metadata` field. This could affect discoverability.

If content or metadata is not ingested successfully, check for these error messages.

Table 1. Create Error Responses

Status Code	Error Message	Possible Causes
`400 Bad Request`	`<pre>Error while storing entry in catalog: </pre>`	*Malformed XML Response*: If the XML being ingested has formatting errors.
		Request with Unknown Schema: If ingest is attempted with a schema that is unknown, unsupported, or not configured by the endpoint, DDF creates a generic resource metacard with the provided XML as content for the `metadata` XML field in the metacard.

1.6.1.2. Catalog REST Read Operation Examples

The `read` operation can be used to retrieve metadata in different formats.

1. Send a `GET` request to the endpoint.
2. Optionally add a `transform` query parameter to the end of the URL with the transformer to be used (such as `transform=kml`). By default, the response body will include the XML representation of the Metacard.

Read Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>
```

If successful, a status of `200 OK` will be returned, along with the content of the metacard requested.

Read Success Response Example

```
<metacard xmlns="urn:catalog:metacard" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil="http://www.w3.org/2001/SMIL20/"
xmlns:smillang="http://www.w3.org/2001/SMIL20/Language" gml:id="<METACARD_ID>">
    <type>ddf.metacard</type>
    <source>ddf.distribution</source>
    <string name="title">
        <value>Test REST Metacard</value>
    </string>
    <string name="point-of-contact">
        <value>email@example.com</value>
    </string>
    <dateTime name="metacard.created">
        <value>2019-12-16</value>
    </dateTime>
    <dateTime name="effective">
        <value>2019-12-16</value>
    </dateTime>
    <dateTime name="modified">
        <value>2019-12-16</value>
    </dateTime>
    <dateTime name="created">
        <value>2019-12-16</value>
    </dateTime>
    <string name="description">
        <value>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla scelerisque
cursus mi.</value>
    </string>
    <string name="metacard-tags">
        <value>resource</value>
        <value>VALID</value>
    </string>
    <dateTime name="metacard.modified">
        <value>2019-12-16</value>
    </dateTime>
</metacard>
```

- To receive metadata in an alternate format, add a transformer to the request URL.

Metacard Transform Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>?transform=<TRANSFORMER_ID>
```

Metacard Transform Response (`transform=geojson`)


```
{
    "geometry": null,
    "type": "Feature",
    "properties": {
        "effective": "2019-12-16",
        "point-of-contact": "email@example.com",
        "created": "2019-12-16",
        "metacard.modified": "2019-12-16",
        "metacard-tags": [
            "resource",
            "VALID"
        ],
        "modified": "2019-12-16",
        "description": "Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.",
        "id": "3a59483ba44e403a9f0044580343007e",
        "metacard-type": "ddf.metacard",
        "title": "Test REST Metacard",
        "source-id": "ddf.distribution",
        "metacard.created": "2019-12-16"
    }
}
```


To retrieve a metacard from a specific federated source, add `sources/<SOURCE_ID>` to the URL.

Federated Read Request URL

```
https://<FQDN>:<PORT>/services/catalog/sources/<sourceId>/<metacardId>?transform=<TRANSFO
RMER_ID>
```

To retrieve the resource associated with a metacard, use the `resource` transformer with the `GET` request.

Retrieve Resource Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>?transform=resource
```

See Metacard Transformers for details on metacard transformers.

Read Error Response Examples

If the metacard or resource is not returned successfully, check for these errors.

Table 2. Read Error Responses

Status Code	Error Message	Possible Causes
`404 Not Found`	`<pre>Unable to retrieve requested metacard.</pre>`	Invalid Metacard ID
`500 Server Error`	`<pre>Unknown error occurred while processing request.</pre>`	Transformer is invalid, unsupported, or not configured.
	`<pre>Unable to transform Metacard. Try different transformer: </pre>`	Metacard does not have an associated resource (is metadata only).
	`<pre>READ failed due to unexpected exception: </pre>`	Invalid source ID, or source unavailable.

1.6.1.3. Catalog Rest Update Operation Examples

To update the metadata for a metacard, send a `PUT` request with the ID of the Metacard to be updated appended to the end of the URL and the updated metadata is contained in the HTTP body.

Optionally, specify the transformer to use when parsing an override of a metadata attribute.

Update Request URL

```
https://<FQDN>:<PORT>/<metacardId>?transform=<input transformer>
```

Table 3. Update Error Response Examples

Status Code	Error Message	Possible Causes
`400 Bad Request`	`<pre>Error cataloging updated metadata: </pre>`	Invalid metacard ID.
`500 Server Error`	`<pre>Error cataloging updated metadata: </pre>`	Invalid transformer ID.

1.6.1.4. Catalog REST Delete Operation Examples

To delete a metacard, send a `DELETE` request with the metacard ID to be deleted appended to the end of the URL.

Delete Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>
```

Table 4. Delete Error Response Examples

Status Code	Error Message	Possible Causes
`400 Bad Request`	`<pre>Error deleting entry from catalog: </pre>`	Invalid metacard ID.

1.6.1.5. Catalog REST Sources Operation Examples

To retrieve information about federated sources, including `sourceId`, `availability`, `contentTypes`,and `version`, send a `GET` request to the endpoint.

Sources Response URL

```
https://<FQDN>:<PORT>//sources/
```

Sources Response Example

```
[
   {
      "id" : "DDF-OS",
      "available" : true,
      "contentTypes" :
         [
         ],
      "version" : "2.21.2"
   },
   {
      "id" : "ddf.distribution",
      "available" : true,
      "contentTypes" :
         [
         ],
      "version" : "2.21.2"
   }
]
```

Table 5. Sources Error Responses

Status Code	Error Message	Possible Causes
403	Problem accessing /ErrorServlet. Reason: <pre> Forbidden</pre> `	Connection error or service unavailable.

1.6.2. CSW Endpoint

The CSW endpoint enables a client to search collections of descriptive information (metadata) about geospatial data and services.

The CSW endpoint supports metadata operations only.

For more information about the Catalogue Services for Web (CSW) standard.

