



Integrating DDF

Version 2.13.8. Copyright (c) Codice Foundation

Table of Contents

License	1
1. Data and Metadata	2
1.1. Metacards	3
1.1.1. Metacard Type	3
1.1.1.1. Default Metacard Type and Attributes	3
1.1.1.2. Extensible Metacards	3
1.1.2. Metacard Type Registry	4
1.1.3. Attributes	5
1.1.3.1. Attribute Types	5
1.1.3.1.1. Attribute Format	5
1.1.3.1.2. Attribute Naming Conventions	6
1.1.3.2. Result	6
1.1.4. Creating Metacards	6
1.1.4.1. Limitations	7
1.1.4.2. Processing Metacards	7
1.1.4.3. Basic Types	7
1.2. JSON "Definition" Files	8
1.2.1. Definition File Format	9
1.2.2. Deploying Definition Files	9
1.3. Data Validation	9
1.3.1. Validation with Schematron	9
1.3.1.1. Configuring Schematron Services	10
2. Endpoints	10
2.1. Available Endpoints	10
2.1.1. Application Upload Endpoint	11
2.1.1.1. Installing Application Upload Endpoint	11
2.1.1.2. Application Upload Endpoint	11
2.1.1.3. Configuring Application Upload Endpoint	11
2.1.2. Catalog REST Endpoint	11
2.1.2.1. Installing the Catalog REST Endpoint	12
2.1.2.2. Installing the Catalog REST Endpoint	12
2.1.2.3. Configuring the Catalog REST Endpoint	12
2.1.2.4. Using the Catalog REST Endpoint	12
2.1.2.4.1. Sources Operation Example	14
2.1.2.5. Interacting with the REST CRUD Endpoint	15
2.1.2.6. Metacard Transforms with the REST CRUD Endpoint	16

2.1.2.6.1. POST Metadata	16
2.1.2.6.2. Example Responses for ReST Endpoint Error Conditions	17
2.1.3. CometD Endpoint	20
2.1.3.1. Installing CometD Endpoint	20
2.1.3.2. Configuring CometD Endpoint	20
2.1.3.3. Using CometD Endpoint	20
2.1.3.3.1. CometD Queries	21
2.1.3.3.2. Query Request Examples	22
2.1.3.3.3. Query Response Channel	22
2.1.3.3.4. Query Response Examples	23
2.1.3.3.5. CometD Notifications	25
2.1.3.3.6. Using CometD Notifications	25
2.1.3.3.7. Receive Notifications	26
2.1.3.3.8. Notification Events	26
2.1.3.3.9. Persistence of Notifications	27
2.1.3.3.10. Notification Operations Channel	27
2.1.3.3.11. Activity Events Channel	27
2.1.3.3.12. Activity Operations Channel	29
2.1.4. CSW Endpoint	29
2.1.4.1. Installing CSW Endpoint	29
2.1.4.2. Configuring CSW Endpoint	30
2.1.4.3. CSW Endpoint URL	30
2.1.4.3.1. CSW Endpoint Operations	30
2.1.4.3.2. Transaction Operations	65
2.1.4.3.3. Transaction Insert Sub-Operation HTTP POST	66
2.1.4.3.4. Transaction Insert Response	67
2.1.4.3.5. Transaction Update Sub-Operation HTTP POST	68
2.1.4.3.6. Transaction Delete Sub-Operation HTTP POST	73
2.1.4.3.7. Subscription GetRecords Operation	75
2.1.4.3.8. Subscription GetRecords HTTP GET	75
2.1.4.3.9. Subscription GetRecords HTTP POST	75
2.1.4.3.10. Subscription GetRecords HTTP PUT	76
2.1.4.3.11. Subscription HTTP GET or HTTP DELETE Request	78
2.1.4.3.12. Subscription HTTP GET or HTTP DELETE Response	78
2.1.4.3.13. Example Responses for CSW Endpoint Error Conditions	79
2.1.5. FTP Endpoint	88
2.1.5.1. Installing the FTP Endpoint	88
2.1.5.2. Configuring the FTP Endpoint	88

2.1.5.3. Using FTP Endpoint	89
2.1.5.3.1. From Code:	89
2.1.5.3.2. From an FTP client:	89
2.1.6. KML Endpoint	89
2.1.6.1. Installing the KML Endpoint	89
2.1.6.2. Configuring the KML Endpoint	90
2.1.6.3. Using the KML Endpoint	90
2.1.7. Metrics Endpoint	91
2.1.7.1. Installing the Metrics Endpoint	91
2.1.7.2. Configuring the Metrics Endpoint	91
2.1.7.3. Using the Metrics Endpoint	91
2.1.7.3.1. Metrics Data Supported Formats	92
2.1.7.3.2. Add Custom Metrics to the Metrics Tab	95
2.1.7.4. Usage Limitations of the Metrics Endpoint	96
2.1.8. OpenSearch Endpoint	96
2.1.8.1. Installing the OpenSearch Endpoint	96
2.1.8.2. Configuring the OpenSearch Endpoint	96
2.1.8.3. OpenSearch URL	96
2.1.8.3.1. From Code:	96
2.1.8.3.2. From a Web Browser:	96
2.1.8.3.3. Parameter List	97
2.1.8.3.4. Supported Complex Contextual Query Format	101
2.1.9. WPS Endpoint	101
2.1.9.1. Installing WPS Endpoint	101
2.1.9.2. Configuring WPS Endpoint	102
2.1.9.3. WPS Endpoint URL	102
2.1.10. WPS Endpoint Operations	102
2.2. Endpoint Utility Services	109
2.2.1. Compression Services	109
2.2.1.1. Installing a Compression Service	110
2.2.1.2. Configuring Compression Services	110
3. Eventing	111
3.1. Eventing Intro	111
3.2. Eventing Components	111
3.3. Subscriptions	112
3.3.1. Subscriptions	112
3.3.1.1. Subscription Lifecycle	112
3.3.1.1.1. Creation	112

3.3.1.1.2. Evaluation	112
3.3.1.1.3. Update Evaluation	112
3.3.1.1.4. Durability	112
3.3.2. Creating a Subscription	113
3.3.2.1. Event Processing and Notification	113
3.3.2.1.1. Using DDF Implementation	114
3.3.2.2. Delivery Method	114
4. Security Services	115
4.1. Encryption Service	115
4.1.1. Encryption Command	115
4.2. Expansion Service	115
4.2.1. Configuring Expansion Service	115
4.2.1.1. Expansion Service Instances and Configuration	115
4.2.1.2. Expansion Command Examples and Explanation	117
4.2.1.2.1. security:expansions	117
4.2.1.2.2. security:expand	117
4.3. Security IdP	118
4.4. Security STS	118
4.4.1. Security STS Client Config	119
4.4.1.1. Installing the Security STS Client Config	119
4.4.1.2. Configuring the Security STS Client Config	119
4.4.2. External/WS-S STS Support	119
4.4.2.1. Security STS WSS	119
4.4.2.2. Security STS Address Provider	119
4.4.3. Security STS LDAP Login	120
4.4.3.1. Installing the Security STS LDAP Login	120
4.4.3.2. Configuring the Security STS LDAP Login	120
4.4.4. Security STS LDAP Claims Handler	120
4.4.4.1. Installing Security STS LDAP Claims Handler	121
4.4.4.2. Configuring the Security STS LDAP Claims Handler	121
4.4.5. Security STS Server	122
4.4.5.1. Installing the Security STS Server	122
4.4.5.2. Configuring the Security STS Server	122
4.4.6. Security STS Service	123
4.4.6.1. Installing the Security STS Realm	123
4.4.6.2. Configuring the Security STS Realm	123

License

Copyright (c) Codice Foundation.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

This document last updated: 2019-04-09 18:03:35:259.

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.

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

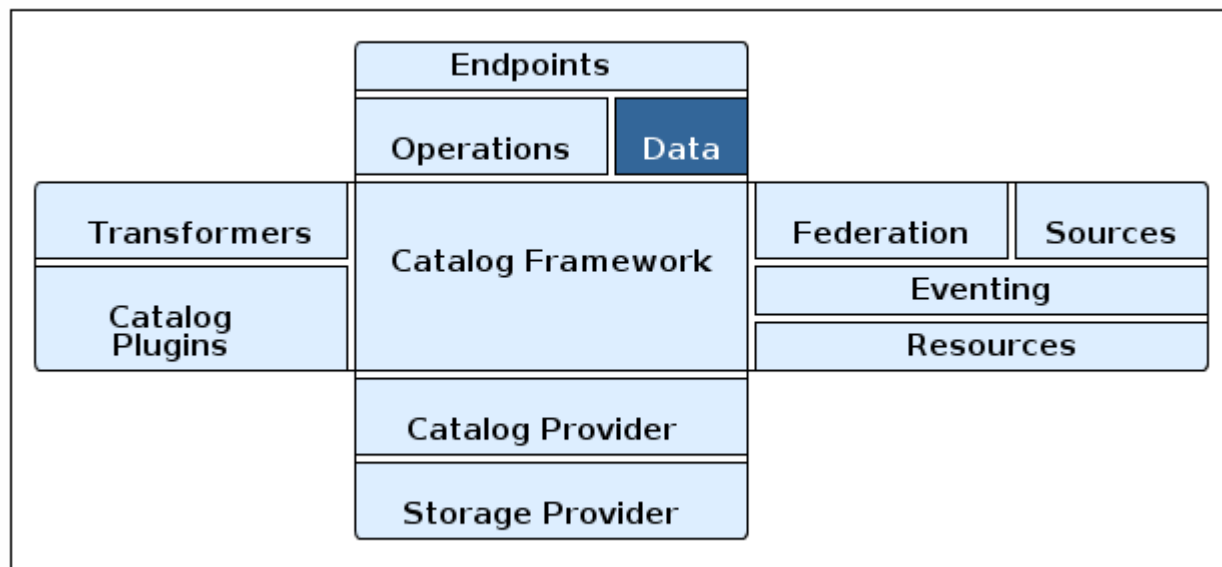
If integrating with an existing installation of DDF, continue to the following sections on endpoints and data/metadata management.

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. Data and Metadata



Catalog Architecture Diagram: Data

The Catalog stores and translates Metadata, which can be transformed into many data formats, shared, and queried. The primary form of this metadata is the metacard. A **Metacard** is a container for metadata. **CatalogProviders** accept **Metacards** as input for ingest, and **Sources** search for metadata and return matching **Results** that include **Metacards**.

1.1. Metacards

A metacard is a single instance of metadata in the Catalog (an instance of a metacard type) which generally contains general information about the product, such as the title of the product, the product's geo-location, the date the product was created and/or modified, the owner or producer, and/or the security classification.

1.1.1. Metacard Type

A metacard type indicates the attributes available for a particular metacard. It is a model used to define the attributes of a metacard, much like a schema.

A metacard type indicates the attributes available for a particular type of data. For example, an image may have different attributes than a PDF document, so each could be defined to have their own metacard type.

1.1.1.1. Default Metacard Type and Attributes

Most metacards within the system are created using the default metacard type or a metacard type based on the default type. The default metacard type of the system can be programmatically retrieved by calling `ddf.catalog.data.impl.MetacardImpl.BASIC_METACARD`. The name of the default `MetacardType` can be retrieved from `ddf.catalog.data.MetacardType.DEFAULT_METACARD_TYPE_NAME`.

The default metacard type has the following required attributes. Though the following attributes are required on all metacard types, setting their values is optional except for ID.

Core Attributes

NOTE

It is highly recommended when referencing a default attribute name to use the `ddf.catalog.data.types.*` interface constants whenever possible. Mapping to a normalized taxonomy allows for higher quality transformations between different formats and for improved federation. This neutral profile facilitates improved search and discovery across disparate data types.

WARNING

Every [Source](#) should at the very least return an ID attribute according to Catalog API. Other fields may or may not be applicable, but a unique ID must be returned by a source.

1.1.1.2. Extensible Metacards

Metacard extensibility is achieved by creating a new `MetacardType` that supports attributes in addition to the required attributes listed above.

Required attributes must be the base of all extensible metacard types.

WARNING

Not all [Catalog Providers](#) support extensible metacards. Nevertheless, each Catalog Provider should at least have support for the default [MetacardType](#); i.e., it should be able to store and query on the attributes and attribute formats specified by the default metacard type. Catalog providers are neither expected nor required to store attributes that are not in a given metacard's type.

Consult the documentation of the Catalog Provider in use for more information on its support of extensible metacards.

Often, the [BASIC_METACARD MetacardType](#) does not provide all the functionality or attributes necessary for a specific task. For performance or convenience purposes, it may be necessary to create custom attributes even if others will not be aware of those attributes. One example could be if a user wanted to optimize a search for a date field that did not fit the definition of [CREATED](#), [MODIFIED](#), [EXPIRATION](#), or [EFFECTIVE](#). The user could create an additional [java.util.Date](#) attribute in order to query the attribute separately.

[Metacard](#) objects are extensible because they allow clients to store and retrieve standard and custom key/value Attributes from the [Metacard](#). All [Metacards](#) must return a [MetacardType](#) object that includes an [AttributeDescriptor](#) for each [Attribute](#), indicating its key and value type. [AttributeType](#) support is limited to those types defined by the Catalog.

New [MetacardType](#) implementations can be made by implementing the [MetacardType](#) interface.

1.1.2. Metacard Type Registry

WARNING

The [MetacardTypeRegistry](#) is experimental. While this component has been tested and is functional, it may change as more information is gathered about what is needed and as it is used in more scenarios.

The [MetacardTypeRegistry](#) allows DDF components, primarily catalog providers and sources, to make available the [MetacardTypes](#) that they support. It maintains a list of all supported [MetacardTypes](#) in the [CatalogFramework](#), so that other components such as [Endpoints](#), [Plugins](#), and [Transformers](#) can make use of those [MetacardTypes](#). The [MetacardType](#) is essential for a component in the [CatalogFramework](#) to understand how it should interpret a metacard by knowing what attributes are available in that metacard.

For example, an endpoint receiving incoming metadata can perform a lookup in the [MetacardTypeRegistry](#) to find a corresponding [MetacardType](#). The discovered [MetacardType](#) will then be used to help the endpoint populate a metacard based on the specified attributes in the [MetacardType](#). By doing this, all the incoming metadata elements can then be available for processing, cataloging, and searching by the rest of the [CatalogFramework](#).

[MetacardTypes](#) should be registered with the [MetacardTypeRegistry](#). The [MetacardTypeRegistry](#) makes those [MetacardTypes](#) available to other DDF [CatalogFramework](#) components. Other components that need to know how to interpret metadata or metacards should look up the appropriate [MetacardType](#) from the

registry. By having these **MetacardTypes** available to the **CatalogFramework**, these components can be aware of the custom attributes.

The **MetacardTypeRegistry** is accessible as an OSGi service. The following blueprint snippet shows how to inject that service into another component:

MetacardTypeRegistry Service Injection

```
<bean id="sampleComponent" class="ddf.catalog.SampleComponent">
  <argument ref="metacardTypeRegistry" />
</bean>

<!-- Access MetacardTypeRegistry -->
<reference id="metacardTypeRegistry" interface="ddf.catalog.data.MetacardTypeRegistry"/>
```

The reference to this service can then be used to register new **MetacardTypes** or to lookup existing ones.

Typically, new **MetacardTypes** will be registered by **CatalogProviders** or sources indicating they know how to persist, index, and query attributes from that type. Typically, Endpoints or **InputTransformers** will use the lookup functionality to access a **MetacardType** based on a parameter in the incoming metadata. Once the appropriate **MetacardType** is discovered and obtained from the registry, the component will know how to translate incoming raw metadata into a DDF Metacard.

1.1.3. Attributes

An attribute is a single field of a metacard, an instance of an attribute type. Attributes are typically indexed for searching by a source or catalog provider.

1.1.3.1. Attribute Types

An attribute type indicates the attribute format of the value stored as an attribute. It is a model for an attribute.

1.1.3.1.1. Attribute Format

An enumeration of attribute formats are available in the catalog. Only these attribute formats may be used.

Table 1. Attribute Formats

AttributeFormat	Description
BINARY	Attributes of this attribute format must have a value that is a Java byte[] and AttributeType.getBinding() should return Class<ArrayOf byte> .
BOOLEAN	Attributes of this attribute format must have a value that is a Java boolean.

AttributeFormat	Description
DATE	Attributes of this attribute format must have a value that is a Java date.
DOUBLE	Attributes of this attribute format must have a value that is a Java double.
FLOAT	Attributes of this attribute format must have a value that is a Java float.
GEOMETRY	Attributes of this attribute format must have a value that is a WKT-formatted Java string.
INTEGER	Attributes of this attribute format must have a value that is a Java integer.
LONG	Attributes of this attribute format must have a value that is a Java long.
OBJECT	Attributes of this attribute format must have a value that implements the serializable interface.
SHORT	Attributes of this attribute format must have a value that is a Java short.
STRING	Attributes of this attribute format must have a value that is a Java string and treated as plain text.
XML	Attributes of this attribute format must have a value that is a XML-formatted Java string.

1.1.3.1.2. Attribute Naming Conventions

Catalog taxonomy elements follow the naming convention of `group-or-namespace.specific-term`, except for extension fields outside of the core taxonomy. These follow the naming convention of `ext.group-or-namespace.specific-term` and must be namespaced. Nesting is not permitted.

1.1.3.2. Result

A single "hit" included in a query response.

A result object consists of the following:

- a metacard.
- a relevance score if included.
- distance in meters if included.

1.1.4. Creating Metacards

The quickest way to create a `Metacard` is to extend or construct the `MetacardImpl` object. `MetacardImpl` is the most commonly used and extended `Metacard` implementation in the system because it provides a convenient way for developers to retrieve and set `Attributes` without having to create a

new `MetacardType` (see below). `MetacardImpl` uses `BASIC_METACARD` as its `MetacardType`.

1.1.4.1. Limitations

A given developer does not have all the information necessary to programmatically interact with any arbitrary source. Developers hoping to query custom fields from extensible `Metacards` of other sources cannot easily accomplish that task with the current API. A developer cannot question a source for all its *queryable* fields. A developer only knows about the `MetacardTypes` which that individual developer has used or created previously.

The only exception to this limitation is the `Metacard.ID` field, which is required in every `Metacard` that is stored in a source. A developer can always request `Metacards` from a source for which that developer has the `Metacard.ID` value. The developer could also perform a wildcard search on the `Metacard.ID` field if the source allows.

1.1.4.2. Processing Metacards

As `Metacard` objects are created, updated, and read throughout the Catalog, care should be taken by all catalog components to interrogate the `MetacardType` to ensure that additional `Attributes` are processed accordingly.

