Integrating DDF

Table of Contents

1. License

2. Overview

This section supports integrating DDF with existing applications or frameworks.

2.1. Understanding Metadata and Metacards

Metadata is information about a resource, organized into a schema to make it possible to search against. The DDF Catalog stores this metadata and allows access to it. If desired, the ${ddf-content} application can be installed to store the resources themselves. Metacards are single instances of metadata, representing a single record, in the Metadata Catalog (MDC). Metacards follow one of several schemas to ensure reliable, accurate, and complete metadata. Essentially, Metacards function as containers of metadata.

2.2. Populating Metacards (during ingest)

Upon ingest, a metacard transformer will read the data from the ingested file and populate the fields of the metacard. Exactly how this is accomplished depends on the origin of the data, but most fields (except id) are imported directly.

2.3. Searching Metadata

DDF provides the capability to search the Metadata Catalog (MDC) for metadata. There are a number of different types of searches that can be performed on the MDC, and these searches are accessed using one of several interfaces. This section provides a very high level overview of introductory concepts of searching with DDF. These concepts are expanded upon in later sections.

2.3.1. Search Types

There are four basic types of metadata search. Additionally, any of the types can be combined to create a compound search.

Contextual Search

A contextual search is used when searching for textual information. It is similar to a Google search over the metadata contained in the MDC. Contextual searches may use wildcards, logical operators, and approximate matches.

Spatial Search

A spatial search is used for Area of Interest (AOI) searches. Polygon and point radius searches are supported. Specifically, the spatial search looks at the metacards' location attribute and coordinates are specified in **WGS 84** decimal degrees.

Temporal Search

A temporal search finds information from a specific time range. Two types of temporal searches are supported: *relative* and *absolute*. Relative searches contain an offset from the current time, while absolute searches contain a start and an end timestamp. Temporal searches can look at the effective date attribute or the modified date.

Datatype

A datatype search is used to search for metadata based on the datatype, and optional versions. Wildcards (*) can be used in both the datatype and version fields. Metadata that matches any of the datatypes (and associated versions if specified) will be returned. If a version is not specified, then all metadata records for the specified datatype(s) regardless of version will be returned.

Compound Search

These search types may be combined to create Compound searches. For example, a Contextual and Spatial search could be combined into one Compound search to search for certain text in metadata in a particular region of the world.

2.3.2. Search Interfaces

DDF Search UI Application

The DDF Search UI application provides a graphic interface to return results and locate them on an interactive globe or map.

SSH

Additionally, it is possible to use a client script to remotely access DDF via SSH and send console commands to search and ingest data.

2.4. Catalog Search Result Objects

Data is returned from searches as Catalog Search Result objects. This is a subtype of Catalog Entry that also contains additional data based on what type of sort policy was applied to the search. Because it is a subtype of Catalog Entry, a Catalog Search Result has all Catalog Entry’s fields such as metadata, effective time, and modified time. It also contains some of the following fields, depending on type of search, that are populated by DDF when the search occurs:

- Distance: Populated when a point radius spatial search occurs. Numerical value that indicates the

result's distance from the center point of the search.

- Units: Populated when a point radius spatial search occurs. Indicates the units (kilometer, mile, etc.) for the distance field.
- Relevance: Populated when a contextual search occurs. Numerical value that indicates how relevant the text in the result is to the text originally searched for.

2.4.1. Search Programmatic Flow

Searching the catalog involves three basic steps:

1. Define the search criteria (contextual, spatial, temporal, or compound – a combination of two or more types of searches).
   a. Optionally define a sort policy and assign it to the criteria.
   b. For contextual search, optionally set the `fuzzy` flag to `true` or `false` (the default value for the `Metadata Catalog fuzzy` flag is `true`, while the `portal` default value is `false`).
   c. For contextual search, optionally set the caseSensitive flag to true (the default is that caseSensitive flag is NOT set and queries are not case sensitive). Doing so enables case sensitive matching on the search criteria. For example, if caseSensitive is set to true and the phrase is "Baghdad" then only metadata containing "Baghdad" with the same matching case will be returned. Words such as "baghdad", "BAGHDAD", and "baghDad" will not be returned because they do not match the exact case of the search term.
2. Issue a search
3. Examine the results

Sort Policies

Searches can also be sorted according to various built-in policies. A sort policy is applied to the search criteria after its creation but before the search is issued. The policy specifies to the DDF the order the MDC search results should be in when they are returned to the requesting client. Only one sort policy may be defined per search.

There are three policies available.

Table 1. Sort Policies

Sort Policy	Sorts By	Default Order	Available for
Temporal	The catalog search result's effective time field	Newest to oldest	All Search Types

Sort Policy	Sorts By	Default Order	Available for
Distance	The catalog search result's distance field	Nearest to farthest	Point-Radius Spatial searches
Relevance	The catalog search result's relevance field	Most to least relevant	Contextual

If no sort policy is defined for a particular search, the temporal policy will automatically be applied.

WARNING

For Compound searches, the parent Compound search's sort policy is used. For example, if a Spatial search and Contextual search are the components of a Compound search, the Spatial search might have a distance policy and the Contextual search might have a relevance policy. The parent Compound search, though, does not use the policy of its child objects to define its sorting approach. The Compound search itself has its own temporal sort policy field that it will use to order the results of the search.