The CSW Endpoint can be used for one or more of these operations on an instance of DDF:

- Ingest metadata into the DDF catalog.
- Read metacards from the catalog.
- Update metadata in the catalog.
- Publish and/or subscribe to catalog events.
- Delete metadata from the catalog.
- Get capabilities of the catalog and the URLs used to access.

NOTE	*Sample Responses May Not Match Actual Responses* Actual responses may vary from these samples, depending on your configuration. Send a GET or POST request to obtain an accurate response.

1.6.2.1. CSW Endpoint Create Examples

Metacards are ingested into the catalog via the `Insert` sub-operation.

The schema of the record needs to conform to a schema of the information model that the catalog supports.

Send a `POST` request to the CSW endpoint URL.

CSW Endpoint Ingest URL

```
https://<FQDN>:<PORT>/services/csw
```

Include the metadata to ingest within a `csw:Insert` block in the body of the request.

Sample XML Transaction `Insert` Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    verboseResponse="true"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Insert typeName="csw:Record">
        <csw:Record
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dct="http://purl.org/dc/terms/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <dc:identifier></dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
            <dc:subject>Hydrography--Dictionaries</dc:subject>
            <dc:format>application/pdf</dc:format>
            <dc:date>2019-12-16</dc:date>
            <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.</dct:abstract>
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
                <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
                <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:Record>
    </csw:Insert>
</csw:Transaction>
```

To specify the document type being ingested and select the appropriate input transformer, use the `typeName` attribute in the `csw:Insert` element

```
<csw:Insert typeName="xml">
```

To receive a copy of the metacard in the response, specify `verboseResponse="true"` in the `csw:Transaction`. The `InsertResult` element of the response will hold the metacard information added to the catalog.

```
<csw:Transaction service="CSW" version="2.0.2" verboseResponse="true" [...]
```

Sample XML Transformer Insert

```
<csw:Transaction service="CSW" version="2.0.2" verboseResponse="true" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2">
  <csw:Insert typeName="xml">
    <metacard xmlns="urn:catalog:metacard" xmlns:ns2="http://www.opengis.net/gml"
          xmlns:ns3="http://www.w3.org/1999/xlink" xmlns:ns4=
"http://www.w3.org/2001/SMIL20/"
          xmlns:ns5="http://www.w3.org/2001/SMIL20/Language">
        <type>ddf.metacard</type>
        <string name="title">
            <value>PlainXml near</value>
        </string>
    </metacard>
  </csw:Insert>
</csw:Transaction>
```

A successful ingest will return a status of `200 OK` and `csw:TransactionResponse`.

Sample XML Transaction `Insert` Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:ns3="http://www.w3.org/1999/xlink"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:ns5="http://www.w3.org/2001/SMIL20/"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:ows="http://www.opengis.net/ows"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                         version="2.0.2"
                         ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:TransactionSummary>
        <csw:totalInserted>1</csw:totalInserted>
        <csw:totalUpdated>0</csw:totalUpdated>
        <csw:totalDeleted>0</csw:totalDeleted>
    </csw:TransactionSummary>
    <csw:InsertResult>
        <csw:BriefRecord>
            <dc:identifier><METACARD ID</dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
            <ows:BoundingBox crs="EPSG:4326">
                <ows:LowerCorner>-6.171 44.792</ows:LowerCorner>
                <ows:UpperCorner>-2.228 51.126</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:BriefRecord>
    </csw:InsertResult>
</csw:TransactionResponse>
```

Table 6. Create Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	`ExceptionText` with description of error.	XML error. Check for formatting errors in record.
		Schema error. Verify metadata is compliant with defined schema.

1.6.2.2. CSW Endpoint Query Examples

To query through the CSW Enpoint, send a `POST` request to the CSW endpoint.

CSW Endpoint Query URL

```
https://<FQDN>:<PORT>/services/csw
```

Within the body of the request, include a `GetRecords` operation to define the query. Define the service and version to use (CSW, 2.0.2). The output format must be `application/xml`. Specify the output schema. (To get a list of supported schemas, send a Get Capabilities request to the CSW endpoint.)

GetRecords Syntax

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
```

Include the query within the `GetRecords` request. Optionally, set the `ElementSetName` to determine how much detail to return.

- Brief: the least possible detail.
- Summary: (Default)
- Full: All metadata elements for the record(s).

Within the `Constraint` element, define the query as an OSG or CQL filter.

```
<Query typeNames="Record">
    <ElementSetName>summary</ElementSetName>
    <Constraint version="1.1.0">
        <ogc:Filter>
            <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                <ogc:PropertyName>AnyText</ogc:PropertyName>
                <ogc:Literal>%</ogc:Literal>
            </ogc:PropertyIsLike>
        </ogc:Filter>
    </Constraint>
</Query>
```

```
<Query typeNames="Record">
    <ElementSetName>summary</ElementSetName>
    <Constraint version="2.0.0">
        <ogc:CqlText>
           "AnyText" = '%'
        </ogc:CqlText>
    </csw:Constraint>
</Query>
```

`GetRecords` *XML Request Example*

```
<?xml version="1.0" ?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <Query typeNames="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                    <ogc:PropertyName>AnyText</ogc:PropertyName>
                    <ogc:Literal>%</ogc:Literal>
                </ogc:PropertyIsLike>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

`GetRecords` *Sample Response (`application/xml`)*

```
<?xml version='1.0' encoding='UTF-8'?>
<csw:GetRecordsResponse xmlns:dct="http://purl.org/dc/terms/"
                        xmlns:xml="http://www.w3.org/XML/1998/namespace"
                        xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                        xmlns:ows="http://www.opengis.net/ows"
                        xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0.2">
  <csw:SearchStatus timestamp="2019-12-16"/>
  <csw:SearchResults numberOfRecordsMatched="1" numberOfRecordsReturned="1" nextRecord="
0" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">
    <csw:Record xmlns:ows="http://www.opengis.net/ows"
                xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                xmlns:dc="http://purl.org/dc/elements/1.1/"
                xmlns:dct="http://purl.org/dc/terms/"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <dc:identifier/>
        <dc:title>Aliquam fermentum purus quis arcu</dc:title>
        <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
        <dc:subject>Hydrography--Dictionaries</dc:subject>
        <dc:format>application/pdf</dc:format>
        <dc:date>2019-12-16</dc:date>
        <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.</dct:abstract>
        <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
            <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
            <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
        </ows:BoundingBox>
    </csw:Record>
  </csw:SearchResults>
</csw:GetRecordsResponse>
```

Querying a Specific Source with the CSW Endpoint

To query a `Specific Source`, specify a query for a `source-id`. To find a valid `source-id` , send a Get Capabilities request. Configured sources will be listed in the `FederatedCatalogs` section of the response.