1.1.4.3. Basic Types

The Catalog includes definitions of several basic types all found in the `ddf.catalog.data.BasicTypes` class.

Table 2. Basic Types

Name	Type	Description
<code>BASIC_METACARD</code>	<code>MetacardType</code>	Represents all required Metacard Attributes.
<code>BINARY_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.BINARY</code> .
<code>BOOLEAN_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.BOOLEAN</code> .
<code>DATE_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.DATE</code> .
<code>DOUBLE_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.DOUBLE</code> .
<code>FLOAT_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.FLOAT</code> .

Name	Type	Description
GEO_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.GEOMETRY</code> .
INTEGER_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.INTEGER</code> .
LONG_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.LONG</code> .
OBJECT_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.OBJECT</code> .
SHORT_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.SHORT</code> .
STRING_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.STRING</code> .
XML_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.Format.XML</code> .

1.2. JSON "Definition" Files.

WARNING


This section concerns capabilities that are considered experimental. The features described in this section may change or be removed in a future version of the application.

DDF supports adding new attribute types, metacard types, validators, and more using json-formatted definition files.

The following may be defined in a JSON definition file:

- [Attribute Types](#)
- [Metacard Types](#)
- [Global Attribute Validators](#)
- [Default Attribute Values](#)
- [Attribute Injections](#)

1.2.1. Definition File Format

A definition file follows the JSON format as specified in [ECMA-404](#) . All definition files must be valid JSON in order to be parsed.

A single definition file may define as many of the types as needed. This means that types can be defined across multiple files for grouping or clarity.

1.2.2. Deploying Definition Files

The file must have a `.json` extension in order to be picked up by the deployer. Once the definition file is ready to be deployed, put the definition file `<filename>.json` into the `etc/definitions` folder.

Definition files can be added, updated, and/or deleted in the `etc/definitions` folder. The changes are applied dynamically and no restart is required.

If a definition file is removed from the `etc/definitions` folder, the changes that were applied by that file will be undone.

1.3. Data Validation

DDF can be configured to perform validation on ingested documents to verify the integrity of the metadata brought into the catalog.

Available Validation Services

Validation with Schematron

Introduction to validation with schematron.

1.3.1. Validation with Schematron

DDF uses [Schematron Validation](#) to validate metadata ingested into the catalog.

Custom schematron rulesets can be used to validate metacard metadata. Multiple services can be created, and each service can have multiple rulesets associated with it. Namespaces are used to distinguish services. The root schematron files may be placed anywhere on the file system as long as they are configured with an absolute path. Any root schematron files with a relative path are assumed to be relative to `<DDF_HOME>/schematron`.

TIP

Schematron files may reference other schematron files using an include statement with a relative path. However, when using the document function within a schematron ruleset to reference another file, the path must be absolute or relative to the DDF installation home directory.

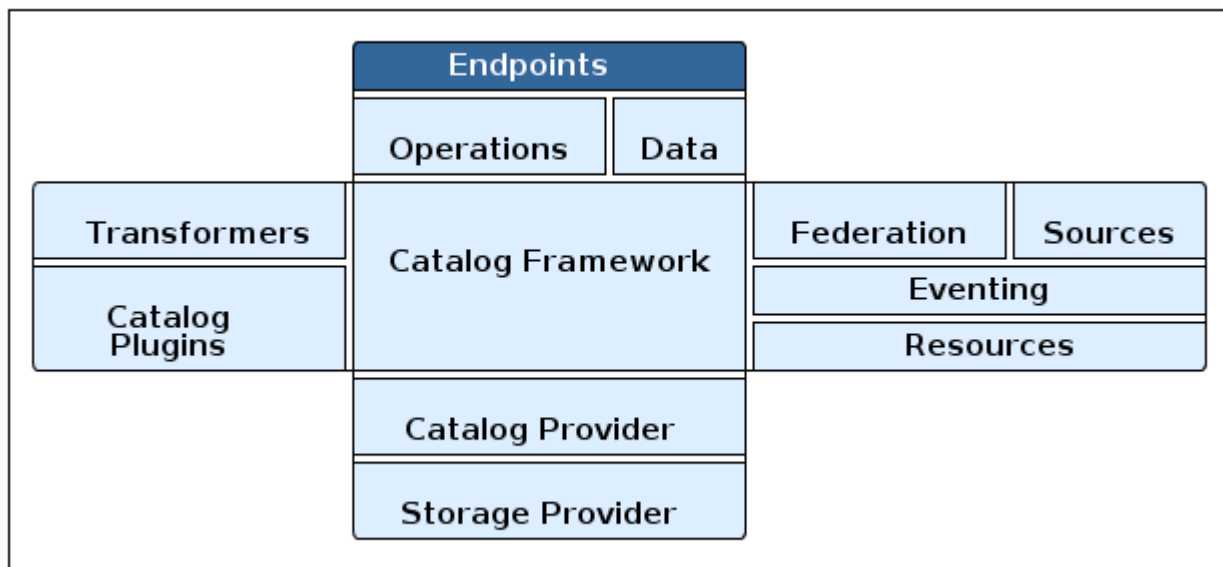
1.3.1.1. Configuring Schematron Services

Schematron validation services are configured with a namespace and one or more schematron rulesets. Additionally, warnings may be suppressed so that only errors are reported.

To create a new service:

- Navigate to the **Admin Console**.
- Select the **Catalog**.
- Select **Configuration**.
- Ensure that `catalog-schematron-plugin` is started.
- Select **Schematron Validation Services**.

2. Endpoints



Endpoints

Endpoints act as a proxy between the client and the [Catalog Framework](#).

2.1. Available Endpoints

The following endpoints are available in a standard installation of DDF:

Application Upload Endpoint

Enables uploading new and upgraded applications to the system.

Catalog REST Endpoint

Allows clients to perform CRUD operations on the Catalog using REST, a simple architectural style that performs communication using HTTP.

CometD Endpoint

Enables asynchronous search capabilities.

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.

KML Endpoint

Generates a view-based KML Query Results Network Link.

Metrics Endpoint

Reports on system metrics.

OpenSearch Endpoint

Sends query parameters and receives search results.

WPS Endpoint

Execute and monitor long running processes.

2.1.1. Application Upload Endpoint

The Application Upload Endpoint enables uploading new and upgraded applications to the system.

2.1.1.1. Installing Application Upload Endpoint

The Application Upload Endpoint is installed by default with a standard installation as part of the Admin application.

2.1.1.2. Application Upload Endpoint

The Application endpoint is available at `https://{FQDN}:{PORT}/services/application`.

2.1.1.3. Configuring Application Upload Endpoint

No configuration necessary.

2.1.2. Catalog REST Endpoint

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

2.1.2.1. Installing the Catalog REST Endpoint

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

The URL exposing the REST functionality is located at `https://{FQDN}:{PORT}/services/catalog`.

2.1.2.2. Installing the Catalog REST Endpoint

The Catalog REST Endpoint is installed by default with a standard installation in the Catalog application.

2.1.2.3. Configuring the Catalog REST Endpoint

The RESTful CRUD Endpoint has no configurable properties. It can only be installed or uninstalled.

2.1.2.4. Using the Catalog REST Endpoint

The Catalog REST Endpoint provides the capability to query, create, update, and delete metacards and associated resource in the catalog provider as follows:

Operation	HTTP Request	Details	Example URL
<code>create</code>	HTTP POST	HTTP request body contains the input to be ingested. <code><input transformer></code> is the name of the transformer to use when parsing metadata (optional).	<code>http://{FQDN}:{PORT}/services/catalog?transform=<input transformer></code>

Operation	HTTP Request	Details	Example URL
update	HTTP PUT	<p>The ID of the Metacard to be updated is appended to the end of the URL. The updated metadata is contained in the HTTP body.</p> <p><code><metacardId></code> is the Metacard.ID of the metacard to be updated and <code><input transformer></code> is the name of the transformer to use when parsing an override metadata attribute (optional).</p>	http://{FQDN}:{PORT}/services/catalog/<metacardId>?transform=<input transformer>
delete	HTTP DELETE	<p>The ID of the Metacard to be deleted is appended to the end of the URL.</p> <p><code><metacardId></code> is the Metacard.ID of the metacard to be deleted.</p>	http://{FQDN}:{PORT}/services/catalog/<metacardId>

Operation	HTTP Request	Details	Example URL
read	HTTP GET	<p>The ID of the Metacard to be retrieved is appended to the end of the URL.</p> <p>By default, the response body will include the XML representation of the Metacard.</p> <p><metacardId> is the Metacard.ID of the metacard to be retrieved.</p>	http://{FQDN}:{PORT}/services/catalog/<metacardId>
federated read	HTTP GET	<p>The SOURCE ID of a federated source is appended to the URL before the ID of the Metacard to be retrieved is appended to the end.</p> <p><sourceId> is the FEDERATED SOURCE ID and <metacardId> is the Metacard.ID of the Metacard to be retrieved.</p>	http://{FQDN}:{PORT}/services/catalog/sources/<sourceId>/<metacardId>
sources	HTTP GET	Retrieves information about federated sources, including sourceId, availability, contentTypes, and version.	http://{FQDN}:{PORT}/services/catalog/sources/

2.1.2.4.1. Sources Operation Example

In the example below there is the local DDF distribution and a DDF OpenSearch federated source with id "DDF-OS".

```
[
  {
    "id" : "DDF-OS",
    "available" : true,
    "contentTypes" :
      [
      ],
    "version" : "2.0"
  },
  {
    "id" : "ddf.distribution",
    "available" : true,
    "contentTypes" :
      [
      ],
    "version" : "2.5.0-SNAPSHOT"
  }
]
```

Note that for all RESTful CRUD commands only one metacard ID is supported in the URL, i.e., bulk operations are not supported.

2.1.2.5. Interacting with the REST CRUD Endpoint

Any web browser can be used to perform a REST read. Various other tools and libraries can be used to perform the other HTTP operations on the REST endpoint (e.g., soapUI, cURL, etc.)

The REST endpoint can be used to upload resources as attachments. 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 1 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**).

For example:

```

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--

```

2.1.2.6. Metacard Transforms with the REST CRUD Endpoint

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

1. Install the appropriate feature for the desired transformer. If desired transformer is already installed such as those that come out of the box (**xml,html,etc**), then skip this step.
2. Make a read request to the REST URL specifying the catalog id.
3. Add a transform query parameter to the end of the URL specifying the shortname of the transformer to be used (e.g., **transform=kml**).

Example Metacard Transform

```
http://{FQDN}:{PORT}/services/catalog/<metacardId>?transform=<TRANSFORMER_ID>
```

TIP Transforms also work on read operations for metacards in federated sources.
https://{FQDN}:{PORT}/services/catalog/sources/<sourceId>/<metacardId>?transform=<TRANSFORMER_ID>

See [Metacard Transformers](#) for details on metacard transformers.

2.1.2.6.1. POST Metadata

The following is a successful post of well-formed XML data sent to the Catalog ReST endpoint.

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>
```

2.1.2.6.2. Example Responses for ReST Endpoint Error Conditions

The following are example data and expected errors responses that will be returned for each error condition.

HTTP error codes are also returned. https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_Client_errors

Malformed XML

The following request with malformed XML data sent to the Catalog ReST endpoint.

Malformed XML Example

```
<?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>
</document>
```

A HTTP 400 is returned and the following response body is returned. The specific error is logged in the

error log.

Malformed XML Response

```
<pre>Error while storing entry in catalog: </pre>
```

Request with Unknown Schema

The following is a malformed XML document sent to the Catalog ReST endpoint.

Malformed XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<mydoc xmlns="http://example.com/unknown" 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>
</mydoc>
```

Creates a generic resource metacard with the provided XML as content for the **metadata** XML field in the metacard.

Request with Missing XML Prologue

The following is an example request with a missing XML prologue sent to the Catalog ReST endpoint.

Missing XML Tag 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="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>
```

Metacard is created successfully

Request with Non-XML Data

The following is an example request with non-XML data sent to the Catalog ReST endpoint.

Non-XML data Example

```
title: Non-XML title
id: abc123
```

Metacard will be created and the content will stored in the **metadata** field.

Request with Invalid Transform

Testing valid data with an invalid **transform=invalid** appended to the POST URL:
{public_url}/services/catalog?transform=blah

Valid data with an invalid `?transform=invalid`

```
<?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>
```


A HTTP 400 is returned and the following response body is returned. The specific error is logged in the error log.

Malformed XML Response

```
<pre>Error while storing entry in catalog: </pre>
```

2.1.3. CometD Endpoint

IMPORTANT This Feature has been **DEPRECATED** and will be removed in a future version.

The [CometD](#)  endpoint enables asynchronous search capabilities. The CometD protocol is used to execute searches, retrieve results, and receive notifications.

For an example of using CometD within a webapp see: [distribution/sdk/sample-cometd/](#)

2.1.3.1. Installing CometD Endpoint

The CometD Endpoint is installed by default with a standard installation in the Search UI application.

2.1.3.2. Configuring CometD Endpoint

The CometD endpoint has no configurable properties.

2.1.3.3. Using CometD Endpoint

```
https://{FQDN}:{PORT}/search/cometd
```

2.1.3.3.1. CometD Queries

Queries can be executed over CometD using the `/service/query` channel. Query messages are JSON-formatted and use CQL alongside several other parameters.

Table 3. Query Parameters

Parameter Name	Description	Required
<code>src</code>	Comma-delimited list of federated sites to search over	No
<code>cql</code>	CQL query. See OpenGIS® Catalogue Services Specification ↗ for more information about CQL.	Yes
<code>sort</code>	Sort Type The format for the sort options is <code><Sort Field>:<Sort Order></code> , where <code><Sort Order></code> can be either <code>asc</code> or <code>desc</code> .	No
<code>id</code>	Query ID (Must be a unique ID, such as a UUID). This determines the channel that the query results will be returned on.	Yes
<code>count</code>	Number of entries to return in the response. Default is 10.	No
<code>start</code>	Specifies the number of the first result that should be returned.	No
<code>timeout</code>	Time (in milliseconds) to wait for response.	No

Before a query is published the client should subscribe to the channel that will be passed in to the `id` field in order to receive query results once the query is executed.

For example if the following id was generated `3b19bc9c-2155-4ca6-bae8-65a9c8e373f6`, the client should subscribe to `/3b19bc9c-2155-4ca6-bae8-65a9c8e373f6`

Then the following example query could be executed:

`/service/query`

```
{
  "src": "ddf.distribution",
  "cql": "(\"anyText\" ILIKE 'foo')",
  "id": "3b19bc9c-2155-4ca6-bae8-65a9c8e373f6",
  "sort": "asc"
}
```

This would return any results matching the text `foo` on the `/3b19bc9c-2155-4ca6-bae8-65a9c8e373f6`

channel

2.1.3.3.2. Query Request Examples

Enterprise Contextual Query

```
"data": {  
  "count": 250,  
  "format": "geojson",  
  "id": "4303ba5d-21af-4878-9a4c-808e80052e6c",  
  "cql": "anyText LIKE '*'",  
  "start": 1  
}
```

Multiple Site Temporal Absolute Query

```
"data": {  
  "count": 250,  
  "format": "geojson",  
  "id": "4303ba5d-21af-4878-9a4c-808e80052e6c",  
  "cql": "modified DURING 2014-09-01T00:00:00Z/2014-09-30T00:00:00Z",  
  "src": "DDF-OS,ddf.distribution",  
  "start": 1  
}
```

Enterprise Spatial Bounding Box Query

```
"data": {  
  "count": 250,  
  "format": "geojson",  
  "id": "4303ba5d-21af-4878-9a4c-808e80052e6c",  
  "cql": "INTERSECTS(anyGeo, POLYGON ((-112.7786 32.2159, -112.7786 45.6441, -83.7297 45.6441, -83.7297 32.2159, -112.7786 32.2159)))",  
  "start": 1  
}
```

2.1.3.3.3. Query Response Channel

The query responses are returned on the `<id>` channel, which should be subscribed to in order to retrieve the results. Replace `<id>` with the id that was used in the request. The [Subscribing to Notifications](#) section details how to subscribe to a CometD channel.

The response is returned as a data map that contains an internal map with the following keys:

Table 4. Query Response Message Format

Map Key	Description	Value Type
<code>id</code>	ID that corresponds to the request.	String
<code>hits</code>	Total number of query hits that were found by the server. Depending on the 'count' in the request, not all of the results may be returned.	Integer >= 0
<code>results</code>	Array of metacard results, formatted as defined by the GeoJSON Metacard Transformer.	Array of Maps
<code>results/metacard/is-resource-local</code>	A property indicating whether a metacard's associated resource is cached.	Boolean
<code>results/metacard/actions</code>	An array of actions that applies to each metacard, injected into each metacard containing an id, title, description, and url.	Array of Maps
<code>status</code>	Array of status for each source queried.	Array
<code>status.state</code>	Specifies the state of the query: SUCCEEDED, FAILED, ACTIVE.	String
<code>status.elapsed</code>	Time in milliseconds that it took for the source to complete the request.	Integer >= 0
<code>status.hits</code>	Number of records that were found on the source that matched the query	Integer >= 0
<code>status.id</code>	ID of the federated source	String
<code>status.results</code>	Number of results that were returned in this response from the source	Integer >= 0
<code>types</code>	A Map mapping a metacard-type's name to a map about that metacard-type. Only metacard-types represented by the metacards returned in the query are represented. The Map defining a particular <code>metacard-type</code> maps the fields supported by that <code>metacardtype</code> to the datatype for that particular field.	Map of Maps

2.1.3.3.4. Query Response Examples

Example Query Response

```
{
  "data": {
    "hits": 1,
    "metacard-types": {
      "ddf.metacard": {...}
    },
    "id": "6f0e04e9-acd1-4935-b9dd-c83e770a36d5",
    "results": [
      {
        "metacard": {
```

```

        "is-resource-local": false,
        "cached": "2016-07-13T19:22:18.220+0000",
        "geometry": {
            "coordinates": [
                -84.415337,
                42.729925
            ],
            "type": "Point"
        },
        "type": "Feature",
        "actions": [...],
        "properties": {
            "thumbnail": "...",
            "metadata": "<?xml version=\"1.0\" encoding=\"UTF-8
\\\"?><metadata>...</metadata>",
            "resource-size": "362417",
            "created": "2010-06-10T12:07:26.000+0000",
            "resource-uri": "content:faade630a2a247468ca9a9b57303b437",
            "metacard-tags": [
                "resource"
            ],
            "checksum-algorithm": "Adler32",
            "metadata-content-type": "image/jpeg",
            "metacard-type": "ddf.metacard",
            "resource-download-url":
"https://{FQDN}:{PORT}services/catalog/sources/ddf.distribution/faade630a2a247468ca9a9b57
303b437?transform=resource",
            "title": "example.jpg",
            "source-id": "ddf.distribution",
            "effective": "2016-07-13T19:22:06.966+0000",
            "point-of-contact": "",
            "checksum": "dc7337c5",
            "modified": "2010-06-10T12:07:26.000+0000",
            "id": "faade630a2a247468ca9a9b57303b437"
        }
    }
},
],
"status": [
    {
        "hits": 1,
        "elapsed": 453,
        "reasons": [],
        "id": "ddf.distribution",
        "state": "SUCCEEDED",
        "results": 1
    }
],

```