NOTE	The `DistributedSearch` element must be specific with a `hopCount` greater than 1 to identify it as a federated query, otherwise the `source-id`'s will be ignored.

Querying a Specific Source Sample Request

```
<?xml version="1.0" ?>
<csw:GetRecords resultType="results"
    outputFormat="application/xml"
    outputSchema="urn:catalog:metacard"
    startPosition="1"
    maxRecords="10"
    service="CSW"
    version="2.0.2"
    xmlns:ns2="http://www.opengis.net/ogc" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:ns4="http://www.w3.org/1999/xlink"
xmlns:ns3="http://www.opengis.net/gml" xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
xmlns:ns5="http://www.opengis.net/ows" xmlns:ns6="http://purl.org/dc/elements/1.1/"
xmlns:ns7="http://purl.org/dc/terms/" xmlns:ns8="http://www.w3.org/2001/SMIL20/">
  <csw:DistributedSearch hopCount="2" />
    <ns10:Query typeNames="csw:Record" xmlns="" xmlns:ns10=
"http://www.opengis.net/cat/csw/2.0.2">
        <ns10:ElementSetName>full</ns10:ElementSetName>
        <ns10:Constraint version="1.1.0">
            <ns2:Filter>
              <ns2:And>
                <ns2:PropertyIsEqualTo wildCard="*" singleChar="#" escapeChar="!">
                  <ns2:PropertyName>source-id</ns2:PropertyName>
                  <ns2:Literal>Source1</ns2:Literal>
                </ns2:PropertyIsEqualTo>_
                <ns2:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!">
                  <ns2:PropertyName>title</ns2:PropertyName>
                    <ns2:Literal>*</ns2:Literal>
                </ns2:PropertyIsLike>
              </ns2:And>
            </ns2:Filter>
        </ns10:Constraint>
    </ns10:Query>
</csw:GetRecords>
```

Querying for GMD Output Schema

To receive a response to a `GetRecords` query that conforms to the GMD specification, set the Namespace(xmlns),`outputschema`, and `typeName` elements for GML schema.

GML Output Schema Sample Request

```
<?xml version="1.0" ?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:gmd="http://www.isotc211.org/2005/gmd"
        xmlns:gml="http://www.opengis.net/gml"
        service="CSW"
        version="2.0.2"
        maxRecords="8"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.isotc211.org/2005/gmd"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <Query typeNames="gmd:MD_Metadata">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                    <ogc:PropertyName>apiso:Title</ogc:PropertyName>
                    <ogc:Literal>%</ogc:Literal>
                </ogc:PropertyIsLike>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

Querying by UTM Coordinates

UTM coordinates can be used when making a CSW GetRecords request using an `ogc:Filter`. UTM coordinates should use `EPSG:326XX`as the `srsName` where `XX` is the zone within the northern hemisphere. UTM coordinates should use `EPSG:327XX` as the `srsName` where `XX` is the zone within the southern hemisphere.

NOTE	UTM coordinates are only supported with requests providing an `ogc:Filter`, but not with CQL as there isn't a way to specify the UTM `srsName` in CQL.

UTM Northern Hemisphere Zone 36 Sample Request

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:gml="http://www.opengis.net/gml"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <Query typeNames="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:Intersects>
                    <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                    <gml:Envelope srsName="EPSG:32636">
                        <gml:lowerCorner>171070 1106907</gml:lowerCorner>
                        <gml:upperCorner>225928 1106910</gml:upperCorner>
                    </gml:Envelope>
                </ogc:Intersects>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

Querying by Metacard ID

To locate a record by Metacard ID, send a `POST` request with a `GetRecordById` element specifying the ID.

`GetRecordById` *Request Example*

```
<GetRecordById xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  service="CSW"
  version="2.0.2"
  outputFormat="application/xml"
  outputSchema="http://www.opengis.net/cat/csw/2.0.2"
  xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
../../../csw/2.0.2/CSW-discovery.xsd">
 <ElementSetName>full</ElementSetName>
 <Id><em><METACARD-ID></em></Id>
</GetRecordById>
```

Table 7. CSW Record to Metacard Mapping

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dc:title	title	1-n	1-n	0-n
dc:creator				0-n
dc:subject			0-n	0-n
dc:description				0-n
dc:publisher				0-n
dc:contributor				0-n
dc:date	modified			0-n
dc:type	metadata-content-type	0-1	0-1	0-n
dc:format			0-n	0-n
dc:identifier	id	1-n	1-n	0-n
dc:source	source-id			0-n
dc:language				0-n
dc:relation			0-n	0-n
dc:coverage				0-n
dc:rights				0-n
dct:abstract	description		0-n	0-n
dct:accessRights				0-n
dct:alternative	title			0-n
dct:audience				0-n

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dct:available				0-n
dct:bibliographicCitation	id			0-n
dct:conformsTo				0-n
dct:created	created			0-n
dct:dateAccepted	effective			0-n
dct:Copyrighted	effective			0-n
dct:dateSubmitted	modified			0-n
dct:educationLevel				0-n
dct:extent				0-n
dct:hasFormat				0-n
dct:hasPart				0-n
dct:hasVersion				0-n
dct:isFormatOf				0-n
dct:isPartOf				0-n
dct:isReferencedBy				0-n
dct:isReplacedBy				0-n
dct:isRequiredBy				0-n
dct:issued	modified			0-n
dct:isVersionOf				0-n
dct:license				0-n
dct:mediator				0-n
dct:medium				0-n
dct:modified	modified		0-n	0-n
dct:provenance				0-n
dct:references				0-n
dct:replaces				0-n
dct:requires				0-n
dct:rightsHolder				0-n

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dct:spatial	location		0-n	0-n
dct:tableOfContents				0-n
dct:temporal	effective + " - " + expiration			0-n
dct:valid	expiration			0-n
ows:BoundingBox		0-n	0-n	0-n

Table 8. Query Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	<ows:ExceptionText>ddf.catalog.util.impl.CatalogQueryException: ddf.catalog.federation.FederationException: SiteNames could not be resolved due to invalid site names, none of the sites were available, or the current subject doesn't have permission to access the sites.</ows:ExceptionText>	A query to a specific source has specified a source that is unavailable.
200 OK	<csw:SearchResults numberOfRecordsMatched="0" numberOfRecordsReturned="0" nextRecord="0"	No results found for query. Verify input.

1.6.2.3. CSW Endpoint Update Examples

The CSW Endpoint can edit the metadata attributes of a metacard.

Send a POST request to the CSW Endpoint URL:

CSW Endpoint Update URL

```
https://<FDQN>:<PORT>/services/csw
```

Replace the <METACARD-ID> value with the metacard id being updated, and edit any properties within the csw:Record.

CSW Update Record Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Update>
        <csw:Record
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dct="http://purl.org/dc/terms/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <dc:identifier><METACARD-ID></dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
            <dc:subject>Hydrography--Dictionaries</dc:subject>
            <dc:format>application/pdf</dc:format>
            <dc:date>2019-12-16</dc:date>
            <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.</dct:abstract>
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
                <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
                <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:Record>
    </csw:Update>
</csw:Transaction>
```

CSW Update Record Sample Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ows="http://www.opengis.net/ows"
                         xmlns:ns2="http://www.w3.org/1999/xlink"
                         xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:ns6="http://www.w3.org/2001/SMIL20/"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance" version=
"2.0.2"
                         ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:TransactionSummary>
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>1</csw:totalUpdated>
        <csw:totalDeleted>0</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

Updating Individual Attributes

Within the `csw:Transaction` element, use the `csw:RecordProperty` to update individual metacard attributes.

Use the `Name` element to specify the name of the record property to be updated and set the `Value` element to the value to update in the record. The values in the `Update` will completely replace those that are already in the record.

```
<csw:RecordProperty>
    <csw:Name>title</csw:Name>
    <csw:Value>Updated Title</csw:Value>
</csw:RecordProperty>
```

Removing Attributes

To remove a non-required attribute, send the `csw:Name` without a `csw:Value`.

```
<csw:RecordProperty>
    <csw:Name>title</csw:Name>
</csw:RecordProperty>
```

Required attributes are set to a default value if no `Value` element is provided.

Table 9. `RecordProperty` Default Values

Property	Default Value
`metadata-content-type`	Resource
`created`	*current time*
`modified`	*current time*
`effective`	*current time*
`metadata-content-type-version`	*myVersion*
`metacard.created`	*current time*
`metacard.modified`	*current time*
`metacard-tags`	resource, VALID
`point-of-contact`	system@localhost
`title`	*current time*

Use a `csw:Constraint` to specify the metacard ID. The constraint can be an OGC Filter or a CQL query.

```
<csw:Constraint version="2.0.0">
    <ogc:Filter>
        <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>id</ogc:PropertyName>
            <ogc:Literal><METACARD-ID></ogc:Literal>
        </ogc:PropertyIsEqualTo>
    </ogc:Filter>
</csw:Constraint>
```

```
<csw:Constraint version="2.0.0">
    <ogc:CqlText>
        "id" = '<METACARD-ID>'
    </ogc:CqlText>
</csw:Constraint>
```

WARNING These filters can search on any arbitrary query criteria, but take care to only affect desired records.

Sample XML Transaction `Update` Request with OGC filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Update>
        <csw:RecordProperty>
            <csw:Name>title</csw:Name>
            <csw:Value>Updated Title</csw:Value>
        </csw:RecordProperty>
            <csw:Constraint version="2.0.0">
                <ogc:Filter>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>id</ogc:PropertyName>
                        <ogc:Literal><METACARD-ID></ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                </ogc:Filter>
            </csw:Constraint>
    </csw:Update>
</csw:Transaction>
```

Sample XML Transaction `Update` Request with CQL filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Update>
        <csw:RecordProperty>
            <csw:Name>title</csw:Name>
            <csw:Value>Updated Title</csw:Value>
        </csw:RecordProperty>
        <csw:RecordProperty>
        </csw:RecordProperty>
        <csw:Constraint version="2.0.0">
            <ogc:CqlText>
                "id" = '<METACARD-ID>'
            </ogc:CqlText>
        </csw:Constraint>
    </csw:Update>
</csw:Transaction>
```

Sample XML Transaction Update Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:ns3="http://www.w3.org/1999/xlink"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:ns5="http://www.w3.org/2001/SMIL20/"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:ows="http://www.opengis.net/ows"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                         ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd"
                         version="2.0.2">
    <csw:TransactionSummary>
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>1</csw:totalUpdated>
        <csw:totalDeleted>0</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

Table 10. Update Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	`<ows:ExceptionText>Unable to update record(s).</ows:ExceptionText>`	XML or CSW schema error. Verify input.
200 OK	`<csw:totalUpdated>0</csw:totalUpdated>`	No records were updated. Verify metacard id or search parameters.

1.6.2.4. CSW Endpoint Publication/Subscription Examples

The subscription `GetRecords` operation is very similar to the `GetRecords` operation used to search the catalog but it subscribes to a search and sends events to a `ResponseHandler` endpoint as metacards are ingested matching the `GetRecords` request used in the subscription. The `ResponseHandler` must use the https protocol and receive a HEAD request to poll for availability and POST/PUT/DELETE requests for creation, updates, and deletions. The response to a `GetRecords` request on the subscription url will be an acknowledgement containing the original `GetRecords` request and a `requestId`. The client will be assigned a `requestId` (URN).

A Subscription listens for events from federated sources if the `DistributedSearch` element is present and the catalog is a member of a federation.

Adding a Subscription

Send a `POST` request to the CSW endpoint.