```

    "successful": true
  },
  "channel": "/6f0e04e9-acd1-4935-b9dd-c83e770a36d5"
},
{
  "successful": true
},
{
  "channel": "/service/query",
  "id": "142",
  "successful": true
}

```

2.1.3.3.5. CometD Notifications

Notifications are messages that are sent to clients to inform them of some significant event happening. Clients must subscribe to a notification channel to receive these messages.

Notifications are published by the server on several notification channels depending on the type.

- subscribing to `/ddf/notifications/**` will cause the client to receive all notifications.
- subscribing to `/ddf/notifications/catalog/downloads` will cause the client to only receive notifications of downloads.

2.1.3.3.6. Using CometD Notifications

NOTE

The DDF Search UI serves as a reference implementation of how clients can use notifications.

Notifications are currently being utilized in the Catalog application for resource retrieval. When a user initiates a resource retrieval, the channel `/ddf/notification/catalog/downloads` is opened, where notifications indicating the progress of that resource download are sent. Any client interested in receiving these progress notifications must subscribe to that channel.

DDF starts downloading the resource to the client that requested it, a notification with a status of "Started" will be broadcast. If the resource download fails, a notification with a status of "Failed" will be broadcast. Or, if the resource download is being attempted again after a failure, "Retry" will be broadcast. When a notification is received, DDF Search UI displays a popup containing the contents of the notification, so a user is made aware of how their downloads are proceeding. Behind the scenes, the DDF Search UI invokes the REST endpoint to retrieve a resource.

In this request, it adds the query parameter "user" with the CometD session ID or the unique User ID as the value. This allows the CometD server to know which subscriber is interested in the notification. For example,

`https://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1w1qlo79j6tscii19jszwp9s2i55` notifications contain the following

information:

Table 5. Notification Contents

Property Name	Description	Always Included with Notification
<code>application</code>	Name of the application that caused the notification to be sent.	Yes
<code>id</code>	ID of the notification "thread" – Notifications about the same event should use the same id to allow clients to filter out notifications that may be outdated.	Yes
<code>title</code>	Resource/file name for resource retrieval.	Yes
<code>message</code>	Human-readable message containing status details.	Yes
<code>timestamp</code>	Timestamp in milliseconds when notification was sent.	Yes
<code>session</code>	CometD Session ID or unique User ID.	Yes

Example: Notification Message

```
"data": {
  "application": "Downloads",
  "title": "Product retrieval successful",
  "message": "The requested product was retrieved successfully
             and is available for download.",
  "id": "27ec3222af1144ff827a351b1962a236",
  "timestamp": "1403734355420",
  "user": "admin"
}
```

2.1.3.3.7. Receive Notifications

- If interested in retrieve resource notifications, a client must subscribe to the CometD channel `/ddf/notification/catalog/downloads`.
- If interested in all notification types, a client must subscribe to the CometD channel `/ddf/notification/**`
- A client will only receive notifications for resources they have requested.
- Standard UI is subscribed to all notifications of interest to that user/browser session: `/ddf/notification/**`
- See [Notification Contents](#) for the data that a notification contains.

2.1.3.3.8. Notification Events

Notifications are messages sent to clients to inform them of a significant event happening. Clients must subscribe to a notification channel to receive these messages.

2.1.3.3.9. Persistence of Notifications

Notifications are persisted between sessions, however due to the nature of CometD communications, they will not be visible at first connection/subscription to `/ddf/notifications/**`.

In order to retrieve notifications that were persisted or may have occurred since the previous session a client simply must publish an empty json message, `{}` to `/ddf/notifications`. This will return all existing notifications to the user.

2.1.3.3.10. Notification Operations Channel

Notification Operations are commands that change the behavior of future notifications. A notification operation is performed by publishing a list of commands to the CometD endpoint at `/notification/action`

Table 6. Operation Format

Map Key	Description	Value Type
<code>action</code>	Type of action to request. If a client publishes with the <code>remove</code> action, it dismisses the notification and makes it unavailable again when notifications are retrieved. "remove" is currently only used action.	String
<code>id</code>	ID of the notification to which the action relates	String

Example: Notification Operation Request

```
"data": [ {  
  "action": "remove",  
  "id": "27ec3222af1144ff827a351b1962a236"  
} ]
```

2.1.3.3.11. Activity Events Channel

To receive all activity updates, follow the instructions at [Subscribing to Notifications](#) and subscribe to `/ddf/activities/**`

Activity update messages follow a specific format when they are published to the activities channel. These messages contain a data map that encapsulates the activity information.

Table 7. CometD Activity Format

Property	Description	Value Type
<code>category</code>	Category of the activity	String
<code>id</code>	ID that uniquely identifies the activity that sent out the update. Not required to be unique per update.	String

Property	Description	Value Type
message	User-readable message that explains the current activity status	String
operations	<p>Map of operations that can be performed on this activity. If the value is a URL, the client should invoke the URL as a result of the user invoking the activity operation.</p> <p>If the value is not a URL, the client should send a message back to the server on the same topic with the operation name.</p> <p>Note: the DDF UI will interpret several values with special icons:</p> <ul style="list-style-type: none"> * cancel * download * remove 	JSON Map
progress	Percentage value of activity completion	String (Integer between 0 - 100 followed by a %)
status	Enumerated value that displays the current state of the activity	String + * STARTED * RUNNING * COMPLETED * STOPPED * PAUSED * FAILED
timestamp	Time that the activity update was sent	Date-Time
title	User-readable title for the activity update	String
subject	User who started the activity	String
bytes	Number of bytes the activity consumed (upload or download)	Positive Integer
session	The session ID of the user/subject	String
Custom Value	Additional keys can be inserted by the component sending the activity notification	Any JSON Type

Example: Activity update with custom 'bytes' field

```
data: {
  "category": "Product Retrieval",
  "id": "a62f6666-fc41-4a19-91f1-485e73a564b5",
  "message": "The requested product is being retrieved. Standby.",
  "operations": {
    "cancel" : true
  },
  "progress": "45",
  "status": "RUNNING",
  "timestamp": "1403801920875",
  "title": "Product downloading",
  "user": "admin",
  "bytes": 635084800
}
```

2.1.3.3.12. Activity Operations Channel

Different operations can be performed on activities through the `/service/action` channel.

Table 8. CometD Activity Format

Map Key	Description	Value Type	action
The requested action. This value is based on the operations map that comes in from an activity event.	String * "cancel" * "download" * "remove"	id	ID of the activity to which the requested operation relates

Example: Activity Operation Request Message

```
"data": [ {
  "action": "cancel",
  "id": "a62f6666-fc41-4a19-91f1-485e73a564b5"
} ]
```

2.1.4. CSW Endpoint

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

2.1.4.1. Installing CSW Endpoint

The CSW Endpoint is installed by default with a standard installation in the Spatial application.

2.1.4.2. Configuring CSW Endpoint

The CSW endpoint has no configurable properties. It can only be installed or uninstalled.


2.1.4.3. CSW Endpoint URL

The CSW endpoint is accessible from `https://{FQDN}:{PORT}/services/csw`.

2.1.4.3.1. CSW Endpoint Operations

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

GetCapabilities Operation

The **GetCapabilities** operation is meant to describe 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](#) .


GetCapabilities HTTP GET

The **HTTP GET** form of **GetCapabilities** uses query parameters via the following URL:

GetCapabilities KVP (Key-Value Pairs) Encoding

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

GetCapabilities HTTP POST

The **HTTP POST** form of **GetCapabilities** operates on the root CSW endpoint URL (`https://{FQDN}:{PORT}/services/csw`) with an XML message body that is defined by the **GetCapabilities** element of the [CSW-Discovery XML Schema](#) .

GetCapabilities XML 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:Operation>
  </ows:OperationsMetadata>
</csw:Capabilities>

```

```

<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="typeName">
    <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:Scalar_Capabilities>
</ogc:Filter_Capabilities>

```

```

    <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>

```

DescribeRecord Operation

The **describeRecord** operation retrieves the type definition used by metadata of one or more registered resource types. There are two request types one for **GET** and one for **POST**. Each request has the following common data parameters:

Namespace

In **POST** operations, namespaces are defined in the xml. In **GET** operations, namespaces are defined in a comma separated list of the form: `xmlns([prefix=]namespace-url)(,xmlns([prefix=]namespace-url))*`

Service

The service being used, in this case it is fixed at CSW.

Version

The version of the service being used (2.0.2).

OutputFormat

The requester wants the response to be in this intended output. Currently, only one format is supported (application/xml). If this parameter is supplied, it is validated against the known type. If this parameter is not supported, it passes through and returns the XML response upon success.

SchemaLanguage

The schema language from the request. This is validated against the known list of schema languages supported (refer to <http://www.w3.org/XML/Schema>).

DescribeRecord HTTP GET

The **HTTP GET** request differs from the **POST** request in that the typeName is a comma-separated list of

namespace prefix qualified types as strings (e.g., csw:Record,xyz:MyType). These prefixes are then matched against the prefix qualified namespaces in the request. This is converted to a list of QName(s). In this way, it behaves exactly as the post request that uses a list of QName(s) in the first place.

DescribeRecord KVP (Key-Value Pairs) Encoding

```
https://{FQDN}:{PORT}/services/csw?service=CSW&version=2.0.2&request=DescribeRecord&NAMESPACE=xmlns(http://www.opengis.net/cat/csw/2.0.2)&outputFormat=application/xml&schemaLanguage=http://www.w3.org/XML/Schema
```

DescribeRecord HTTP POST

The HTTP POST request `DescribeRecordType` has the `typeName` as a List of QName(s). The QNames are matched against the namespaces by prefix, if prefixes exist.

DescribeRecord XML Request

```
<?xml version="1.0" ?>
<DescribeRecord
  version="2.0.2"
  service="CSW"
  outputFormat="application/xml"
  schemaLanguage="http://www.w3.org/XML/Schema"
  xmlns="http://www.opengis.net/cat/csw/2.0.2">
</DescribeRecord>
```

DescribeRecord Sample Response (application/xml)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:DescribeRecordResponse 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" ns10:schemaLocation=
"http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
  <csw:SchemaComponent targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
schemaLanguage="http://www.w3.org/XML/Schema">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault=
"qualified" id="csw-record" targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
version="2.0.2">
      <xsd:annotation>
        <xsd:appinfo>
          <dc:identifier>
http://schemas.opengis.net/csw/2.0.2/record.xsd</dc:identifier>
          </xsd:appinfo>
```

```
<xsd:documentation xml:lang="en">
```

This schema defines the basic record types that must be supported by all CSW implementations. These correspond to full, summary, and brief views based on DCMI metadata terms.

```
</xsd:documentation>
```

```
</xsd:annotation>
```

```
<xsd:import namespace="http://purl.org/dc/terms/" schemaLocation="rec-dcterms.xsd"/>
```

```
<xsd:import namespace="http://purl.org/dc/elements/1.1/" schemaLocation="rec-dcmes.xsd"/>
```

```
<xsd:import namespace="http://www.opengis.net/ows" schemaLocation="
../../ows/1.0.0/owsAll.xsd"/>
```

```
<xsd:element abstract="true" id="AbstractRecord" name="AbstractRecord" type=
"csw:AbstractRecordType"/>
```

```
<xsd:complexType abstract="true" id="AbstractRecordType" name=
"AbstractRecordType"/>
```

```
<xsd:element name="DCMIRecord" substitutionGroup="csw:AbstractRecord" type=
"csw:DCMIRecordType"/>
```

```
<xsd:complexType name="DCMIRecordType">
```

```
<xsd:annotation>
```

```
<xsd:documentation xml:lang="en">
```

This type encapsulates all of the standard DCMI metadata terms, including the Dublin Core refinements; these terms may be mapped to the profile-specific information model.

```
</xsd:documentation>
```

```
</xsd:annotation>
```

```
<xsd:complexContent>
```

```
<xsd:extension base="csw:AbstractRecordType">
```

```
<xsd:sequence>
```

```
<xsd:group ref="dct:DCMI-terms"/>
```

```
</xsd:sequence>
```

```
</xsd:extension>
```

```
</xsd:complexContent>
```

```
</xsd:complexType>
```

```
<xsd:element name="BriefRecord" substitutionGroup="csw:AbstractRecord" type=
"csw:BriefRecordType"/>
```

```
<xsd:complexType final="#all" name="BriefRecordType">
```

```
<xsd:annotation>
```

```
<xsd:documentation xml:lang="en">
```

This type defines a brief representation of the common record format. It extends AbstractRecordType to include only the dc:identifier and dc:type properties.

```

</xsd:documentation>

</xsd:annotation>
<xsd:complexContent>
  <xsd:extension base="csw:AbstractRecordType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:identifier"/>
      <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:title"/>
      <xsd:element minOccurs="0" ref="dc:type"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>

    </xsd:sequence>

  </xsd:extension>

</xsd:complexContent>

</xsd:complexType>
<xsd:element name="SummaryRecord" substitutionGroup="csw:AbstractRecord"
type="csw:SummaryRecordType"/>
<xsd:complexType final="#all" name="SummaryRecordType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This type defines a summary representation of the common record
      format. It extends AbstractRecordType to include the core
      properties.
    </xsd:documentation>

    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="csw:AbstractRecordType">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:identifier"/>
          <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:title"/>
          <xsd:element minOccurs="0" ref="dc:type"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:subject"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:format"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:relation"/>
          <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:modified"/>

```

```

        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:abstract"/>
        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:spatial"/>
        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>

    </xsd:sequence>

</xsd:extension>

</xsd:complexContent>

</xsd:complexType>
<xsd:element name="Record" substitutionGroup="csw:AbstractRecord" type=
"csw:RecordType"/>
<xsd:complexType final="#all" name="RecordType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            This type extends DCMIRecordType to add ows:BoundingBox;
            it may be used to specify a spatial envelope for the
            catalogued resource.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="csw:DCMIRecordType">
            <xsd:sequence>
                <xsd:element maxOccurs="unbounded" minOccurs="0" name=
"AnyText" type="csw:EmptyType"/>
                <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>

</xsd:complexType>
<xsd:complexType name="EmptyType"/>
</xsd:schema>
</csw:SchemaComponent>
<csw:SchemaComponent targetNamespace="http://www.isotc211.org/2005/gmd"
schemaLanguage="http://www.w3.org/XML/Schema">
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gco=
"http://www.isotc211.org/2005/gco" xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified"

```

```

targetNamespace="http://www.isotc211.org/2005/gmd" version="2012-07-13">
  <xs:annotation>
    <xs:documentation>
      Geographic MetaData (GMD) extensible markup language is a component of the
      XML Schema Implementation of Geographic Information Metadata documented in ISO/TS
      19139:2007. GMD includes all the definitions of http://www.isotc211.org/2005/gmd
      namespace. The root document of this namespace is the file gmd.xsd. This
      identification.xsd schema implements the UML conceptual schema defined in A.2.2 of ISO
      19115:2003. It contains the implementation of the following classes: MD_Identification,
      MD_BrowseGraphic, MD_DataIdentification, MD_ServiceIdentification,
      MD_RepresentativeFraction, MD_Usage, MD_Keywords, DS_Association,
      MD_AggregateInformation, MD_CharacterSetCode, MD_SpatialRepresentationTypeCode,
      MD_TopicCategoryCode, MD_ProgressCode, MD_KeywordTypeCode, DS_AssociationTypeCode,
      DS_InitiativeTypeCode, MD_ResolutionTypeCode.
    </xs:documentation>

    </xs:annotation>
    <xs:import namespace="http://www.isotc211.org/2005/gco" schemaLocation=
"http://schemas.opengis.net/iso/19139/20070417/gco/gco.xsd"/>
    <xs:include schemaLocation="gmd.xsd"/>
    <xs:include schemaLocation="constraints.xsd"/>
    <xs:include schemaLocation="distribution.xsd"/>
    <xs:include schemaLocation="maintenance.xsd"/>
    <xs:complexType abstract="true" name="AbstractMD_Identification_Type">
      <xs:annotation>
        <xs:documentation>Basic information about data</xs:documentation>

      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
          <xs:sequence>
            <xs:element name="citation" type=
"gm:CI_Citation_PropertyType"/>
            <xs:element name="abstract" type=
"gm:CharacterString_PropertyType"/>
            <xs:element minOccurs="0" name="purpose" type=
"gm:CharacterString_PropertyType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="credit"
type="gm:CharacterString_PropertyType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="status"
type="gm:MD_ProgressCode_PropertyType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name=
"pointOfContact" type="gm:CI_ResponsibleParty_PropertyType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name=
"resourceMaintenance" type="gm:MD_MaintenanceInformation_PropertyType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name=
"graphicOverview" type="gm:MD_BrowseGraphic_PropertyType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name=

```

```

"resourceFormat" type="gmd:MD_Format_PropertyType"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name=
"descriptiveKeywords" type="gmd:MD_Keywords_PropertyType"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name=
"resourceSpecificUsage" type="gmd:MD_Usage_PropertyType"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name=
"resourceConstraints" type="gmd:MD_Constraints_PropertyType"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name=
"aggregationInfo" type="gmd:MD_AggregateInformation_PropertyType"/>

    </xs:sequence>

</xs:extension>

</xs:complexContent>

</xs:complexType>
<xs:element abstract="true" name="AbstractMD_Identification" type=
"gmd:AbstractMD_Identification_Type"/>
<xs:complexType name="MD_Identification_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:AbstractMD_Identification"/>

    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>

</xs:complexType>
<xs:complexType name="MD_BrowseGraphic_Type">
    <xs:annotation>
        <xs:documentation>
            Graphic that provides an illustration of the dataset (should include a
            legend for the graphic)
        </xs:documentation>

    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence>
                <xs:element name="fileName" type=
                "gco:CharacterString_PropertyType"/>
                <xs:element minOccurs="0" name="fileDescription" type=
                "gco:CharacterString_PropertyType"/>
                <xs:element minOccurs="0" name="fileType" type=
                "gco:CharacterString_PropertyType"/>