CSW Add Subscription Sample URL

```
https://<FQDN>:<PORT>/services/csw/subscription
```

Subscription `GetRecords` XML Request

```
<?xml version="1.0" ?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <ResponseHandler>https://some.ddf/services/csw/subscription/event</ResponseHandler>
    <Query typeNames="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                    <ogc:PropertyName>xml</ogc:PropertyName>
                    <ogc:Literal>%</ogc:Literal>
                </ogc:PropertyIsLike>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

Updating a Subscription

To update an existing subscription, send a `PUT` request with the `requestid` URN appended to the url.

CSW Endpoint Subscription Update URL

```
https://{FQDN}:{PORT}/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

Subscription `GetRecords` *XML Response*

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2019-12-16T18:49:45" xmlns=
"http://www.opengis.net/cat/csw/2.0.2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
  <EchoedRequest>
    <GetRecords
            requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
            service="CSW"
            version="2.0.2"
            maxRecords="4"
            startPosition="1"
            resultType="results"
            outputFormat="application/xml"
            outputSchema="urn:catalog:metacard">
        <ResponseHandler>
https://some.ddf/services/csw/subscription/event</ResponseHandler>
        <Query typeNames="Record">
            <ElementSetName>summary</ElementSetName>
            <Constraint version="1.1.0">
                <ogc:Filter>
                    <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                        <ogc:PropertyName>xml</ogc:PropertyName>
                        <ogc:Literal>%</ogc:Literal>
                    </ogc:PropertyIsLike>
                </ogc:Filter>
            </Constraint>
        </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

Subscription `GetRecords` Event Sample Response

```
<csw:GetRecordsResponse version="2.0.2" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows" xmlns:xs=
"http://www.w3.org/2001/XMLSchema"  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <csw:SearchStatus timestamp="2014-02-19T15:33:44.602-05:00"/>
    <csw:SearchResults numberOfRecordsMatched="1" numberOfRecordsReturned="1" nextRecord
="5" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">
      <csw:SummaryRecord>
        <dc:identifier>f45415884c11409497e22db8303fe8c6</dc:identifier>
        <dc:title>Product10</dc:title>
        <dc:type>pdf</dc:type>
        <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>
        <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
          <ows:LowerCorner>20.0 10.0</ows:LowerCorner>
          <ows:UpperCorner>20.0 10.0</ows:UpperCorner>
        </ows:BoundingBox>
      </csw:SummaryRecord>
    </csw:SearchResults>
  </csw:GetRecordsResponse>
```

Retrieving an Active Subscription

To retrieve an active subscription, send a `GET` request with the `requestid` URN appended to the url.

Retrieve.

```
https://<FQDN>:<PORT>/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

Subscription `HTTP GET` Sample Response

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2019-12-16T18:49:45" xmlns=
"http://www.opengis.net/cat/csw/2.0.2"
                                                 xmlns:ogc="http://www.opengis.net/ogc"
                                                 xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
                                                 xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-discovery.xsd">
  <EchoedRequest>
    <GetRecords
            requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
            service="CSW"
            version="2.0.2"
            maxRecords="4"
            startPosition="1"
            resultType="results"
            outputFormat="application/xml"
            outputSchema="urn:catalog:metacard">
        <ResponseHandler>
https://some.ddf/services/csw/subscription/event</ResponseHandler>
        <Query typeNames="Record">
            <ElementSetName>summary</ElementSetName>
            <Constraint version="1.1.0">
                <ogc:Filter>
                    <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                        <ogc:PropertyName>xml</ogc:PropertyName>
                        <ogc:Literal>%</ogc:Literal>
                    </ogc:PropertyIsLike>
                </ogc:Filter>
            </Constraint>
        </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

Deleting a Subscription

To delete a subscription, send a `DELETE` request with the `requestid` URN appended to the url.

Delete Subscription Sample URL

```
https://<FQDN>:<PORT>/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

1.6.2.5. CSW Endpoint Delete Examples

To delete metacards via the CSW Endpoint, send a `POST` request with a `csw:Delete` to the CSW Endpoint URL.

```
https://<FQDN>:<PORT>/services/csw
```

Define the records to delete with the `csw:Constraint` field. The constraint can be either an OGC or CQL filter.

Sample XML Transaction `Delete` Request with OGC filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Delete typeName="csw:Record" handle="something">
        <csw:Constraint version="2.0.0">
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                   <ogc:PropertyName>id</ogc:PropertyName>
                   <ogc:Literal><em><METACARD-ID></em></ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
        </csw:Constraint>
    </csw:Delete>
</csw:Transaction>
```

Sample XML Transaction `Delete` Request with CQL filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Delete typeName="csw:Record" handle="something">
        <csw:Constraint version="2.0.0">
            <ogc:CqlText>
                 "id" = '<em><METACARD-ID></em>'
            </ogc:CqlText>
        </csw:Constraint>
    </csw:Delete>
</csw:Transaction>
```

Sample XML Transaction Delete Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ows="http://www.opengis.net/ows"
                         xmlns:ns2="http://www.w3.org/1999/xlink"
                         xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:ns8="http://www.w3.org/2001/SMIL20/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                         version="2.0.2" ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:TransactionSummary>
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>0</csw:totalUpdated>
        <csw:totalDeleted>1</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

Table 11. Delete Error Response Examples

Status Code	Error Message	Possible Causes
`200 OK`	`<csw:totalDeleted>0</csw:totalDeleted>`	No records matched filter criteria. Verify metacard ID.
`400 Bad Request	`<ows:Exception>` with details of error.	XML or CSW formatting error. Verify request.

1.6.2.6. CSW Endpoint Get Capabilities Examples

The `GetCapabilities` operation describes the operations the catalog supports and the URLs used to access those operations. The CSW endpoint supports both `HTTP GET` and `HTTP POST` requests for the `GetCapabilities` operation. The response to either request will always be a `csw:Capabilities` XML document. This XML document is defined by the CSW-Discovery XML Schema.

CSW Endpoint `GetCapabilities` URL for GET request

```
https://<FQDN>:<PORT>/services/csw?service=CSW&version=2.0.2&request=GetCapabilities
```

Alternatively, send a `POST` request to the root CSW endpoint URL.

CSW Endpoint `GetCapabilities` URL for GET request

```
$https://<FQDN>:<PORT>/services/csw
```

Include an XML message body with a `GetCapabilities` element.

`GetCapabilities` *Sample Request*

```
<?xml version="1.0" ?>
<csw:GetCapabilities
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
  service="CSW"
  version="2.0.2" >
</csw:GetCapabilities>
```

`GetCapabilities` *Sample Response (`application/xml`)*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Capabilities xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml=
"http://www.opengis.net/gml" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:ns6=
"http://www.w3.org/2001/SMIL20/" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dct=
"http://purl.org/dc/terms/" xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance" version="2.0.2"
ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
    <ows:ServiceIdentification>
        <ows:Title>Catalog Service for the Web</ows:Title>
        <ows:Abstract>DDF CSW Endpoint</ows:Abstract>
        <ows:ServiceType>CSW</ows:ServiceType>
        <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>DDF</ows:ProviderName>
        <ows:ProviderSite/>
        <ows:ServiceContact/>
    </ows:ServiceProvider>
    <ows:OperationsMetadata>
        <ows:Operation name="GetCapabilities">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
```

```
            </ows:DCP>
            <ows:Parameter name="sections">
                <ows:Value>ServiceIdentification</ows:Value>
                <ows:Value>ServiceProvider</ows:Value>
                <ows:Value>OperationsMetadata</ows:Value>
                <ows:Value>Filter_Capabilities</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Operation name="DescribeRecord">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="typeName">
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputFormat">
                <ows:Value>application/xml</ows:Value>
                <ows:Value>application/json</ows:Value>
                <ows:Value>application/atom+xml</ows:Value>
                <ows:Value>text/xml</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="schemaLanguage">
                <ows:Value>http://www.w3.org/XMLSchema</ows:Value>
                <ows:Value>http://www.w3.org/XML/Schema</ows:Value>
                <ows:Value>http://www.w3.org/2001/XMLSchema</ows:Value>
                <ows:Value>http://www.w3.org/TR/xmlschema-1/</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Operation name="GetRecords">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="ResultType">
```

```
                <ows:Value>hits</ows:Value>
                <ows:Value>results</ows:Value>
                <ows:Value>validate</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputFormat">
                <ows:Value>application/xml</ows:Value>
                <ows:Value>application/json</ows:Value>
                <ows:Value>application/atom+xml</ows:Value>
                <ows:Value>text/xml</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputSchema">
                <ows:Value>urn:catalog:metacard</ows:Value>
                <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
                <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="typeNames">
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ConstraintLanguage">
                <ows:Value>Filter</ows:Value>
                <ows:Value>CQL_Text</ows:Value>
            </ows:Parameter>
            <ows:Constraint name="FederatedCatalogs">
                <ows:Value>Source1</ows:Value>
                <ows:Value>Source2</ows:Value>
            </ows:Constraint>
        </ows:Operation>
        <ows:Operation name="GetRecordById">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="OutputSchema">
                <ows:Value>urn:catalog:metacard</ows:Value>
                <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
                <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
                <ows:Value>http://www.iana.org/assignments/media-types/application/octet-
stream</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputFormat">
                <ows:Value>application/xml</ows:Value>
```

```
                <ows:Value>application/json</ows:Value>
                <ows:Value>application/atom+xml</ows:Value>
                <ows:Value>text/xml</ows:Value>
                <ows:Value>application/octet-stream</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ResultType">
                <ows:Value>hits</ows:Value>
                <ows:Value>results</ows:Value>
                <ows:Value>validate</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ElementSetName">
                <ows:Value>brief</ows:Value>
                <ows:Value>summary</ows:Value>
                <ows:Value>full</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Operation name="Transaction">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="typeNames">
                <ows:Value>xml</ows:Value>
                <ows:Value>appxml</ows:Value>
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
                <ows:Value>tika</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ConstraintLanguage">
                <ows:Value>Filter</ows:Value>
                <ows:Value>CQL_Text</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Parameter name="service">
            <ows:Value>CSW</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="version">
            <ows:Value>2.0.2</ows:Value>
        </ows:Parameter>
    </ows:OperationsMetadata>
    <ogc:Filter_Capabilities>
        <ogc:Spatial_Capabilities>
            <ogc:GeometryOperands>
```

```
                <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
                <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
                <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
            </ogc:GeometryOperands>
            <ogc:SpatialOperators>
                <ogc:SpatialOperator name="BBOX"/>
                <ogc:SpatialOperator name="Beyond"/>
                <ogc:SpatialOperator name="Contains"/>
                <ogc:SpatialOperator name="Crosses"/>
                <ogc:SpatialOperator name="Disjoint"/>
                <ogc:SpatialOperator name="DWithin"/>
                <ogc:SpatialOperator name="Intersects"/>
                <ogc:SpatialOperator name="Overlaps"/>
                <ogc:SpatialOperator name="Touches"/>
                <ogc:SpatialOperator name="Within"/>
            </ogc:SpatialOperators>
        </ogc:Spatial_Capabilities>
        <ogc:Scalar_Capabilities>
            <ogc:LogicalOperators/>
            <ogc:ComparisonOperators>
                <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
            </ogc:ComparisonOperators>
        </ogc:Scalar_Capabilities>
        <ogc:Id_Capabilities>
            <ogc:EID/>
        </ogc:Id_Capabilities>
    </ogc:Filter_Capabilities>
</csw:Capabilities>
```

1.6.3. FTP Endpoint

The FTP endpoint provides a method for ingesting files directly into the DDF catalog using the FTP protocol.

The FTP endpoint can be accessed from any FTP client of choice. Some common clients are FileZilla, PuTTY, or the FTP client provided in the terminal. The default port number is **8021**. If FTPS is enabled with 2-way TLS, a client that supports client authentication is required.

1.6.3.1. FTP Endpoint Create Examples

To ingest files into DDF send an FTP `PUT` request to the DDF server.

FTP Endpoint URL

```
ftp://<FQDN>:<PORT>
```

1.6.3.2. FTP Endpoint Rename Command

The FTP endpoint also supports renaming files as they are ingested with the FTP `RNTO` operation.

Files named with a leading `.`, such as `.<FILENAME>`, are held by DDF without being ingested until the `rename` command is sent.

PUT Command Example