            </xs:sequence>

```

```

        </xs:extension>

        </xs:complexContent>

        </xs:complexType>
        <xs:element name="MD_BrowseGraphic" type="gmd:MD_BrowseGraphic_Type"/>
        <xs:complexType name="MD_BrowseGraphic_PropertyType">
            <xs:sequence minOccurs="0">
                <xs:element ref="gmd:MD_BrowseGraphic"/>
            </xs:sequence>
            <xs:attributeGroup ref="gco:ObjectReference"/>
            <xs:attribute ref="gco:nilReason"/>
        </xs:complexType>
        <xs:complexType name="MD_DataIdentification_Type">
            <xs:complexContent>
                <xs:extension base="gmd:AbstractMD_Identification_Type">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="
spatialRepresentationType" type="gmd:MD_SpatialRepresentationTypeCode_PropertyType"/>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="
spatialResolution" type="gmd:MD_Resolution_PropertyType"/>
                        <xs:element maxOccurs="unbounded" name="language" type="
gco:CharacterString_PropertyType"/>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="
characterSet" type="gmd:MD_CharacterSetCode_PropertyType"/>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="
topicCategory" type="gmd:MD_TopicCategoryCode_PropertyType"/>
                        <xs:element minOccurs="0" name="environmentDescription" type="
gco:CharacterString_PropertyType"/>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="extent"
type="gmd:EX_Extent_PropertyType"/>
                        <xs:element minOccurs="0" name="supplementalInformation"
type="gco:CharacterString_PropertyType"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>

            </xs:complexType>
            <xs:element name="MD_DataIdentification" substitutionGroup=
"gmd:AbstractMD_Identification" type="gmd:MD_DataIdentification_Type"/>
            <xs:complexType name="MD_DataIdentification_PropertyType">
                <xs:sequence minOccurs="0">
                    <xs:element ref="gmd:MD_DataIdentification"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

```

```

        </xs:sequence>
        <xs:attributeGroup ref="gco:ObjectReference"/>
        <xs:attribute ref="gco:nilReason"/>

    </xs:complexType>
    <xs:complexType name="MD_ServiceIdentification_Type">
        <xs:annotation>
            <xs:documentation>See 19119 for further info</xs:documentation>

        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="gmd:AbstractMD_Identification_Type"/>

        </xs:complexContent>

    </xs:complexType>
    <xs:element name="MD_ServiceIdentification" substitutionGroup=
    "gmd:AbstractMD_Identification" type="gmd:MD_ServiceIdentification_Type"/>
    <xs:complexType name="MD_ServiceIdentification_PropertyType">
        <xs:sequence minOccurs="0">
            <xs:element ref="gmd:MD_ServiceIdentification"/>

        </xs:sequence>
        <xs:attributeGroup ref="gco:ObjectReference"/>
        <xs:attribute ref="gco:nilReason"/>

    </xs:complexType>
    <xs:complexType name="MD_RepresentativeFraction_Type">
        <xs:complexContent>
            <xs:extension base="gco:AbstractObject_Type">
                <xs:sequence>
                    <xs:element name="denominator" type="
gco:Integer_PropertyType"/>

                </xs:sequence>

            </xs:extension>

        </xs:complexContent>

    </xs:complexType>
    <xs:element name="MD_RepresentativeFraction" type=
    "gmd:MD_RepresentativeFraction_Type"/>
    <xs:complexType name="MD_RepresentativeFraction_PropertyType">
        <xs:sequence minOccurs="0">
            <xs:element ref="gmd:MD_RepresentativeFraction"/>

```



```

        </xs:sequence>
        <xs:attributeGroup ref="gco:ObjectReference"/>
        <xs:attribute ref="gco:nilReason"/>

</xs:complexType>
<xs:complexType name="MD_Usage_Type">
    <xs:annotation>
        <xs:documentation>
            Brief description of ways in which the dataset is currently used.
        </xs:documentation>

    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence>
                <xs:element name="specificUsage" type="
gco:CharacterString_PropertyType"/>
                <xs:element minOccurs="0" name="usageDateTime" type="
gco:DateTime_PropertyType"/>
                <xs:element minOccurs="0" name="userDeterminedLimitations"
type="gco:CharacterString_PropertyType"/>
                <xs:element maxOccurs="unbounded" name="userContactInfo"
type="gmd:CI_ResponsibleParty_PropertyType"/>

            </xs:sequence>

        </xs:extension>

    </xs:complexContent>

</xs:complexType>
<xs:element name="MD_Usage" type="gmd:MD_Usage_Type"/>
<xs:complexType name="MD_Usage_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:MD_Usage"/>

    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>

</xs:complexType>
<xs:complexType name="MD_Keywords_Type">
    <xs:annotation>
        <xs:documentation>Keywords, their type and reference
source</xs:documentation>

    </xs:annotation>
    <xs:complexContent>

```

```

        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence>
                <xs:element maxOccurs="unbounded" name="keyword" type=
"gco:CharacterString_PropertyType"/>
                <xs:element minOccurs="0" name="type" type=
"gmd:MD_KeywordTypeCode_PropertyType"/>
                <xs:element minOccurs="0" name="thesaurusName" type=
"gmd:CI_Citation_PropertyType"/>
            </xs:sequence>
        </xs:extension>

    </xs:complexContent>

</xs:complexType>
<xs:element name="MD_Keywords" type="gmd:MD_Keywords_Type"/>
<xs:complexType name="MD_Keywords_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:MD_Keywords"/>
    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:complexType name="DS_Association_Type">
    <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence/>
        </xs:extension>
    </xs:complexContent>

</xs:complexType>
<xs:element name="DS_Association" type="gmd:DS_Association_Type"/>
<xs:complexType name="DS_Association_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:DS_Association"/>
    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:complexType name="MD_AggregateInformation_Type">
    <xs:annotation>

```

```

        <xs:documentation>Encapsulates the dataset aggregation
information</xs:documentation>

    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="gco:AbstractObject_Type">
            <xs:sequence>
                <xs:element minOccurs="0" name="aggregateDataSetName" type=
"gmd:CI_Citation_PropertyType"/>
                <xs:element minOccurs="0" name="aggregateDataSetIdentifier"
type="gmd:MD_Identifier_PropertyType"/>
                <xs:element name="associationType" type=
"gmd:DS_AssociationTypeCode_PropertyType"/>
                <xs:element minOccurs="0" name="initiativeType" type=
"gmd:DS_InitiativeTypeCode_PropertyType"/>

            </xs:sequence>

        </xs:extension>

    </xs:complexContent>

</xs:complexType>
<xs:element name="MD_AggregateInformation" type=
"gmd:MD_AggregateInformation_Type"/>
<xs:complexType name="MD_AggregateInformation_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:MD_AggregateInformation"/>

    </xs:sequence>
    <xs:attributeGroup ref="gco:ObjectReference"/>
    <xs:attribute ref="gco:nilReason"/>

</xs:complexType>
<xs:complexType name="MD_Resolution_Type">
    <xs:choice>
        <xs:element name="equivalentScale" type=
"gmd:MD_RepresentativeFraction_PropertyType"/>
        <xs:element name="distance" type="gco:Distance_PropertyType"/>

    </xs:choice>

</xs:complexType>
<xs:element name="MD_Resolution" type="gmd:MD_Resolution_Type"/>
<xs:complexType name="MD_Resolution_PropertyType">
    <xs:sequence minOccurs="0">
        <xs:element ref="gmd:MD_Resolution"/>

```

```

        </xs:sequence>
        <xs:attribute ref="gco:nilReason"/>

    </xs:complexType>
    <xs:simpleType name="MD_TopicCategoryCode_Type">
        <xs:annotation>
            <xs:documentation>
                High-level geospatial data thematic classification to assist in the
                grouping and search of available geospatial datasets
            </xs:documentation>

        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="farming"/>
            <xs:enumeration value="biota"/>
            <xs:enumeration value="boundaries"/>
            <xs:enumeration value="climatologyMeteorologyAtmosphere"/>
            <xs:enumeration value="economy"/>
            <xs:enumeration value="elevation"/>
            <xs:enumeration value="environment"/>
            <xs:enumeration value="geoscientificInformation"/>
            <xs:enumeration value="health"/>
            <xs:enumeration value="imageryBaseMapsEarthCover"/>
            <xs:enumeration value="inlandWaters"/>
            <xs:enumeration value="intelligenceMilitary"/>
            <xs:enumeration value="location"/>
            <xs:enumeration value="oceans"/>
            <xs:enumeration value="planningCadastre"/>
            <xs:enumeration value="society"/>
            <xs:enumeration value="structure"/>
            <xs:enumeration value="transportation"/>
            <xs:enumeration value="utilitiesCommunication"/>

        </xs:restriction>

    </xs:simpleType>
    <xs:element name="MD_TopicCategoryCode" substitutionGroup=
    "gco:CharacterString" type="gmd:MD_TopicCategoryCode_Type"/>
    <xs:complexType name="MD_TopicCategoryCode_PropertyType">
        <xs:sequence minOccurs="0">
            <xs:element ref="gmd:MD_TopicCategoryCode"/>

        </xs:sequence>
        <xs:attribute ref="gco:nilReason"/>

    </xs:complexType>
    <xs:element name="MD_CharacterSetCode" substitutionGroup="
    gco:CharacterString" type="gco:CodeListValue_Type"/>

```

```

<xs:complexType name="MD_CharacterSetCode_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:MD_CharacterSetCode"/>
  </xs:sequence>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:element name="MD_SpatialRepresentationTypeCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_SpatialRepresentationTypeCode_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:MD_SpatialRepresentationTypeCode"/>
  </xs:sequence>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:element name="MD_ProgressCode" substitutionGroup="gco:CharacterString"
type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_ProgressCode_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:MD_ProgressCode"/>
  </xs:sequence>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:element name="MD_KeywordTypeCode" substitutionGroup="gco:CharacterString"
type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_KeywordTypeCode_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:MD_KeywordTypeCode"/>
  </xs:sequence>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:element name="DS_AssociationTypeCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>
<xs:complexType name="DS_AssociationTypeCode_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:DS_AssociationTypeCode"/>
  </xs:sequence>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType>
<xs:element name="DS_InitiativeTypeCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>

```

```

        <xs:complexType name="DS_InitiativeTypeCode_PropertyType">
            <xs:sequence minOccurs="0">
                <xs:element ref="gmd:DS_InitiativeTypeCode"/>
            </xs:sequence>
            <xs:attribute ref="gco:nilReason"/>
        </xs:complexType>
    </xs:schema>
</csw:SchemaComponent>
</csw:DescribeRecordResponse>

```

DescribeRecord HTTP POST With TypeNames

The HTTP POST request **DescribeRecordType** has the **typeName** as a List of QName(s). The QNames are matched against the namespaces by prefix, if prefixes exist. **.DescribeRecord XML Request**

```

<?xml version="1.0" ?>
<DescribeRecord
  version="2.0.2"
  service="CSW"
  schemaLanguage="http://www.w3.org/XML/Schema"
  xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <TypeName>csw:Record</TypeName>
</DescribeRecord>

```

DescribeRecord Sample Response (application/xml)

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:DescribeRecordResponse 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" ns10:schemaLocation=
"http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:SchemaComponent targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
schemaLanguage="http://www.w3.org/XML/Schema">
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault=
"qualified" id="csw-record" targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
version="2.0.2">
            <xsd:annotation>
                <xsd:appinfo>
                    <dc:identifier>
http://schemas.opengis.net/csw/2.0.2/record.xsd</dc:identifier>

                </xsd:appinfo>
                <xsd:documentation xml:lang="en">

```

This schema defines the basic record types that must be supported by all CSW implementations. These correspond to full, summary, and brief views based on DCMI metadata terms.

```
</xsd:documentation>
```

```
</xsd:annotation>
```

```
<xsd:import namespace="http://purl.org/dc/terms/" schemaLocation="rec-  
dcterms.xsd"/>
```

```
<xsd:import namespace="http://purl.org/dc/elements/1.1/" schemaLocation="rec-  
dcmes.xsd"/>
```

```
<xsd:import namespace="http://www.opengis.net/ows" schemaLocation=  
"../../ows/1.0.0/owsAll.xsd"/>
```

```
<xsd:element abstract="true" id="AbstractRecord" name="AbstractRecord" type=  
"csw:AbstractRecordType"/>
```

```
<xsd:complexType abstract="true" id="AbstractRecordType" name=  
"AbstractRecordType"/>
```

```
<xsd:element name="DCMIRecord" substitutionGroup="csw:AbstractRecord" type=  
"csw:DCMIRecordType"/>
```

```
<xsd:complexType name="DCMIRecordType">
```

```
<xsd:annotation>
```

```
<xsd:documentation xml:lang="en">
```

This type encapsulates all of the standard DCMI metadata terms, including the Dublin Core refinements; these terms may be mapped to the profile-specific information model.

```
</xsd:documentation>
```

```
</xsd:annotation>
```

```
<xsd:complexContent>
```

```
<xsd:extension base="csw:AbstractRecordType">
```

```
<xsd:sequence>
```

```
<xsd:group ref="dct:DCMI-terms"/>
```

```
</xsd:sequence>
```

```
</xsd:extension>
```

```
</xsd:complexContent>
```

```
</xsd:complexType>
```

```
<xsd:element name="BriefRecord" substitutionGroup="csw:AbstractRecord" type=  
"csw:BriefRecordType"/>
```

```
<xsd:complexType final="#all" name="BriefRecordType">
```

```
<xsd:annotation>
```

```
<xsd:documentation xml:lang="en">
```

This type defines a brief representation of the common record format. It extends AbstractRecordType to include only the dc:identifier and dc:type properties.

```
</xsd:documentation>
```

```

        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="csw:AbstractRecordType">
                <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:identifier"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:title"/>
                    <xsd:element minOccurs="0" ref="dc:type"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>

                </xsd:sequence>

            </xsd:extension>

        </xsd:complexContent>

    </xsd:complexType>
    <xsd:element name="SummaryRecord" substitutionGroup="csw:AbstractRecord"
type="csw:SummaryRecordType"/>
    <xsd:complexType final="#all" name="SummaryRecordType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                This type defines a summary representation of the common record
                format. It extends AbstractRecordType to include the core
                properties.
            </xsd:documentation>

            </xsd:annotation>
            <xsd:complexContent>
                <xsd:extension base="csw:AbstractRecordType">
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:identifier"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:title"/>
                        <xsd:element minOccurs="0" ref="dc:type"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:subject"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:format"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:relation"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:modified"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" ref=

```



```

"dct:abstract"/>
                                <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:spatial"/>
                                <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>

                                </xsd:sequence>

                                </xsd:extension>

                                </xsd:complexContent>

                                </xsd:complexType>
                                <xsd:element name="Record" substitutionGroup="csw:AbstractRecord" type=
"csw:RecordType"/>
                                <xsd:complexType final="#all" name="RecordType">
                                    <xsd:annotation>
                                        <xsd:documentation xml:lang="en">
                                            This type extends DCMIRecordType to add ows:BoundingBox;
                                            it may be used to specify a spatial envelope for the
                                            catalogued resource.
                                        </xsd:documentation>

                                        </xsd:annotation>
                                        <xsd:complexContent>
                                            <xsd:extension base="csw:DCMIRecordType">
                                                <xsd:sequence>
                                                    <xsd:element maxOccurs="unbounded" minOccurs="0" name=
"AnyText" type="csw:EmptyType"/>
                                                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>

                                                </xsd:sequence>

                                                </xsd:extension>

                                            </xsd:complexContent>

                                        </xsd:complexType>
                                        <xsd:complexType name="EmptyType"/>
                                    </xsd:schema>
                                </csw:SchemaComponent>
</csw:DescribeRecordResponse>

```

GetRecords Operation

The **GetRecords** operation is the principal means of searching the catalog. The matching entries may be included with the response. The client may assign a **requestId** (absolute URI). A distributed search is

performed if the **DistributedSearch** element is present and the catalog is a member of a federation. Profiles may allow alternative query expressions. There are two types of request types: one for **GET** and one for **POST**. Each request has the following common data parameters:

Namespace

In POST operations, namespaces are defined in the XML. In GET operations, namespaces are defined in a comma-separated list of the form `xmlns([prefix=]namespace-url)(,xmlns([pref::=]namespace-url))*`.

Service

The service being used, in this case it is fixed at CSW.

Version

The version of the service being used (2.0.2).

OutputFormat

The requester wants the response to be in this intended output. Currently, only one format is supported (application/xml). If this parameter is supplied, it is validated against the known type. If this parameter is not supported, it passes through and returns the XML response upon success.

OutputSchema

The OutputSchema indicates which schema shall be used to generate the response to the GetRecords operation. The supported output schemas are listed in the GetCapabilities response.

ElementSetName

CodeList with allowed values of “brief”, “summary”, or “full”. The default value is "summary". The predefined set names of “brief”, “summary”, and “full” represent different levels of detail for the source record. "Brief" represents the least amount of detail, and "full" represents all the metadata record elements.

IMPORTANT

The CSW Endpoint expects all geospatial filters using the EPSG:4326 CRS to use "longitude then latitude" coordinate ordering. Similarly, unless the output schema explicitly states otherwise, the GetRecordsResponse will use the same coordinate ordering.

GetRecords HTTP GET

The **HTTP GET** request differs from the **POST** request in that it has the "typeNames" as a comma-separated list of namespace prefix qualified types as strings. For example `csw:Record,xyz:MyType`. These prefixes are then matched against the prefix qualified namespaces in the request. This is converted to a list QName(s). In this way, it behaves exactly as the post request that uses a list of QName(s) in the first place.

GetRecords *KVP (Key-Value Pairs) Encoding*

```
https://{FQDN}:{PORT}/services/csw?service=CSW&version=2.0.2&request=GetRecords&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPACE=xmlns(csw=http://www.opengis.net/cat/csw/2.0.2)&resultType=results&typeNames=csw:Record&ElementSetName=brief&ConstraintLanguage=CQL_TEXT&constraint=AnyText Like '%25'
```

GetRecords *HTTP POST*

The **HTTP POST** request GetRecords has the **typeNames** as a List of QName(s). The QNames are matched against the namespaces by prefix, if prefixes exist.

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">
  <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 *Specific Source*

It is possible to query a **Specific Source** by specifying a query for that source-id. The valid **source-id**'s will be listed in the **FederatedCatalogs** section of the **GetCapabilities** Response. The example below shows how to query for a specific source.

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.