```
PUT .<FILENAME>.txt
```

Rename Command Example

```
rename .<NEW_FILENAME>.txt
```

The endpoint will complete the ingest process when the rename command is sent. The filename on the original file system will NOT be changed.

1.6.4. OpenSearch Endpoint

The OpenSearch Endpoint enables a client to send query parameters and receive search results. This endpoint uses the input query parameters to create an OpenSearch query. The client does not need to specify all of the query parameters, only the query parameters of interest.

The OpenSearch specification defines a file format to describe an OpenSearch endpoint. This file is XML-based and is used to programatically retrieve a site's endpoint, as well as the different parameter options a site holds. The parameters are defined via the OpenSearch and CDR IPT Specifications.

1.6.4.1. OpenSearch Contextual Queries

To use the OpenSearch endpoint for a query, send a `GET` request with the query options as parameters

OpenSearch Query URL

```
https://<FQDN>:<PORT>/services/catalog/query?<NAME>="<VALUE>"
```

Table 12. OpenSearch Parameter List

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`searchTerms`	`q`	URL-encoded, space-delimited list of search terms	Complex contextual search string.
`count`	`count`	Integer >= 0	Maximum # of results to retrieve. default: `10`
`startIndex`	`start`	integer > 0	Index of first result to return. This value uses a one-based index for the results. default: `1`
`format`	`format`	Requires a transformer shortname as a string, possible values include, when available, `atom`, `html`, and `kml`. See Query Response transformers for more possible values.	Defines the format that the return type should be in. default: `atom`

Sample OpenSearch Textual Query

```
https://<FQDN>:<PORT>/services/catalog/query?q="Aliquam"&count=20
```

1.6.4.1.1. Complex OpenSearch Contextual Query Format

The OpenSearch Endpoint supports the following operators: `AND`, `OR`, and `NOT`. These operators are case sensitive. Implicit `ANDs` are also supported.

Use parentheses to change the order of operations. Use quotes to group keywords into literal expressions.

See the OpenSearch specification for more syntax specifics.

OpenSearch Endpoint Complex Query Example

```
https://<FQDN>:<PORT>/services/catalog/query?q='cat OR dog'
```

1.6.4.2. OpenSearch Temporal Queries

Queries can also specify a start and end time to narrow results.

Table 13. OpenSearch Temporal Parameters

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`start`	`dtstart`	RFC-3399-defined value:`YYYY-MM-DDTHH:mm:ssZ` or `yyyy-MM-dd'T'HH:mm:ss.SSSZZ`	Specifies the beginning of the time slice of the search. Default value of "1970-01-01T00:00:00Z" is used when `dtend` is specified but `dtstart` is not specified.
`end`	`dtend`	RFC-3399-defined value:`YYYY-MM-DDTHH:mm:ssZ` or `yyyy-MM-dd'T'HH:mm:ss.SSSZZ`	Specifies the ending of the time slice of the search Current GMT date/time is used when `dtstart` is specified but `dtend` is not specified.

OpenSearch Temporal Query Example

```
https://<FQDN>:<PORT>/services/catalog/query?q='*'&dtstart=2019-12-16T00:00:00Z&dtend=2019-12-16T18:00:00Z
```

NOTE

The start and end temporal criteria must be of the format specified above. Other formats are currently not supported. Example:

`2019-12-16T12:00:00.111-04:00`.

The start and end temporal elements are based on modified timestamps for a metacard.

1.6.4.3. OpenSearch Geospatial Queries

Query by location.

Use geospatial query parameters to create a geospatial `INTERSECTS` query, where `INTERSECTS` means geometries that are not `DISJOINT` to the given geospatial parameters.

Table 14. Opensearch Geospatial Parameters

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`lat`	`lat`	`EPSG:4326 (WGS84)` decimal degrees	Used in conjunction with the `lon` and `radius` parameters.
`lon`	`lon`	`EPSG:4326 (WGS84)` decimal degrees	Used in conjunction with the `lat` and `radius` parameters.
`radius`	`radius`	`EPSG:4326 (WGS84)` meters along the Earth's surface > 0	Specifies the search distance in meters from the `lon`,`lat` point. Used in conjunction with the `lat` and `lon` parameters. default: `5000`
`polygon`	`polygon`	Comma-delimited list of lat/lon (`EPSG:4326 (WGS84)` decimal degrees) pairs, in clockwise order around the polygon, where the last point is the same as the first in order to close the polygon. (e.g. `-80,-170,0,-170,80,-170,80,170,0,170,-80,170,-80,-170`)	According to the OpenSearch Geo Specification this is **deprecated**. Use the `geometry` parameter instead.
`box`	`bbox`	4 comma-delimited `EPSG:4326 (WGS84)` decimal degrees coordinates in the format West,South,East,North	

OpenSearch Element	HTTPS Parameter	Possible Values	Comments

`geometry`	`geometry`	WKT Geometries
		
		Examples:
		
		`POINT(10 20)` where 10 is the longitude and 20 is the latitude.
		
		`POLYGON ( ( 30 10, 10 20, 20 40, 40 40, 30 10 ) )`. 30 is longitude and 10 is latitude for the first point.
		
		`MULTIPOLYGON ( ( (40 40, 20 45, 45 30, 40 40) ), ( (20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20) ) )`
		
		`GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))`
			Make sure to repeat the starting point as the last point to close the polygon.

OpenSearch GeoSpatial Query Example

```
https://localhost:8993/services/catalog/query?q='*'&lon=44.792&lat=-6.171
```

1.6.4.4. Additional OpenSearch Query Parameters

The OpenSearch Endpoint can also use these additional parameters to refine queries

Table 15. OpenSearch Query Extensions

OpenSearch Element	HTTPS Parameter	Possible Values	Comments

`sort`	`sort`	`<sbfield>:<sborder>` where
		
		`<sbfield>` is `date` or `relevance`
		
		`<sborder>` is `asc` or `desc`
			`<sborder>` is optional but has a value of `asc` or `desc` (default is `desc`). However, when `<sbfield>` is `relevance`, `<sborder>` must be `desc`.
			
			Sorting by `date` will sort the results by the `effective` date.
			
			default: `relevance:desc`

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`maxResults`	`mr`	Integer >= 0	Maximum # of results to return. If `count` is also specified, the `count` value will take precedence over the `maxResults` value. default: `1000`
`maxTimeout`	`mt`	Integer > 0	Maximum timeout (milliseconds) for query to respond. default: `300000` (5 minutes)
`dateOffset`	`dtoffset`	Integer > 0	Specifies an offset (milliseconds), backwards from the current time, to search on the modified time field for entries.
`type`	`type`	Any valid datatype (e.g. `Text`)	Specifies the type of data to search for.
`version`	`version`	Comma-delimited list of strings (e.g. 20,30)	Version values for which to search.
`selector`	`selector`	Comma-delimited list of XPath string selectors (e.g. `//namespace:example`, //example `)	Selectors to narrow the query.

Table 16. Federated Search

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`routeTo`	`src`	Comma-delimited list of site names to query. Varies depending on the names of the sites in the federation. `local` specifies to query the local site.	If `src` is not provided, the default behavior is to execute an enterprise search to the entire federation.

1.6.5. Queries Endpoint

The queries endpoint enables an application to create, retrieve, update, and delete query metacards.

Query metacards represent queries within the UI. A query metacard is what is persisted in the data store.

The queries endpoint can be used for one or more of these operations on an instance of DDF:

- Create query metacards and store them in the DDF catalog.
- Retrieve all query metacards stored in the DDF catalog and sort them based on attribute and sort order.
- Retrieve a specific query metacard stored in the DDF catalog.
- Update query metacards that are stored in the DDF catalog.
- Delete query metacards that are stored in the DDF catalog.

Queries Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries
```

1.6.5.1. Queries Endpoint Create Examples

To create a query metacard through the queries endpoint, send a `POST` request to the queries endpoint.

Queries Endpoint Create Request Body

```
{
    "cql":"(\"anyText\" ILIKE 'foo bar')",
    "filterTree":"{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "federation":"enterprise",
    "sorts":[
        {
            "attribute":"modified",
            "direction":"descending"
        }
    ],
    "type":"advanced",
    "title":"Search Title"
}
```

A successful create request will return a status of `201 CREATED`.

Queries Endpoint Create Success Response Body


```
{
    "id": "12bfc601cda449d58733eacaf613b93d",
    "title": "Search Title",
    "created": "Apr 18, 2019 10:20:55 AM",
    "modified": "Apr 18, 2019 10:20:55 AM",
    "owner": "admin@localhost.local",
    "cql": "(\"anyText\" ILIKE 'foo bar')",
    "filterTree": "{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "enterprise": null,
    "sources": [],
    "sorts": [
        {
            "attribute": "modified",
            "direction": "descending"
        }
    ],
    "polling": null,
    "federation": "enterprise",
    "type": "advanced",
    "detailLevel": null,
    "schedules": [],
    "facets": []
}
```


An unsuccessful create request will return a status of `500 SERVER ERROR`.

Queries Endpoint Create Failure Response Body


```
{
    "message": "Something went wrong."
}
```


1.6.5.2. Queries Endpoint Retrieve All Examples

To retrieve a query metacard through the queries endpoint, send a `GET` request to the queries endpoint.

Table 17. Path Parameters

Query Param	Description	Default Value	Valid Values	Type
`start`	The starting index of the query to receive.	`1`	`Integer`	`[1, 2^31)`
`count`	The number of queries to return.	`100`	`Integer`	`All integers`

Query Param	Description	Default Value	Valid Values	Type
attr	The attribute to sort the queries by.	modified	All strings	String
sort_by	The sort order to return the queries in.	desc	asc, desc	String
text	A text field to search against a few attributes.	None	All strings	String

A successful retrieval request will return a status of `200 OK`.

1.6.5.3. Queries Endpoint Retrieve All Fuzzy Examples

To retrieve all query metacards based on some text based value through the queries endpoint, send a `GET` request to the queries endpoint specifying a value for `text` as a query parameters.

Retrieve All Queries Fuzzy Search Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries?text=<VALUE>
```

A fuzzy search will only be performed against the `title`, `modified`, `owner`, and `description` attributes.

1.6.5.4. Queries Endpoint Retrieve Examples

Retrieve Specific Query Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries/<ID>
```

To retrieve a specific query metacard through the queries endpoint, send a `GET` request to the queries endpoint with an id.

A successful retrieval request will return a status of `200 OK`.

Query Endpoint Not Found Response Body

```
{
    "message": "Could not find metacard for id: <metacardId>"
}
```

An unsuccessful retrieval request will return a status of `404 NOT FOUND`.

1.6.5.5. Queries Endpoint Update Examples

Update Query Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries/<ID>
```

To update a specific query metacard through the queries endpoint, send a `PUT` request to the queries endpoint with an id.

Update Query Request Request Body

```
{
    "cql":"(\"anyText\" ILIKE 'foo bar')",
    "filterTree":"{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "federation":"enterprise",
    "sorts":[
        {
            "attribute":"modified",
            "direction":"descending"
        }
    ],
    "type":"advanced",
    "title":"New Search Title"
}
```

A successful update request will return a status of `200 OK`.

Update Query Request Response Body


```
{
    "id": "cd6b83db301544e4bb7ece39564261ca",
    "title": "New Search Title",
    "created": "Apr 18, 2019 11:09:35 AM",
    "modified": "Apr 18, 2019 11:09:35 AM",
    "owner": null,
    "cql": "(\"anyText\" ILIKE 'foo bararara')",
    "filterTree": "{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "enterprise": null,
    "sources": [],
    "sorts": [
        {
            "attribute": "modified",
            "direction": "descending"
        }
    ],
    "polling": null,
    "federation": "enterprise",
    "type": "advanced",
    "detailLevel": null,
    "schedules": [],
    "facets": []
}
```


An unsuccessful update request will return a status of `404 NOT FOUND`.

Update Query Unsuccessful Response Body


```
{
    "message": "Form is either restricted or not found."
}
```


1.6.5.6. Queries Endpoint Delete Examples

Delete Query Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries/<ID>
```

To delete a specific query metacard through the queries endpoint, send a `GET` request to the queries endpoint with an id.

A successful deletion request will return a status of `204 NO CONTENT`.

An unsuccessful deletion request will return a status of `404 NOT FOUND`.

Delete Query Not Found Response Body

```
{
    "message": "Form is either restricted or not found."
}
```