GetRecords XML 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>
```

GetRecords Sample Response (application/xml)

```
<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="41" numberOfRecordsReturned="4"
nextRecord="5" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">
  <csw:SummaryRecord>
    <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>
    <dc:title>Product10</dc:title>
    <dc:type>pdf</dc:type>
    <dc:modified>2014-02-19T15:22:51.563-05:00</dc: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:SummaryRecord>
    <dc:identifier>c607440db9b0407e92000d9260d35444</dc:identifier>
    <dc:title>Product03</dc:title>
    <dc:type>pdf</dc:type>
    <dc:modified>2014-02-19T15:22:51.563-05:00</dc:modified>
    <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
      <ows:LowerCorner>6.0 3.0</ows:LowerCorner>
      <ows:UpperCorner>6.0 3.0</ows:UpperCorner>
    </ows:BoundingBox>
  </csw:SummaryRecord>
  <csw:SummaryRecord>
    <dc:identifier>034cc757abd645f0abe6acaccfe194de</dc:identifier>
    <dc:title>Product03</dc:title>
    <dc:type>pdf</dc:type>
    <dc:modified>2014-02-19T15:22:51.563-05:00</dc:modified>
    <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
      <ows:LowerCorner>6.0 3.0</ows:LowerCorner>
      <ows:UpperCorner>6.0 3.0</ows:UpperCorner>
    </ows:BoundingBox>
  </csw:SummaryRecord>
  <csw:SummaryRecord>
    <dc:identifier>5d6e987bd6084bd4919d06b63b77a007</dc:identifier>
    <dc:title>Product01</dc:title>
    <dc:type>pdf</dc:type>
    <dc:modified>2014-02-19T15:22:51.563-05:00</dc:modified>
    <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
      <ows:LowerCorner>2.0 1.0</ows:LowerCorner>
      <ows:UpperCorner>2.0 1.0</ows:UpperCorner>
    </ows:BoundingBox>
  </csw:SummaryRecord>
</csw:SearchResults>
</csw:GetRecordsResponse>

```

GetRecords GMD OutputSchema

It is possible to receive a response to a **GetRecords** query that conforms to the GMD specification.

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"
  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>prod%</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="2016-03-23T11:31:34.531-06:00"/>
  <csw:SearchResults numberOfRecordsMatched="7" numberOfRecordsReturned="1" nextRecord
="2" recordSchema="http://www.isotc211.org/2005/gmd" elementSet="summary">
    <MD_Metadata xmlns="http://www.isotc211.org/2005/gmd" xmlns:gco=
"http://www.isotc211.org/2005/gco">
      <fileIdentifier>
        <gco:CharacterString>
d5f6acd5ccf34d18af5192c38a276b12</gco:CharacterString>
        </fileIdentifier>
```

```

<hierarchyLevel>
  <MD_ScopeCode codeListValue="nitr" codeList="urn:catalog:metacard"/>
</hierarchyLevel>
<contact/>
<dateStamp>
  <gco:DateTime>2015-03-04T17:23:42.332-07:00</gco:DateTime>
</dateStamp>
<identificationInfo>
  <MD_DataIdentification>
    <citation>
      <CI_Citation>
        <title>
          <gco:CharacterString>product.ntr</gco:CharacterString>
        </title>
        <date>
          <CI_Date>
            <date>
              <gco:DateTime>2015-03-04T17:23:42.332-
07:00</gco:DateTime>
            </date>
            <dateType>
              <CI_DateTypeCode codeList="urn:catalog:metacard"
codeListValue="created"/>
            </dateType>
          </CI_Date>
        </date>
      </CI_Citation>
    </citation>
    <abstract>
      <gco:CharacterString></gco:CharacterString>
    </abstract>
    <pointOfContact>
      <CI_ResponsibleParty>
        <organisationName>
          <gco:CharacterString></gco:CharacterString>
        </organisationName>
        <role/>
      </CI_ResponsibleParty>
    </pointOfContact>
    <language>
      <gco:CharacterString>en</gco:CharacterString>
    </language>
    <extent>
      <EX_Extent>
        <geographicElement>
          <EX_GeographicBoundingBox>
            <westBoundLongitude>
              <gco:Decimal>32.975277</gco:Decimal>

```

```

        </westBoundLongitude>
        <eastBoundLongitude>
            <gco:Decimal>32.996944</gco:Decimal>
        </eastBoundLongitude>
        <southBoundLatitude>
            <gco:Decimal>32.305</gco:Decimal>
        </southBoundLatitude>
        <northBoundLatitude>
            <gco:Decimal>32.323333</gco:Decimal>
        </northBoundLatitude>
    </EX_GeographicBoundingBox>
</geographicElement>
</EX_Extent>
</extent>
</MD_DataIdentification>
</identificationInfo>
<distributionInfo>
    <MD_Distribution>
        <distributor>
            <MD_Distributor>
                <distributorContact/>
                <distributorTransferOptions>
                    <MD_DigitalTransferOptions>
                        <onLine>
                            <CI_OnlineResource>
                                <linkage>
                                    <URL>http://example.com</URL>
                                </linkage>
                            </CI_OnlineResource>
                        </onLine>
                    </MD_DigitalTransferOptions>
                </distributorTransferOptions>
            </MD_Distributor>
        </distributor>
    </MD_Distribution>
</distributionInfo>
</MD_Metadata>
</csw:SearchResults>
</csw:GetRecordsResponse>

```

GetRecords XML Request using 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.

GetRecords XML Request - UTM Northern Hemisphere Zone 36

```
<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 typeName="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>
```

GetRecordById Operation

The **GetRecordById** operation request retrieves the default representation of catalog records using their identifier. This operation presumes that a previous query has been performed in order to obtain the identifiers that may be used with this operation. For example, records returned by a **GetRecords** operation may contain references to other records in the catalog that may be retrieved using the **GetRecordById** operation. This operation is also a subset of the **GetRecords** operation and is included as a convenient short form for retrieving and linking to records in a catalog.

Clients can also retrieve products from the catalog using the **GetRecordById** operation. The client sets the output schema to <http://www.iana.org/assignments/media-types/application/octet-stream> and the output format to **application/octet-stream** within the request. The endpoint will do the following: check that only one Id is provided, otherwise an error will occur as multiple products cannot be retrieved. If both output format and output schema are set to values mentioned above, the catalog

framework will retrieve the resource for that Id. The HTTP content type is then set to the resource's MIME type and the data is sent out. The endpoint also supports the resumption of partial downloads. This would typically occur at the request of a browser when a download was prematurely terminated.

There are two request types: one for **GET** and one for **POST**. Each request has the following common data parameters:

Namespace

In POST operations, namespaces are defined in the XML. In GET operations namespaces are defined in a comma separated list of the form: `xmlns([prefix=]namespace-url)(,xmlns([prefix=]namespace-url))*`.

Service

The service being used, in this case it is fixed at "CSW".

Version

The version of the service being used (2.0.2).

OutputFormat

The requester wants the response to be in this intended output. Currently, two output formats are supported: `application/xml` for retrieving records, and `application/octet-stream` for retrieving a product. If this parameter is supplied, it is validated against the known type. If this parameter is not supported, it passes through and returns the XML response upon success.

OutputSchema

The OutputSchema indicates which schema shall be used to generate the response to the GetRecordById operation. The supported output schemas are listed in the GetCapabilities response.

ElementSetName

CodeList with allowed values of "brief", "summary", or "full". The default value is "summary". The predefined set names of "brief", "summary", and "full" represent different levels of detail for the source record. "Brief" represents the least amount of detail, and "full" represents all the metadata record elements.

Id

The Id parameter is a comma-separated list of record identifiers for the records that CSW returns to the client. In the XML encoding, one or more <Id> elements may be used to specify the record identifier to be retrieved.

GetRecordById HTTP GET KVP (Key-Value Pairs) Encoding

```
https://{FQDN}:{PORT}/services/csw?service=CSW&version=2.0.2&request=GetRecordById&NAMESPACE=xmlns="http://www.opengis.net/cat/csw/2.0.2"&ElementSetName=full&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&id=fd7ff1535dfe47db8793b550d4170424,ba908634c0eb439b84b5d9c42af1f871
```

GetRecordById HTTP POST

```
<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>182fb33103414e5cbb06f8693b526239</Id>
  <Id>c607440db9b0407e92000d9260d35444</Id>
</GetRecordById>
```

GetRecordByIdResponse *Sample Response (application/xml)*

```
<csw:GetRecordByIdResponse 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:Record>
    <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>
    <dct:bibliographicCitation>182fb33103414e5cbb06f8693b526239</dct:bibliographicCitation>
    <dc:title>Product10</dc:title>
    <dct:alternative>Product10</dct:alternative>
    <dc:type>pdf</dc:type>
    <dc:date>2014-02-19T15:22:51.563-05:00</dc:date>
    <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>
    <dct:created>2014-02-19T15:22:51.563-05:00</dct:created>
    <dct:dateAccepted>2014-02-19T15:22:51.563-05:00</dct:dateAccepted>
    <dct:dateCopyrighted>2014-02-19T15:22:51.563-05:00</dct:dateCopyrighted>
    <dct:dateSubmitted>2014-02-19T15:22:51.563-05:00</dct:dateSubmitted>
    <dct:issued>2014-02-19T15:22:51.563-05:00</dct:issued>
    <dc:source>ddf.distribution</dc:source>
    <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:Record>
  <csw:Record>
    <dc:identifier>c607440db9b0407e92000d9260d35444</dc:identifier>
    <dct:bibliographicCitation>c607440db9b0407e92000d9260d35444</dct:bibliographicCitation>
    <dc:title>Product03</dc:title>
    <dct:alternative>Product03</dct:alternative>
    <dc:type>pdf</dc:type>
    <dc:date>2014-02-19T15:22:51.563-05:00</dc:date>
    <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>
    <dct:created>2014-02-19T15:22:51.563-05:00</dct:created>
    <dct:dateAccepted>2014-02-19T15:22:51.563-05:00</dct:dateAccepted>
    <dct:dateCopyrighted>2014-02-19T15:22:51.563-05:00</dct:dateCopyrighted>
    <dct:dateSubmitted>2014-02-19T15:22:51.563-05:00</dct:dateSubmitted>
    <dct:issued>2014-02-19T15:22:51.563-05:00</dct:issued>
    <dc:source>ddf.distribution</dc:source>
    <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
      <ows:LowerCorner>6.0 3.0</ows:LowerCorner>
      <ows:UpperCorner>6.0 3.0</ows:UpperCorner>
    </ows:BoundingBox>
  </csw:Record>
</csw:GetRecordByIdResponse>
```

Table 9. 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
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

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
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
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

2.1.4.3.2. Transaction Operations

Transactions define the operations for creating, modifying, and deleting catalog records. The supported sub-operations for the Transaction operation are Insert, Update, and Delete.

The CSW Transactions endpoint only supports **HTTP POST** requests since there are no KVP (Key-Value Pairs) operations.

2.1.4.3.3. Transaction Insert Sub-Operation HTTP POST

The Insert sub-operation is a method for one or more records to be inserted into the catalog. The schema of the record needs to conform to the schema of the information model that the catalog supports as described using the `DescribeRecord` operation.

The following example shows a request for a record to be inserted.

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>2006-05-12</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>
```

NOTE

The `typeName` attribute in the `csw:Insert` element can be used to specify the document type that's being inserted and to select the appropriate input transformer.

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>
```

2.1.4.3.4. Transaction Insert Response

The following is an example of an `application/xml` response to the Transaction Insert sub-operation:

Note that you will only receive the `InsertResult` element if you specify `verboseResponse="true"`.


```

<?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>2dbcfba3f3e24e3e8f68c50f5a98a4d1</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>

```

2.1.4.3.5. Transaction Update Sub-Operation **HTTP POST**

The Update sub-operation is a method to specify values used to change existing information in the catalog. If individual record property values are specified in the **Update** element, using the **RecordProperty** element, then those individual property values of a catalog record are replaced. The **RecordProperty** contains a **Name** and **Value** element. The **Name** element is used to specify the name of the record property to be updated. The **Value** element contains the value that will be used to update the record in the catalog. The values in the **Update** will completely replace those that are already in the record.

Some properties are given default **Value**'s if no **Value** is provided.

Table 10. **RecordProperty** Default Values

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

Other properties are removed if the **RecordProperty** contains a **Name** but not a **Value**.

The number of records affected by an Update operation is determined by the contents of the **Constraint** element, which contains a filter for limiting the update to a specific record or group of records.

The following example shows how the newly inserted record could be updated to modify the date field. If your update request contains a **<cs:Record>** rather than a set of **<RecordProperty>** elements plus a **<Constraint>**, the existing record with the same ID will be replaced with the new record.

Sample XML Transaction Update Request

```
<?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>2dbcfba3f3e24e3e8f68c50f5a98a4d1</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>2008-08-10</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>
```

The following example shows how the newly inserted record could be updated to modify the date field while using a filter constraint with title equal to *Aliquam fermentum purus quis arcu*.

Sample XML Transaction **Update** Request with 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:Name>date</csw:Name>
      <csw:Value>2015-08-25</csw:Value>
    </csw:RecordProperty>
    <csw:RecordProperty>
      <csw:Name>format</csw:Name>
      <csw:Value></csw:Value>
    </csw:RecordProperty>
    <csw:Constraint version="2.0.0">
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>title</ogc:PropertyName>
          <ogc:Literal>Aliquam fermentum purus quis arcu</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Update>
</csw:Transaction>
```

The following example shows how the newly inserted record could be updated to modify the date field while using a CQL filter constraint with title equal to **Aliquam fermentum purus quis arcu**.

```
<?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:Name>date</csw:Name>
      <csw:Value>2015-08-25</csw:Value>
    </csw:RecordProperty>
    <csw:RecordProperty>
      <csw:Name>format</csw:Name>
      <csw:Value></csw:Value>
    </csw:RecordProperty>
    <csw:Constraint version="2.0.0">
      <ogc:CqlText>
        title = 'Aliquam fermentum purus quis arcu'
      </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>
```

2.1.4.3.6. Transaction Delete Sub-Operation HTTP POST

The Delete sub-operation is a method to identify a set of records to be deleted from the catalog.

The following example shows a delete request for all records with a SpatialReferenceSystem name equal to **WGS-84**.

Sample XML Transaction Delete Request with 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>SpatialReferenceSystem</ogc:PropertyName>
          <ogc:Literal>WGS-84</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Delete>
</csw:Transaction>
```

The following example shows a delete operation specifying a CQL constraint to delete all records with a title equal to *Aliquam fermentum purus quis arcu*

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>
        title = 'Aliquam fermentum purus quis arcu'
      </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:csw="http://www.opengis.net/cat/csw/2.0.2"
    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>0</csw:totalUpdated>
    <csw:totalDeleted>1</csw:totalDeleted>
  </csw:TransactionSummary>
</csw:TransactionResponse>
```

2.1.4.3.7. Subscription **GetRecords** Operation

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.

2.1.4.3.8. Subscription **GetRecords** HTTP GET

GetRecords KVP (Key-Value Pairs) Encoding

```
https://{FQDN}:{PORT}/services/csw/subscription?service=CSW&version=2.0.2&request=GetRecords&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPACE=xmlns(csw=http://www.opengis.net/cat/csw/2.0.2)&resultType=results&typeName=csw:Record&elementSetName=brief&ResponseHandler=https%3A%2F%2Fsome.ddf%2Fservices%2Fcs%2Fsubscription%2Fevent&ConstraintLanguage=CQL_TEXT&constraint=Text Like '%25'
```

2.1.4.3.9. Subscription **GetRecords** HTTP POST

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 typeName="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>
```

2.1.4.3.10. Subscription **GetRecords** HTTP PUT

The HTTP **PUT** request **GetRecords** is used to update an existing subscription. It is the same as the **POST**, except the **requestid** URN is appended to the url.

Subscription **GetRecords** XML Request

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

Subscription GetRecords XML Response

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2008-09-28T18:49:45" 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 typeName="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 Response

The following is an example of an `application/xml` event sent to a subscribers `ResponseHandler` using an `HTTP POST` for a create, `HTTP PUT` for an update, and `HTTP DELETE` for a delete using the default `outputSchema` of `http://www.opengis.net/cat/csw/2.0.2` if you specified another supported schema format in the subscription it will be returned in that format.

Subscription GetRecords event XML 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>182fb33103414e5cbb06f8693b526239</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>
```

2.1.4.3.11. Subscription HTTP GET or HTTP DELETE Request

The following is an example **HTTP GET** Request to retrieve an active subscription

Subscription HTTP GET or HTTP DELETE

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

2.1.4.3.12. Subscription HTTP GET or HTTP DELETE Response

The following is an example **HTTP GET** Response retrieving an active subscription

```

<?xml version="1.0" ?>
<Acknowledgement timeStamp="2008-09-28T18:49:45" 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 typeName="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>

```

2.1.4.3.13. Example Responses for CSW Endpoint Error Conditions

The following are example data and expected errors responses that will be returned for each error condition.

HTTP error codes are also returned. https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_Client_errors

No Transaction Contents

This will not generate an error, but the response will tell you that nothing was processed as part of the transaction. For security purposes the `ows:ExceptionText` on invalid data is generic. The log file should be consulted for more information.

Example CSW Request with no payload

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" verboseResponse="true" version="2.0.2" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2">
</csw:Transaction>
```

No Payload CSW 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:csw=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:gml="http://www.opengis.net/gml" 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>0</csw:totalUpdated>
    <csw:totalDeleted>0</csw:totalDeleted>
  </csw:TransactionSummary>
</csw:TransactionResponse>
```

Malformed XML

The follow example sends malformed XML to the CSW Endpoint.

Example Malformed XML 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>2006-05-12</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>
```

An HTTP 400 Bad request response is returned. The error is logged in the log file and the following response body is returned.

Malformed XML CSW Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ows:ExceptionReport xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:gml="http://www.opengis.net/gml" 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="1.2.0"
ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
  <ows:Exception exceptionCode="MissingParameterValue">
    <ows:ExceptionText>Error parsing the request. XML parameters may be missing or
invalid.</ows:ExceptionText>
  </ows:Exception>
</ows:ExceptionReport>
```

Non-CSW Request

The following example sends a non-CSW request to the CSW endpoint.

Example Non-CSW request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<document>
  <title>boucle dampish caulkers</title>
  <id>abc123</id>
</document>
```

An HTTP 400 Bad request response is returned, and the following response body is returned.

Non-CSW Data Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ows:ExceptionReport xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:gml="http://www.opengis.net/gml" 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="1.2.0"
ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
  <ows:Exception exceptionCode="InvalidParameterValue" locator="service">
    <ows:ExceptionText>Unknown Service</ows:ExceptionText>
  </ows:Exception>
</ows:ExceptionReport>
```

Request with Unknown Schema

This type of request will succeed and attribute names that match the expected names for the typeName (e.g. csw:Record) will get mapped into the metacard. In the example, the `title` attribute will get mapped to the metacard `title` attribute since it's the same attribute name as `<dc:title>` that `csw:Record` is configured to parse.

Example Unknown Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" verboseResponse="true" version="2.0.2" 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"
      xmlns:unk="http://example.com/unknown">
      <unk:id>123</unk:id>
      <unk:title>Aliquam fermentum purus quis arcu</unk:title>
    </csw:Record>
  </csw:Insert>
</csw:Transaction>
```

Metacard is created successfully.

Example Successful Unknown Schema 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>1</csw:totalInserted>
    <csw:totalUpdated>0</csw:totalUpdated>
    <csw:totalDeleted>0</csw:totalDeleted>
  </csw:TransactionSummary>
  <csw:InsertResult>
    <csw:BriefRecord>
      <dc:identifier>4ec3ec03f75344a7b4404773f97e5a03</dc:identifier>
      <dc:title>Aliquam fermentum purus quis arcu</dc:title>
      <dc:type/>
    </csw:BriefRecord>
  </csw:InsertResult>
</csw:TransactionResponse>
```

Well-formed, but Invalid TypeName

The `typeName` on the `csw:Insert` specifies the transformer to use when parsing the data. If the name specified is not configured, an error response is returned.

Example Invalid typeName

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" verboseResponse="true" version="2.0.2" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2">
  <csw:Insert typeName="invalid-data">
    <root>
      <id>abcd16df29413796b388b02ee017a315</id>
    </document>
  </csw:Insert>
</csw:Transaction>
```

An HTTP 400 Bad request response is returned. The error is logged in the log file and the following response body is returned.

Invalid typeName Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ows:ExceptionReport xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:cs=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:gml="http://www.opengis.net/gml" xmlns:ns6=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dct=
"http://purl.org/dc/terms/" xmlns:ns9="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance" version="1.2.0"
ns10:schemaLocation="http://www.opengis.net/csw/ogc/csw/2.0.2/CSW-publication.xsd">
  <ows:Exception exceptionCode="MissingParameterValue">
    <ows:ExceptionText>Error parsing the request. XML parameters may be missing or
invalid.</ows:ExceptionText>
  </ows:Exception>
</ows:ExceptionReport>
```

Request with Missing XML Prologue

The following example sends XML data to the CSW Endpoint without the XML prologue.

Example Missing XML Tag

```
<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>2006-05-12</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>
```

Metacard is created successfully.

Example Missing XML Tag 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>1</csw:totalInserted>
    <csw:totalUpdated>0</csw:totalUpdated>
    <csw:totalDeleted>0</csw:totalDeleted>
  </csw:TransactionSummary>
  <csw:InsertResult>
    <csw:BriefRecord>
      <dc:identifier>c318d32e9c9a4bb5b1cd00bc1aafd704</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>44.792 -6.171</ows:LowerCorner>
        <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
      </ows:BoundingBox>
    </csw:BriefRecord>
  </csw:InsertResult>
</csw:TransactionResponse>
```

Request with Non-XML Data

The following is a non-XML request sent to the CSW Endpoint.

Non-XML data Example

```
title: Non-XML title
id: abc123
```

An HTTP 400 Bad request response is returned. The error is logged in the log file and the following response body is returned.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ows:ExceptionReport 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="1.2.0"
ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
  <ows:Exception exceptionCode="MissingParameterValue">
    <ows:ExceptionText>Error parsing the request. XML parameters may be missing or
invalid.</ows:ExceptionText>
  </ows:Exception>
</ows:ExceptionReport>
```

2.1.5. FTP Endpoint

The FTP Endpoint provides a method for ingesting files directly into the DDF Catalog using the FTP protocol. The files sent over FTP are not first written to the file system, like the Directory Monitor, but instead the FTP stream of the file is ingested directly into the DDF catalog, thus avoiding extra I/O overhead.

2.1.5.1. Installing the FTP Endpoint

The FTP Endpoint is not installed by default with a standard installation.

To install:

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the **catalog-core-ftp** feature.

2.1.5.2. Configuring the FTP Endpoint

Once installed, the configurable properties for the FTP Endpoint are accessed from the **FTP Endpoint** Configuration:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **FTP Endpoint**.

See [FTP Endpoint configurations](#) for all possible configurations.

2.1.6.2. Configuring the KML Endpoint

This KML Network Link endpoint has the ability to serve up custom KML style documents and Icons to be used within that document. The KML style document must be a valid XML document containing a KML style. The KML Icons should be placed in a single level directory and must be an image type (png, jpg, tif, etc.). The Description will be displayed as a pop-up from the root network link on Google Earth. This may contain the general purpose of the network and URLs to external resources.

See [KML Endpoint configurations](#) for all possible configurations.

2.1.6.3. Using the KML Endpoint

Once installed, the KML Network Link endpoint can be accessed at:

```
https://{FQDN}:{PORT}/services/catalog/kml
```

After the above request is sent, a KML Network Link document is returned as a response to download or open. This KML Network Link can then be opened in Google Earth.

Example Output from KML Endpoint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:ns2="http://www.google.com/kml/ext/2.2"
  xmlns:ns3="http://www.w3.org/2005/Atom" xmlns:ns4=
  "urn:oasis:names:tc:ciq:xdschema:xAL:2.0">
  <NetworkLink>
    <name>DDF</name>
    <open xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
  "http://www.w3.org/2001/XMLSchema" xsi:type="xs:boolean">true</open>
    <Snippet maxLines="0"/>
    <Link>
      <href>http://0.0.0.0:8181/services/catalog/kml/sources</href>
      <refreshMode>onInterval</refreshMode>
      <refreshInterval>43200.0</refreshInterval>
      <viewRefreshMode>never</viewRefreshMode>
      <viewRefreshTime>0.0</viewRefreshTime>
      <viewBoundScale>0.0</viewBoundScale>
    </Link>
  </NetworkLink>
</kml>
```

The KML endpoint can also serve up Icons to be used in conjunction with the KML style document. The request below shows the format to return an icon.

NOTE `<icon-name>` must be the name of an icon contained in the directory being served.

```
https://{FQDN}:{PORT}/services/catalog/kml/icons?<icon-name>
```

2.1.7. Metrics Endpoint

NOTE

EXPERIMENTAL

WARNING

Note that the Metrics endpoint URL is marked "internal." This indicates that this endpoint is intended for internal use by the DDF code. This endpoint is subject to change in future versions.

The Metrics Endpoint is used by the [Metrics Collection Application](#) to report on system metrics.

2.1.7.1. Installing the Metrics Endpoint

The Metrics Endpoint is installed by default with a standard installation in the Platform application.

2.1.7.2. Configuring the Metrics Endpoint

No configuration can be made for the Metrics Endpoint. All of the metrics that it collects data on are pre-configured in DDF.

2.1.7.3. Using the Metrics Endpoint

Metrics Endpoint URL

```
https://{FQDN}:{PORT}/services/internal/metrics/catalogQueries.png?startDate=2013-03-31T06:00:00-07:00&endDate=2013-04-01T11:00:00-07:00
```

The table below lists all of the options for the Metrics endpoint URL to execute custom metrics data requests:

Table 11. Metrics Endpoint URL Options

Parameter	Description	Example	Required
<code>startDate</code>	Specifies the start of the time range of the search on the metric's data (RFC-3339 - Date and Time format, i.e. YYYY-MM-DDTHH:mm:ssZ). Date/time must be earlier than the endDate. <i>This parameter cannot be used with the <code>dateOffset</code> parameter.</i>	<code>startDate=2013-03-31T06:00:00-07:00</code>	true

Parameter	Description	Example	Required
endDate	Specifies the end of the time range of the search on the metric's data (RFC-3339 - Date and Time format, i.e. YYYY-MM-DDTHH:mm:ssZ). Date/time must be later than the startDate. <i>This parameter cannot be used with the dateOffset parameter.</i>	endDate=2013-04-01T11:00:00-07:00	true
dateOffset	Specifies an offset, backwards from the current time, to search on the modified time field for entries. Defined in seconds and must be a positive Integer. <i>This parameter cannot be used with the startDate or endDate parameters.</i>	dateOffset=1800	true
yAxisLabel	The label to apply to the graph's y-axis. Will default to the metric's name, e.g., Catalog Queries. <i>This parameter is only applicable for the metric's graph display format.</i>	Catalog Query Count	false
title	The title to be applied to the graph. + Will default to the metric's name plus the time range used for the graph. + <i>This parameter is only applicable for the metric's graph display format.</i>	Catalog Query Count for the last 15 minutes	false

2.1.7.3.1. Metrics Data Supported Formats

The metric's historical data can be displayed in several formats, including PNG , a CSV file, an Excel .xls file, a PowerPoint .ppt file, an XML file, and a JSON file. The PNG, CSV, and XLS formats are accessed via hyperlinks provided in the Metrics tab web page. The PPT, XML, and JSON formats are accessed by specifying the format in the custom URL, e.g., <http://{FQDN}:{PORT}/services/internal/metrics/catalogQueries.json?dateOffset=1800>.

The table below describes each of the supported formats, how to access them, and an example where applicable. (NOTE: all example URLs begin with https://{FQDN}:{PORT} which is omitted in the table for brevity.)

Table 12. Metrics Formats

Display Format	Description	How To Access	Example URL
PNG	Displays the metric's data as a PNG-formatted graph, where the x-axis is time and the y-axis is the metric's sampled data values.	Via hyperlink on the Metrics tab or directly via custom URL.	Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds):
			/services/internal/metrics/catalogQueries.png?dateOffset=28800&yAxisLabel=mylabel&title=mygraphtitle
			Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:
			/services/internal/metrics/catalogQueries.png?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00&yAxisLabel=mylabel&title=mygraphtitle <i>Note that the <code>yAxisLabel</code> and <code>title</code> parameters are optional.</i>
CSV	Displays the metric's data as a Comma-Separated Value (CSV) file, which can be auto-displayed in Excel based on browser settings. The generated CSV file will consist of two columns of data: Timestamp and Value, where the first row contains the column headers and the remaining rows contain the metric's sampled data over the specified time range.	Via hyperlink on the Metrics tab or directly via custom URL.	Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds):
			/services/internal/metrics/catalogQueries.csv?dateOffset=28800
			Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:
			/services/internal/metrics/catalogQueries.csv?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00
XLS	Displays the metric's data as an Excel (XLS) file, which can be auto-displayed in Excel based on browser settings. The generated XLS file will consist of: Title in first row based on metric's name and specified time range Column headers for Timestamp and Value; Two columns of data containing the metric's sampled data over the specified time range; The total count, if applicable, in the last row	Via hyperlink on the Metrics tab or directly via custom URL.	Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds):
			/services/internal/metrics/catalogQueries.xls?dateOffset=28800
			Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:
			/services/internal/metrics/catalogQueries.xls?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00

Display Format	Description	How To Access	Example URL
PPT	Displays the metric's data as a PowerPoint (PPT) file, which can be auto-displayed in PowerPoint based on browser settings. The generated PPT file will consist of a single slide containing: A title based on the metric's name; The metric's PNG graph embedded as a picture in the slide The total count, if applicable	Via custom URL only	Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds):
			/services/internal/metrics/catalogQueries.ppt?dateOffset=28800
			Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:
			/services/internal/metrics/catalogQueries.ppt?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00
XML	Displays the metric's data as an XML-formatted file.	via custom URL only	Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds):
			/services/internal/metrics/catalogQueries.xml?dateOffset=28800
			Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:
			/services/internal/metrics/catalogQueries.xml?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00
JSON	Displays the metric's data as an JSON-formatted file.	via custom URL only	See Sample XML-formatted output .
			Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds):
			/services/internal/metrics/catalogQueries.json?dateOffset=28800
			Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:
			/services/internal/metrics/catalogQueries.json?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00
			See Sample JSON-Formatted Output .

Sample XML-Formatted Output

```
<catalogQueries>
  <title>Catalog Queries for Apr 15 2013 08:45:53 to Apr 15 2013 09:00:53</title>
  <data>
    <sample>
      <timestamp>Apr 15 2013 08:45:00</timestamp>
      <value>361</value>
    </sample>
    <sample>
      <timestamp>Apr 15 2013 09:00:00</timestamp>
      <value>353</value>
    </sample>
    <totalCount>5721</totalCount>
  </data>
</catalogQueries>
```

Sample JSON-formatted Output

```
{
  "title": "Query Count for Jul 9 1998 09:00:00 to Jul 9 1998 09:50:00",
  "totalCount": 322,
  "data": [
    {
      "timestamp": "Jul 9 1998 09:20:00",
      "value": 54
    },
    {
      "timestamp": "Jul 9 1998 09:45:00",
      "value": 51
    }
  ]
}
```

2.1.7.3.2. Add Custom Metrics to the Metrics Tab

It is possible to add custom (or existing, but non-collected) metrics to the Metrics tab by writing an application. Refer to the SDK example source code for Sample Metrics located in the DDF source code at [sdk/sample-metrics](#) and [sdk/sdk-app](#).

WARNING

The Metrics framework is not an open API, but rather a closed, internal framework that can change at any time in future releases. Be aware that any custom code written may not work with future releases.

2.1.7.4. Usage Limitations of the Metrics Endpoint

The Metrics Collecting Application uses a “round robin” database. It uses one that does not store individual values but, instead, stores the rate of change between values at different times. Due to the nature of this method of storage, along with the fact that some processes can cross time frames, small discrepancies (differences in values of one or two have been experienced) may appear in values for different time frames. These will be especially apparent for reports covering shorter time frames such as 15 minutes or one hour. These are due to the averaging of data over time periods and should not impact the values over longer periods of time.

2.1.8. 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.

2.1.8.1. Installing the OpenSearch Endpoint

The OpenSearch Endpoint is installed by default with a standard installation in the Catalog application.

2.1.8.2. Configuring the OpenSearch Endpoint

The OpenSearch Endpoint has no configurable properties. It can only be installed or uninstalled.

2.1.8.3. OpenSearch URL

```
https://{FQDN}:{PORT}/services/catalog/query
```

2.1.8.3.1. From Code:

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.

2.1.8.3.2. From a Web Browser:

Many modern web browsers currently act as OpenSearch clients. The request call is an HTTP GET with the query options being parameters that are passed.

Example of an OpenSearch request:

```
http://{FQDN}:{PORT}/services/catalog/query?q=Predator
```

This request performs a full-text search for the phrase 'Predator' on the DDF providers and provides the results as Atom-formatted XML for the web browser to render.

2.1.8.3.3. Parameter List

Table 13. Main OpenSearch Standard

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

Table 14. Temporal Extension

OS Element	HTTP Parameter	Possible Values	Comments
<code>start</code>	<code>dtstart</code>	RFC-3399-defined value (e.g. <code>YYYY-MM-DDTHH:mm:ssZ</code> , <code>yyyy-MM-dd'T'HH:mm:ss.SSSZ</code>)	Specifies the beginning of the time slice of the search. Default value of "1970-01-01T00:00:00Z" is used when <code>dtend</code> is specified but <code>dtstart</code> is not specified.

OS Element	HTTP Parameter	Possible Values	Comments
end	dtend	RFC-3399-defined value (e.g. YYYY-MM-DDTHH:mm:ssZ, 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.

NOTE

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

2011-01-01T12:00:00.111-04:00.

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

Geospatial Extension

These geospatial query parameters are used to create a geospatial **INTERSECTS** query, where **INTERSECTS** means geometries that are not **DISJOINT** of the given geospatial parameters.

OS Element	HTTP 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

OS Element	HTTP Parameter	Possible Values	Comments
<code>polygon</code>	<code>polygon</code>	Comma-delimited list of lat/lon (<code>EPSG:4326 (WGS84)</code> 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. <code>-80, -170, 0, -170, 80, -170, 80, 170, 0, 170, -80, 170, -80, -170</code>)	According to the OpenSearch Geo Specification this is deprecated . Use the <code>geometry</code> parameter instead.
<code>box</code>	<code>bbox</code>	4 comma-delimited <code>EPSG:4326 (WGS84)</code> decimal degrees coordinates in the format West,South,East,North	
<code>geometry</code>	<code>geometry</code>	<p>WKT Geometries</p> <p>Examples:</p> <p><code>POINT(10 20)</code> where 10 is the longitude and 20 is the latitude.</p> <p><code>POLYGON ((30 10, 10 20, 20 40, 40 40, 30 10))</code>. 30 is longitude and 10 is latitude for the first point.</p> <p><code>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)</code></p> <p><code>GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))</code></p>	Make sure to repeat the starting point as the last point to close the polygon.

Table 15. Extensions

OS Element	HTTP Parameter	Possible Values	Comments
sort	sort	<p><sbfield>:<sborder> where</p> <p><sbfield> is date or relevance</p> <p><sborder> is asc or desc</p>	<p><sborder> is optional but has a value of asc or desc (default is desc). However, when <sbfield> is relevance, <sborder> must be desc.</p> <p>Sorting by date will sort the results by the effective date.</p> <p>default: relevance:desc</p>
maxResults	mr	Integer >= 0	<p>Maximum # of results to return.</p> <p>If count is also specified, the count value will take precedence over the maxResults value.</p> <p>default: 1000</p>
maxTimeout	mt	Integer > 0	<p>Maximum timeout (milliseconds) for query to respond.</p> <p>default: 300000 (5 minutes)</p>

Table 16. Federated Search

OS Element	HTTP 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.

Table 17. DDF Extensions

OS Element	HTTP Parameter	Possible Values	Comments
<code>dateOffset</code>	<code>dtoffset</code>	Integer > 0	Specifies an offset (milliseconds), backwards from the current time, to search on the <code>modified</code> time field for entries.
<code>type</code>	<code>type</code>	Any valid datatype (e.g. <code>Text</code>)	Specifies the type of data to search for.
<code>version</code>	<code>version</code>	Comma-delimited list of strings (e.g. 20,30)	Version values for which to search.
<code>selector</code>	<code>selector</code>	Comma-delimited list of XPath string selectors (e.g. <code>//namespace:example, //example`</code>)	Selectors to narrow the query.

2.1.8.3.4. Supported Complex Contextual Query Format

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

Using parentheses to change the order of operations is supported. Using quotes to group keywords into literal expressions is supported.

See the [OpenSearch](#) specification for more syntax specifics.

2.1.9. WPS Endpoint

NOTE | EXPERIMENTAL

The WPS endpoint enables a client to execute and monitor long running processes.

2.1.9.1. Installing WPS Endpoint

The WPS Endpoint is not installed by default with a standard installation.

To install:

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the **spatial-wps** feature.

2.1.9.2. Configuring WPS Endpoint

The WPS endpoint has no configurable properties. It can only be installed or uninstalled.

2.1.9.3. WPS Endpoint URL

The WPS endpoint is accessible from <https://{FQDN}:{PORT}/services/WPS>.

2.1.10. WPS Endpoint Operations

For a typical sequence of WPS requests, a client would first issue a GetCapabilities request to the server to obtain an up-to-date listing of available processes. Then, it may issue a DescribeProcess request to find out more details about the particular processes offered, including the supported data formats. To run a process with the desired input data, a client will issue an Execute request. The operations GetStatus and GetResult are used in conjunction with asynchronous execution.

For brevity the examples below use GET Key-value pair requests but POST is also supported. See the OGC WPS 2.0 Interface Standard for more details.

GetCapabilities Operation

This operation allows a client to request information about the server's capabilities and processes offered.

GetCapabilities KVP (Key-Value Pairs) Encoding

```
https://{FQDN}:{PORT}/services/wps?service=WPS&version=2.0.0&request=GetCapabilities&acceptVersions=2.0.0&sections=Contents,OperationsMetadata,ServiceIdentification,ServiceProvider
```

Capabilities (Capabilities)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:Capabilities xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3=
"http://www.w3.org/1999/xlink" xmlns:ns4="http://www.opengis.net/wps/2.0" service="WPS"
version="2.0.0">
  <ns2:ServiceIdentification>
    <ns2:Title>Web Processing Service</ns2:Title>
    <ns2:Abstract>DDF WPS Endpoint</ns2:Abstract>
    <ns2:ServiceType>WPS</ns2:ServiceType>
    <ns2:Fees>NONE</ns2:Fees>
    <ns2:AccessConstraints>NONE</ns2:AccessConstraints>
  </ns2:ServiceIdentification>
  <ns2:ServiceProvider>
    <ns2:ProviderName>DDF</ns2:ProviderName>
    <ns2:ProviderSite/>
    <ns2:ServiceContact/>
  </ns2:ServiceProvider>
```

```

<ns2:OperationsMetadata>
  <ns2:Operation name="GetCapabilities">
    <ns2:DCP>
      <ns2:HTTP>
        <ns2:Get ns3:href="https://host:8993/services/wps"/>
        <ns2:Post ns3:href="https://host:8993/services/wps"/>
      </ns2:HTTP>
    </ns2:DCP>
  </ns2:Operation>
  <ns2:Operation name="DescribeProcess">
    <ns2:DCP>
      <ns2:HTTP>
        <ns2:Get ns3:href="https://host:8993/services/wps"/>
        <ns2:Post ns3:href="https://host:8993/services/wps"/>
      </ns2:HTTP>
    </ns2:DCP>
  </ns2:Operation>
  <ns2:Operation name="Execute">
    <ns2:DCP>
      <ns2:HTTP>
        <ns2:Post ns3:href="https://host:8993/services/wps"/>
      </ns2:HTTP>
    </ns2:DCP>
  </ns2:Operation>
  <ns2:Operation name="GetStatus">
    <ns2:DCP>
      <ns2:HTTP>
        <ns2:Get ns3:href="https://host:8993/services/wps"/>
        <ns2:Post ns3:href="https://host:8993/services/wps"/>
      </ns2:HTTP>
    </ns2:DCP>
  </ns2:Operation>
  <ns2:Operation name="GetResult">
    <ns2:DCP>
      <ns2:HTTP>
        <ns2:Get ns3:href="https://host:8993/services/wps"/>
        <ns2:Post ns3:href="https://host:8993/services/wps"/>
      </ns2:HTTP>
    </ns2:DCP>
  </ns2:Operation>
  <ns2:Operation name="Dismiss">
    <ns2:DCP>
      <ns2:HTTP>
        <ns2:Get ns3:href="https://host:8993/services/wps"/>
        <ns2:Post ns3:href="https://host:8993/services/wps"/>
      </ns2:HTTP>
    </ns2:DCP>
  </ns2:Operation>

```

```

</ns2:OperationsMetadata>
<ns4:Contents>
  <ns4:ProcessSummary jobControlOptions="async-execute" outputTransmission=
"reference" processVersion="1.0">
    <ns2:Title>Test Primitives</ns2:Title>
    <ns2:Abstract>Test for modeled, primitive data types.</ns2:Abstract>
    <ns2:Identifier>testPrimitives</ns2:Identifier>
  </ns4:ProcessSummary>
</ns4:Contents>
</ns4:Capabilities>

```

DescribeProcess Operation

This operation allows a client to request detailed metadata on selected processes offered by a server.

DescribeProcess KVP (Key-Value Pairs) Encoding

```

https://{FQDN}:{PORT}/services/wps?service=WPS&version=2.0.0&request=DescribeProcess&iden
tifier=testPrimitives

```

Describe Process Request

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:ProcessOfferings xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3=
"http://www.w3.org/1999/xlink" xmlns:ns4="http://www.opengis.net/wps/2.0">
  <ns4:ProcessOffering jobControlOptions="async-execute" outputTransmission="reference"
processVersion="1.0">
    <ns4:Process>
      <ns2:Title>Test Primitives</ns2:Title>
      <ns2:Abstract>Test for modeled, primitive data types.</ns2:Abstract>
      <ns2:Identifier>testPrimitives</ns2:Identifier>
      <ns4:Input minOccurs="1" maxOccurs="1">
        <ns2:Title>intParam</ns2:Title>
        <ns2:Abstract>An integer value [-2^31, 2^31-1]</ns2:Abstract>
        <ns2:Identifier>intParam</ns2:Identifier>
        <ns4:LiteralData>
          <ns4:Format encoding="UTF-8" default="true"/>
          <LiteralDataDomain default="true">
            <ns2:AnyValue/>
            <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#integer">Integer</ns2:DataType>
            <ns2:DefaultValue>3</ns2:DefaultValue>
          </LiteralDataDomain>
        </ns4:LiteralData>
      </ns4:Input>
      <ns4:Input minOccurs="1" maxOccurs="1">
        <ns2:Title>doubleParam</ns2:Title>

```

```

<ns2:Abstract>A double-precision floating point value</ns2:Abstract>
<ns2:Identifier>doubleParam</ns2:Identifier>
<ns4:LiteralData>
  <ns4:Format encoding="UTF-8" default="true"/>
  <LiteralDataDomain default="true">
    <ns2:AllowedValues>
      <ns2:Range ns2:rangeClosure="open">
        <ns2:MinimumValue>15.0</ns2:MinimumValue>
        <ns2:MaximumValue>50.0</ns2:MaximumValue>
      </ns2:Range>
    </ns2:AllowedValues>
    <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#double">Double</ns2:DataType>
    <ns2:DefaultValue>50.0</ns2:DefaultValue>
  </LiteralDataDomain>
</ns4:LiteralData>
</ns4:Input>
<ns4:Input minOccurs="1" maxOccurs="1">
  <ns2:Title>byteParam</ns2:Title>
  <ns2:Abstract>A byte value [-128, 127]</ns2:Abstract>
  <ns2:Identifier>byteParam</ns2:Identifier>
  <ns4:LiteralData>
    <ns4:Format encoding="UTF-8" default="true"/>
    <LiteralDataDomain default="true">
      <ns2:AnyValue/>
      <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#byte">Byte</ns2:DataType>
      <ns2:DefaultValue>1</ns2:DefaultValue>
    </LiteralDataDomain>
  </ns4:LiteralData>
</ns4:Input>
<ns4:Input minOccurs="1" maxOccurs="1">
  <ns2:Title>shortParam</ns2:Title>
  <ns2:Abstract>A short value [-32768, 32767]</ns2:Abstract>
  <ns2:Identifier>shortParam</ns2:Identifier>
  <ns4:LiteralData>
    <ns4:Format encoding="UTF-8" default="true"/>
    <LiteralDataDomain default="true">
      <ns2:AnyValue/>
      <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#short">Short</ns2:DataType>
      <ns2:DefaultValue>2</ns2:DefaultValue>
    </LiteralDataDomain>
  </ns4:LiteralData>
</ns4:Input>
<ns4:Input minOccurs="1" maxOccurs="1">
  <ns2:Title>longParam</ns2:Title>
  <ns2:Abstract>A long value [-2^63, 2^63-1]</ns2:Abstract>

```

```

<ns2:Identifier>longParam</ns2:Identifier>
<ns4:LiteralData>
  <ns4:Format encoding="UTF-8" default="true"/>
  <LiteralDataDomain default="true">
    <ns2:AnyValue/>
    <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#long">Long</ns2:DataType>
    <ns2:DefaultValue>4</ns2:DefaultValue>
  </LiteralDataDomain>
</ns4:LiteralData>
</ns4:Input>
<ns4:Input minOccurs="1" maxOccurs="1">
  <ns2:Title>booleanParam</ns2:Title>
  <ns2:Abstract>A boolean value [false, true]</ns2:Abstract>
  <ns2:Identifier>booleanParam</ns2:Identifier>
  <ns4:LiteralData>
    <ns4:Format encoding="UTF-8" default="true"/>
    <LiteralDataDomain default="true">
      <ns2:AnyValue/>
      <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#boolean">Boolean</ns2:DataType>
      <ns2:DefaultValue>false</ns2:DefaultValue>
    </LiteralDataDomain>
  </ns4:LiteralData>
</ns4:Input>
<ns4:Input minOccurs="1" maxOccurs="1">
  <ns2:Title>floatParam</ns2:Title>
  <ns2:Abstract>A long value [-2^63, 2^63-1]</ns2:Abstract>
  <ns2:Identifier>floatParam</ns2:Identifier>
  <ns4:LiteralData>
    <ns4:Format encoding="UTF-8" default="true"/>
    <LiteralDataDomain default="true">
      <ns2:AnyValue/>
      <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-
2/#float">Float</ns2:DataType>
      <ns2:DefaultValue>5.0</ns2:DefaultValue>
    </LiteralDataDomain>
  </ns4:LiteralData>
</ns4:Input>
<ns4:Input minOccurs="1" maxOccurs="1">
  <ns2:Title>Product Id</ns2:Title>
  <ns2:Abstract>Product Identifier</ns2:Abstract>
  <ns2:Identifier>productId</ns2:Identifier>
  <ns4:LiteralData>
    <ns4:Format encoding="UTF-8" default="true"/>
    <LiteralDataDomain default="true">
      <ns2:AnyValue/>
      <ns2:DataType ns2:reference="http://www.w3.org/TR/xmlschema-

```

```

2/#string">String</ns2:DataType>
    </LiteralDataDomain>
  </ns4:LiteralData>
</ns4:Input>
<ns4:Output>
  <ns2:Title>Product</ns2:Title>
  <ns2:Abstract>Raw output</ns2:Abstract>
  <ns2:Identifier>product</ns2:Identifier>
  <ns4:ComplexData>
    <ns4:Format encoding="raw" default="true"/>
  </ns4:ComplexData>
</ns4:Output>
</ns4:Process>
</ns4:ProcessOffering>
</ns4:ProcessOfferings>

```

GetStatus Operation

This operation allows a client to query status information of a processing job.

GetStatus KVP (Key-Value Pairs) Encoding

```

https://{FQDN}:{PORT}/services/wps?service=WPS&version=2.0.0&request=GetStatus&jobId=FB6D
D4B0-A2BB-11E3-A5E2-0800200C9A66

```

Status Info

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:StatusInfo xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3=
"http://www.w3.org/1999/xlink" xmlns:ns4="http://www.opengis.net/wps/2.0">
  <ns4:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</ns4:JobID>
  <ns4:Status>Running</ns4:Status>
  <ns4:PercentCompleted>50</ns4:PercentCompleted>
</ns4:StatusInfo>

```

GetResult Operation

This operation allows a client to query the results of a processing job. The response can be in several formats depending on the request: * If the response attribute in the request is **document** the response will be in the Result format if the response attribute is **raw** then response will be in the format defined in the output definition. * If the job failed an ExceptionReport will be returned. * If the response format is 'raw' and no data is returned than an empty response with an HTTP status of 204 will be returned.

GetResult KVP (Key-Value Pairs) Encoding

```
https://{FQDN}:{PORT}/services/wps?service=WPS&version=2.0.0&request=GetResult&jobId=FB6D4B0-A2BB-11E3-A5E2-0800200C9A66
```

Result

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:Result xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3=
"http://www.w3.org/1999/xlink" xmlns:ns4="http://www.opengis.net/wps/2.0">
  <ns4:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
  <ns4:ExpirationDate>2014-12-24T24:00:00Z</wps:ExpirationDate>
  <ns4:Output id="BUFFERED_GEOMETRY">
    <ns4:Reference xlink:href="http://result.data.server/FB6DD4B0-A2BB-11E3-A5E2-
0800200C9A66/BUFFERED_GEOMETRY.xml"/>
  </ns4:Output>
</ns4:Result>
```

Execute Operation

This operation allows a client to execute a process comprised of a process identifier, the desired data inputs, and the desired output formats. The response can be in several formats depending on the request: * If the mode is **async** the response will be in the StatusInfo format. * If the mode is **sync** and the response attribute in the request is **document** the response will be in the Result format if the response attribute is **raw** then response will be in the format defined in the output definition`. * If the mode is 'auto' then the response can be either of the aforementioned response formats. * If the job failed an ExceptionReport will be returned. * If the response format is 'raw' and no data is returned than an empty response with an HTTP status of 204 will be returned.

PostAsyncExecutionRequest HTTP POST

```
https://{FQDN}:{PORT}/services/wps?service=WPS&version=2.0.0&request=Execute
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:Execute
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd"

  service="WPS"
  version="2.0.0"
  response="document"
  mode="async">

  <ows:Identifier>reprocess</ows:Identifier>
  <wps:Input id="imagery_id">
    <wps:Input id="mission_id">
      <wps:Data>A123</wps:Data>
    </wps:Input>
    <wps:Input id="scene_id">
      <wps:Data>10</wps:Data>
    </wps:Input>
  </wps:Input>
  <wps:Output id="product" transmission="reference"/>

</wps:Execute>
```

Execution Request Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:StatusInfo xmlns:ns2="http://www.opengis.net/ows/2.0" xmlns:ns3=
"http://www.w3.org/1999/xlink" xmlns:ns4="http://www.opengis.net/wps/2.0">
  <ns4:JobID>615f5ed6-adac-4630-8b3e-4ec97b154cf6</ns4:JobID>
  <ns4:Status>Accepted</ns4:Status>
  <ns4:PercentCompleted>0</ns4:PercentCompleted>
</ns4:StatusInfo>
```

2.2. Endpoint Utility Services

DDF also provides a variety of services to enhance the function of the endpoints.

2.2.1. Compression Services

DDF supports compression of outgoing and incoming messages through the Compression Services.

These compression services are based on [CXF](#) message encoding.

The formats supported in DDF are:

gzip

Adds GZip compression to messages through CXF components. Code comes with CXF.

exi

Adds [Efficient XML Interchange \(EXI\)](#) [↗](#) support to outgoing responses. EXI is an W3C standard for XML encoding that shrinks xml to a smaller size than normal GZip compression.

2.2.1.1. Installing a Compression Service

The compression services are not installed by default with a standard installation.

To Install:

- Navigate to the **Admin Console**.
- Select the **System** tab.
- Select the **Features** tab.
- Start the service for the desired compression format:
 - `compression-exi`
 - `compression-gzip`

WARNING

The compression services either need to be installed BEFORE the desired CXF service is started or the CXF service needs to be refreshed / restarted after the compression service is installed.

2.2.1.2. Configuring Compression Services

None.

Compression Imported Services

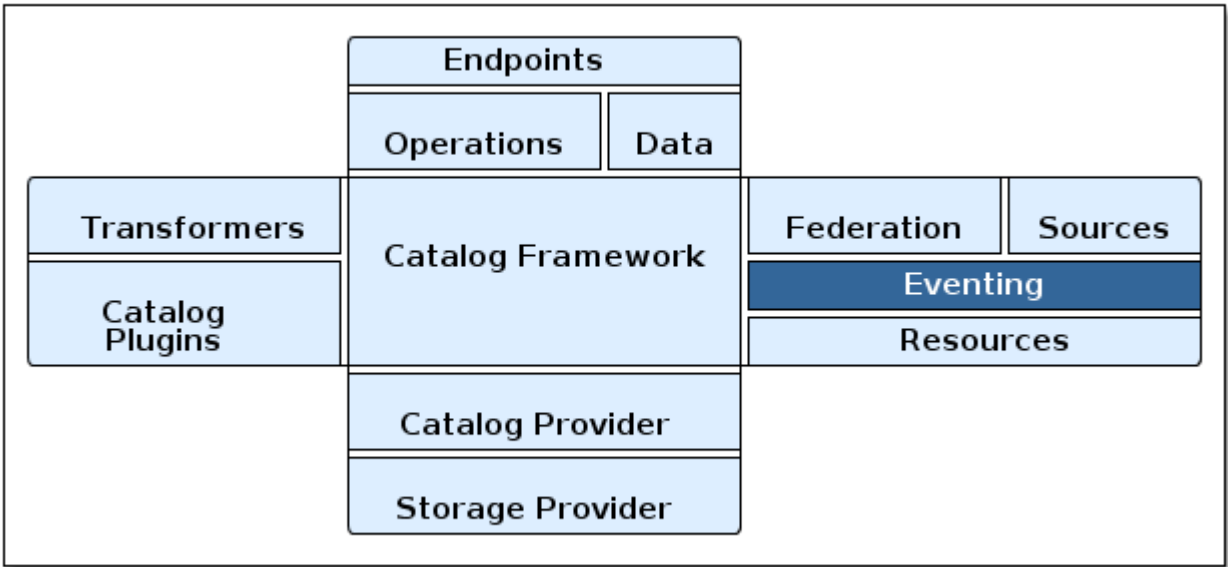
None.

Table 18. Compression Exported Services

Registered Interface	Implemented Class(es)	Service Property	Value
<code>org.apache.cxf.feature.Feature</code>	<code>ddf.compression.exi.EXIFeature</code> <code>org.apache.cxf.transport.common.gzip.GZIPFeature</code>	N/A	N/A

3. Eventing

3.1. Eventing Intro



Eventing Architecture

The Eventing capability of the Catalog allows endpoints (and thus external users) to create a "standing query" and be notified when a matching metacard is created, updated, or deleted.

Notably, the Catalog allows event evaluation on both the previous value (if available) and new value of a Metacard when an update occurs.

Eventing allows DDFs to receive events on operations (e.g. create, update, delete) based on particular queries or actions. Once subscribed, users will receive notifications of events such as update or create on any source.

3.2. Eventing Components

The key components of DDF Eventing include:

- Subscription
- Delivery Method
- [Event Processor](#)

After reading this section, you will be able to:

- Create new subscriptions

- Register subscriptions
- Perform operations on event notification
- Remove a subscription

3.3. Subscriptions

3.3.1. Subscriptions

Subscriptions represent "standing queries" in the Catalog. Like a query, subscriptions are based on the OGC Filter specification.

3.3.1.1. Subscription Lifecycle

A Subscription itself is a series of events during which various plugins or transformers can be called to process the subscription.

3.3.1.1.1. Creation

- Subscriptions are created directly with the [Event Processor](#) or declaratively through use of the Whiteboard Design Pattern.
- The Event Processor will invoke each Pre-Subscription Plugin and, if the subscription is not rejected, the subscription will be activated.

3.3.1.1.2. Evaluation

- When a metacard matching the subscription is created, updated, or deleted in any Source, each Pre-Delivery Plugin will be invoked.
- If the delivery is not rejected, the associated Delivery Method callback will be invoked.

3.3.1.1.3. Update Evaluation

Notably, the Catalog allows event evaluation on both the previous value (if available) and new value of a Metacard when an update occurs.

3.3.1.1.4. Durability

Subscription durability is not provided by the Event Processor. Thus, all subscriptions are transient and will not be recreated in the event of a system restart. It is the responsibility of Endpoints using subscriptions to persist and re-establish the subscription on startup. This decision was made for the sake of simplicity, flexibility, and the inability of the Event Processor to recreate a fully-configured Delivery Method without being overly restrictive.

IMPORTANT

Subscriptions are not persisted by the Catalog itself.

Subscriptions must be explicitly persisted by an endpoint and are not persisted by the Catalog. The Catalog Framework, or more specifically the Event Processor itself, does not persist subscriptions. Certain endpoints, however, can persist the subscriptions on their own and recreate them on system startup.

3.3.2. Creating a Subscription

Currently, the Catalog reference implementation does not contain a subscription endpoint. Therefore, an endpoint that exposes a web service interface to create, update, and delete subscriptions would provide a client's subscription filtering criteria to be used by Catalog's Event Processor to determine which events are of interest to the client. The endpoint client also provides the callback URL of the event consumer to be called when an event matching the subscription's criteria is found. This callback to the event consumer is made by a Delivery Method implementation that the client provides when the subscription is created. Whenever an event occurs in the Catalog matching the subscription, the Delivery Method implementation will be called by the Event Processor. The Delivery Method will, in turn, send the event notification out to the event consumer. As part of the subscription creation process, the Catalog verifies that the event consumer at the specified callback URL is available to receive callbacks. Therefore, the client must ensure the event consumer is running prior to creating the subscription. The Catalog completes the subscription creation by executing any pre-subscription Catalog Plugins, and then registering the subscription with the OSGi Service Registry. The Catalog does not persist subscriptions by default.

3.3.2.1. Event Processing and Notification

If an event matches a subscription's criteria, any pre-delivery plugins that are installed are invoked, the subscription's `DeliveryMethod` is retrieved, and its operation corresponding to the type of ingest event is invoked. For example, the `DeliveryMethod created()` function is called when a metacard is created. The `DeliveryMethod` operations subsequently invoke the corresponding operation in the client's event consumer service, which is specified by the callback URL provided when the `DeliveryMethod` was created. An internal subscription tracker monitors the OSGi registry, looking for subscriptions to be added (or deleted). When it detects a subscription being added, it informs the Event Processor, which sets up the subscription's filtering and is responsible for posting event notifications to the subscriber when events satisfying their criteria are met.

The Standard Event Processor is an implementation of the Event Processor and provides the ability to create/delete subscriptions. Events are generated by the CatalogFramework as metacards are created/updated/deleted and the Standard Event Processor is called since it is also a Post-Ingest Plugin. The Standard Event Processor checks each event against each subscription's criteria.

When an event matches a subscription's criteria the Standard Event Processor:

- invokes each pre-delivery plugin on the metacard in the event.
- invokes the `DeliveryMethod` operation corresponding to the type of event being processed, e.g., `created()` operation for the creation of a metacard.

- [Standard Event Processor](#)

3.3.2.1.1. Using DDF Implementation

If applicable, the implementation of `Subscription` that comes with DDF should be used. It is available at `ddf.catalog.event.impl.SubscriptionImpl` and offers a constructor that takes in all of the necessary objects. Specifically, all that is needed is a `Filter`, `DeliveryMethod`, `Set<String>` of source IDs, and a `boolean` for enterprise.

The following is an example code stub showing how to create a new instance of `Subscription` using the DDF implementation.

Creating a Subscription

```
// Create a new filter using an imported FilterBuilder
Filter filter = filterBuilder.attribute(Metacard.ANY_TEXT).like().text("*");

// Create a implementation of DeliveryMethod
DeliveryMethod deliveryMethod = new MyCustomDeliveryMethod();

// Create a set of source ids
// This set is empty as the subscription is not specific to any sources
Set<String> sourceIds = new HashSet<String>();

// Set the isEnterprise boolean value
// This subscription example should notifications from all sources (not just local)
boolean isEnterprise = true;

Subscription subscription = new SubscriptionImpl(filter, deliveryMethod, sourceIds
, isEnterprise);
```

3.3.2.2. Delivery Method

A Delivery Method provides the operation (created, updated, deleted) for how an event's metacard can be delivered.

A Delivery Method is associated with a subscription and contains the callback URL of the event consumer to be notified of events. The Delivery Method encapsulates the operations to be invoked by the Event Processor when an event matches the criteria for the subscription. The Delivery Method's operations are responsible for invoking the corresponding operations on the event consumer associated with the callback URL.

4. Security Services

4.1. Encryption Service

DDF includes an encryption service to encrypt plain text such as passwords.

4.1.1. Encryption Command

An encrypt security command is provided with DDF to encrypt text. This is useful when displaying password fields to users.

Below is an example of the `security:encrypt` command used to encrypt the plain text `myPasswordToEncrypt`. The output is the encrypted value.

security:encrypt Command Example

```
ddf@local>security:encrypt myPasswordToEncrypt
```

security:encrypt Command Output

```
ddf@local>bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

4.2. Expansion Service

The expansion service defines rulesets to map metacard attributes and user attributes to more complete sets of values. For example if a user has an attribute "alphabet" that contained the value "full", the expansion service can be configured to expand the "full" value out to ["a","b","c",...].

4.2.1. Configuring Expansion Service

To use the expansion service, modify the following two files within the `<DDF_HOME>/etc/pdp` directory:

- `<DDF_HOME>/etc/pdp/ddf-metacard-attribute-ruleset.cfg`
- `<DDF_HOME>/etc/pdp/ddf-user-attribute-ruleset.cfg`

Within these files, the following configuration details will be defined.

4.2.1.1. Expansion Service Instances and Configuration

It is expected that multiple instances of the expansion service will be running at the same time. Each instance of the service defines a unique property that is useful for retrieving specific instances of the expansion service. The following table lists the two pre-defined instances used by DDF for expanding user attributes and metacard attributes respectively.

Property Name	Value	Description
mapping	<code>security.user.attribute.mapping</code>	This instance is configured with rules that expand the user's attribute values for security checking.
mapping	<code>security.metacard.attribute.mapping</code>	This instance is configured with rules that expand the metacard's security attributes before comparing with the user's attributes.

Each instance of the expansion service can be configured using a configuration file. The configuration file can have three different types of lines:

- comments: any line prefixed with the `#` character is ignored as a comment (for readability, blank lines are also ignored)
- attribute separator: a line starting with `separator=` defines the attribute separator string.
- rule: all other lines are assumed to be rules defined in a string format `<key>:<original value>:<new value>`

The following configuration file defines the rules shown above in the example table (using the space as a separator):

```
# This defines the separator that will be used when the expansion string contains
multiple
# values - each will be separated by this string. The expanded string will be split at
the
# separator string and each resulting attribute added to the attribute set (duplicates
are
# suppressed). No value indicates the default value of ' ' (space).
separator=

# The following rules define the attribute expansion to be performed. The rules are of
the
# form:
#      <attribute name>:<original value>:<expanded value>
# The rules are ordered, so replacements from the first rules may be found in the
original
# values of subsequent rules.
Location:Goodyear:Goodyear AZ
Location:AZ:AZ USA
Location:CA:CA USA
Title:VP-Sales:VP-Sales VP Sales
Title:VP-Engineering:VP-Engineering VP Engineering
```

Table 19. Expansion Commands

Title	Namespace	Description
DDF::Security::Expansion::Commands	security	The expansion commands provide detailed information about the expansion rules in place and the ability to see the results of expanding specific values against the active ruleset.

Command	Description
<code>security:expand</code>	Runs the expansion service on the provided data returning the expanded value.
<code>security:expansions</code>	Dumps the ruleset for each active expansion service.

4.2.1.2. Expansion Command Examples and Explanation

4.2.1.2.1. `security:expansions`

The `security:expansions` command dumps the ruleset for each active expansion service. It takes no arguments and displays each rule on a separate line in the form: `<attribute name> : <original string> : <expanded string>`. The following example shows the results of executing the `expansions` command with no active expansion service.

```
ddf@local>security:expansions
No expansion services currently available.
```

After installing the `expansions` service and configuring it with an appropriate set of rules, the `expansions` command will provide output similar to the following:

```
ddf@local>security:expansions
Location : Goodyear : Goodyear AZ
Location : AZ : AZ USA
Location : CA : CA USA
Title : VP-Sales : VP-Sales VP Sales
Title : VP-Engineering : VP-Engineering VP Engineering
```

4.2.1.2.2. `security:expand`

The `security:expand` command runs the expansion service on the provided data. It takes an attribute and an original value, expands the original value using the current expansion service and ruleset and dumps the results. For the ruleset shown above, the `security:expand` command produces the following results:

```
ddf@local>security:expand Location Goodyear
[Goodyear, USA, AZ]

ddf@local>security:expand Title VP-Engineering
[VP-Engineering, Engineering, VP]

ddf@local>expand Title "VP-Engineering Manager"
[VP-Engineering, Engineering, VP, Manager]
```

4.3. Security IdP



The Security IdP application provides service provider handling that satisfies the [SAML 2.0 Web SSO profile](#)  in order to support external IdPs (Identity Providers) or SPs (Service Providers). This capability allows use of DDF as the SSO solution for an entire enterprise.

Table 20. Security IdP Components

Bundle Name	Located in Feature	Description
<code>security-idp-client</code>	<code>security-idp</code>	The IdP client that interacts with the specified Identity Provider.
<code>security-idp-server</code>	<code>security-idp</code>	An internal Identity Provider solution.

NOTE

Limitations

The internal Identity Provider solution should be used in favor of any external solutions until the IdP Service Provider fully satisfies the [SAML 2.0 Web SSO profile](#) .

4.4. Security STS

The Security STS application contains the bundles and services necessary to run and talk to a Security Token Service (STS). It builds off of the Apache CXF STS code and adds components specific to DDF functionality.

Table 21. Security STS Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
<code>security-sts-realm</code>	<code>security-sts-realm</code>	Security STS Realm
<code>security-sts-ldaplogin</code>	<code>security-sts-ldaplogin</code>	Security STS LDAP Login
<code>security-sts-ldapclaimshandler</code>	<code>security-sts-ldapclaimshandler</code>	Security STS LDAP Claims Handler
<code>security-sts-server</code>	<code>security-sts-server</code>	Security STS Server

Bundle Name	Located in Feature	Description/Link to Bundle Page
security-sts-samlvalidator	security-sts-server	Contains the default CXF SAML validator and exposes it as a service for the STS.
security-sts-x509validator	security-sts-server	Contains the default CXF x509 validator and exposes it as a service for the STS.

4.4.1. Security STS Client Config

The Security STS Client Config bundle keeps track and exposes configurations and settings for the CXF STS client. This client can be used by other services to create their own STS client. Once a service is registered as a watcher of the configuration, it will be updated whenever the settings change for the sts client.

4.4.1.1. Installing the Security STS Client Config

This bundle is installed by default.

4.4.1.2. Configuring the Security STS Client Config

Configure the Security STS Client Config from the Admin Console:

1. Navigate to the Admin Console.
2. Select **Security** Application.
3. Select **Configuration** tab.
4. Select **Security STS Client**.

See [Security STS Client configurations](#) for all possible configurations.

4.4.2. External/WS-S STS Support

4.4.2.1. Security STS WSS

This configuration works just like the STS Client Config for the internal STS, but produces standard requests instead of the custom DDF ones. It supports two new auth types for the context policy manager, WSSBASIC and WSSPKI. Use these auth types when connecting to a non-DDF STS or if ignoring realms.

4.4.2.2. Security STS Address Provider

This allows one to select which STS address will be used (e.g. in SOAP sources) for clients of this service. Default is off (internal).

4.4.3. Security STS LDAP Login

The Security STS LDAP Login bundle enables functionality within the STS that allows it to use an LDAP to perform authentication when passed a `UsernameToken` in a `RequestSecurityToken` SOAP request.

4.4.3.1. Installing the Security STS LDAP Login

This bundle is not installed by default but can be added by installing the `security-sts-ldaplogin` feature.

4.4.3.2. Configuring the Security STS LDAP Login

Configure the Security STS LDAP Login from the Admin Console:

1. Navigate to the Admin Console.
2. Select **Security** Application.
3. Select **Configuration** tab
4. Select **Security STS LDAP Login**.

Table 22. Security STS LDAP Login Settings

Configuration Name	Default Value	Additional Information
LDAP URL	<code>ldaps://{org.codice.ddf.system.hostname}:1636</code>	
StartTLS	<code>false</code>	Ignored if the URL uses ldaps.
LDAP Bind User DN	<code>cn=admin</code>	This user should have the ability to verify passwords and read attributes for any user.
LDAP Bind User Password	<code>secret</code>	This password value is encrypted by default using the Security Encryption application.
LDAP Group User Membership Attribute	<code>uid</code>	Attribute used as the membership attribute for the user in the group. Usually this is uid, cn, or something similar.
LDAP User Login Attribute	<code>uid</code>	Attribute used as the login username. Usually this is uid, cn, or something similar.
LDAP Base User DN	<code>ou=users,dc=example,dc=com</code>	
LDAP Base Group DN	<code>ou=groups,dc=example,dc=com</code>	

4.4.4. Security STS LDAP Claims Handler

The Security STS LDAP Claims Handler bundle adds functionality to the STS server that allows it to

retrieve claims from an LDAP server. It also adds mappings for the LDAP attributes to the STS SAML claims.

NOTE

All claims handlers are queried for user attributes regardless of realm. This means that two different users with the same username in different LDAP servers will end up with both of their claims in each of their individual assertions.

4.4.4.1. Installing Security STS LDAP Claims Handler

This bundle is not installed by default and can be added by installing the `security-sts-ldapclaimshandler` feature.

4.4.4.2. Configuring the Security STS LDAP Claims Handler

Configure the Security STS LDAP Claims Handler from the Admin Console:

1. Navigate to the Admin Console.
2. Select **Security Application**
3. Select **Configuration** tab.
4. Select **Security STS LDAP and Roles Claims Handler**.

Table 23. Security STS LDAP Claims Handler Settings

Configuration Name	Default Value	Additional Information
LDAP URL	<code>ldaps://{org.codice.ddf.system.hostname}:1636</code>	
StartTLS	<code>false</code>	Ignored if the URL uses ldaps.
LDAP Bind User DN	<code>cn=admin</code>	This user should have the ability to verify passwords and read attributes for any user.
LDAP Bind User Password	<code>secret</code>	This password value is encrypted by default using the Security Encryption application.
LDAP Username Attribute	<code>uid</code>	
LDAP Base User DN	<code>ou=users,dc=example,dc=com</code>	
LDAP Group ObjectClass	<code>groupOfNames</code>	<code>ObjectClass</code> that defines structure for group membership in LDAP. Usually this is <code>groupOfNames</code> or <code>groupOfUniqueNames</code>
LDAP Membership Attribute	<code>member</code>	Attribute used to designate the user's name as a member of the group in LDAP. Usually this is <code>member</code> or <code>uniqueMember</code>

Configuration Name	Default Value	Additional Information
LDAP Base Group DN	<code>ou=groups,dc=example,dc=com</code>	
User Attribute Map File	<code>etc/ws-security/attributeMap.properties</code>	Properties file that contains mappings from Claim=LDAP attribute.

Table 24. Security STS LDAP Claims Handler Imported Services

Registered Interface	Availability	Multiple
<code>ddf.security.encryption.EncryptionService</code>	optional	false

Table 25. Security STS LDAP Claims Handler Exported Services

Registered Interface	Implementation Class	Properties Set
<code>org.apache.cxf.sts.claims.ClaimHandler</code>	<code>ddf.security.sts.claimsHandler.LdapClaimsHandler</code>	Properties from the settings
<code>org.apache.cxf.sts.claims.claimHandler</code>	<code>ddf.security.sts.claimsHandler.RoleClaimsHandler</code>	Properties from the settings

4.4.5. Security STS Server

The Security STS Server is a bundle that starts up an implementation of the CXF STS. The STS obtains many of its configurations (Claims Handlers, Token Validators, etc.) from the OSGi service registry as those items are registered as services using the CXF interfaces. The various services that the STS Server imports are listed in the Implementation Details section of this page.

NOTE

The WSDL for the STS is located at the `security-sts-server/src/main/resources/META-INF/sts/wsdll/ws-trust-1.4-service.wsdl` within the source code.

4.4.5.1. Installing the Security STS Server

This bundle is installed by default and is required for DDF to operate.

4.4.5.2. Configuring the Security STS Server

Configure the Security STS Server from the Admin Console:

1. Navigate to the Admin Console.
2. Select **Security Application**
3. Select **Configuration** tab.
4. Select **Security STS Server**.

Table 26. Security STS Server Settings

Configuration Name	Default Value	Additional Information
SAML Assertion Lifetime	1800	
Token Issuer	<code>https://\${org.codice.ddf.system.hostname}:\${org.codice.ddf.system.httpsPort}\${org.codice.ddf.system.rootContext}/idp/login</code>	The name of the server issuing tokens. Generally this is unique identifier of this IdP.
Signature Username	localhost	Alias of the private key in the STS Server's keystore used to sign messages.
Encryption Username	localhost	Alias of the private key in the STS Server's keystore used to encrypt messages.

4.4.6. Security STS Service

The Security STS Service performs authentication of a user by delegating the authentication request to an STS. This is different than the services located within the Security PDP application as those ones only perform authorization and not authentication.

4.4.6.1. Installing the Security STS Realm

This bundle is installed by default and should not be uninstalled.

4.4.6.2. Configuring the Security STS Realm

The Security STS Realm has no configurable properties.

Table 27. Security STS Realm Imported Services

Registered Interface	Availability	Multiple
<code>ddf.security.encryption.EncryptionService</code>	optional	false

Table 28. Security STS Realm Exported Services

Registered Interfaces	Implementation Class	Properties Set
<code>org.apache.shiro.realm.Realm</code>	<code>ddf.security.realm.sts.StsRealm</code>	None