2.5. Asynchronous Search & Retrieval

Asynchronous Search & Retrieval allows a requestor to execute multiple queries at once, begin multiple product downloads while query results are being returned, cancel queries and downloads, and receive status on the state of incoming query results and product downloads.

Table 2. Important Terms for Asynchronous Search

Capability	Description	Endpoint Integration
Asynchronous Search	Search multiple sources simultaneously	Search UI
Product caching	Allows quick retrieval of already downloaded products	DDF Catalog
Canceling Product Downloads	The ability to cancel a download in progress	DDF Catalog
Activities	Activities * `download` * `retry` * `cancel` * `pause` * `remove` * `resume`	DDF Catalog, CometD endpoint
Notifications	Time-stamped messages of an action	DDF Catalog, DDF UI/CometD endpoint
Workspaces	Ability to save and manage queries and save metacards	DDF Platform, DDF UI/CometD endpoint

Capability	Description	Endpoint Integration
3D Map support	Ability to execute a geospatial search using a 3D map	N/A

Product Retrieval

The DDF is used to catalog resources. A Resource is a URI-addressable entity that is represented by a Metacard. Resources may also be known as products or data. Resources may exist either locally or on a remote data store.

- NITF image
- MPEG video
- Live video stream
- Audio recording
- Document
- SOAP Web services
- DDF JSON
- DDF REST

The Query Service Endpoint, the Catalog Framework, and the `CatalogProvider` are key components for processing a retrieve product request. The Endpoint bundle contains a Web service that exposes the interface to retrieve products, also referred to as Resources. The Endpoint calls the `CatalogFramework` to execute the operations of its specification. The `CatalogFramework` relies on the Sources to execute the actual product retrieval. Optional PreResource and PostResource Catalog Plugins may be invoked by the `CatalogFramework` to modify the product retrieval request/response prior to the Catalog Provider processing the request and providing the response. It is possible to retrieve products from specific remote Sources by specifying the site name(s) in the request.

NOTE

Product Caching

Existing DDF clients are able to leverage product caching due to the product cache being implemented in the DDF. Enabling the product cache is an administrator function.

Product Caching is bundled with the `catalog-core-standardframework` feature. It can be configured using the " Enable Product Caching" property in the **Catalog Standard Framework** configuration.

Product Caching is disabled by default.

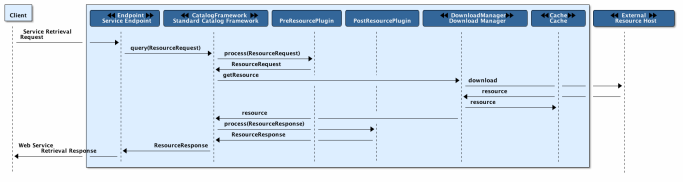


Figure 1. Product Retrieval Request

The Catalog Framework optionally provides caching of products, so future requests to retrieve the same product will be serviced much quicker. If caching is enabled, each time a retrieve product request is received, the Catalog Framework will look in its cache (default location: `<INSTALL_DIR>/data/productcache`)to see if the product has been cached locally. If it has, the product is retrieved from the local site and returned to the client, providing a much quicker turnaround because remote product retrieval and network traffic was avoided. If the requested product is not in the cache, the product is retrieved from the Source (local or remote) and cached locally while returning the product to the client. The caching to a local file of the product and the streaming of the product to the client are done simultaneously so that the client does not have to wait for the caching to complete before receiving the product. If errors are detected during the caching, caching of the product will be abandoned, and the product will be returned to the client.

The Catalog Framework attempts to detect any network problems during the product retrieval, such as long pauses where no bytes are read, implying a network connection was dropped. (The amount of time that a "long pause" is defined as is configurable, with the default value being five seconds.) The Catalog Framework will attempt to retrieve the product up to a configurable number of times (default = three), waiting for a configurable amount of time (default = 10 seconds) between each attempt, trying to successfully retrieve the product. If the Catalog Framework is unable to retrieve the product, an error message is returned to the client.

If the admin has enabled the **Always Cache When Canceled** option, caching of the product will occur even if the client cancels the product retrieval so that future requests will be serviced quickly. Otherwise, caching is canceled if the user cancels the product download.

Product Download Status

As part of the caching of products, the Catalog Framework also posts events to the OSGi notification framework. Information includes when the product download started, whether the download is retrying or failed (after the number of retrieval attempts configured for product caching has been exhausted), and when the download completes. These events are retrieved by the Search UI and presented to the user who initiated the download.

Notifications and Activities

Notifications

Currently, the notifications provide information about product retrieval only. For example, in the DDF Search UI, after a user initiates a resource download, they receive notifications when the download completed, failed, canceled, or is being retried.

Activities

Activities can be enabled by selecting "Show tasks" in the Standard Search UI configuration. Activity events include the status and progress of actions that are being performed by the user, such as searches and downloads. A list of all activities opens in a drop-down menu, from which activities can be read and deleted. If a download activity is being performed, the Activity drop-down menu provides the link to retrieve the product. If caching is enabled, a progress bar is displayed in the Activity (Product Retrieval) drop-down menu until the action being performed is complete.

2.5.4. Integrating with the Asynchronous Capabilities Endpoint

The following section shows examples of integration. The channels are used for clients to subscribe, followed by examples of request and responses. There is only one endpoint hosted at `<DDF_IP>:<DDF_PORT_NUMBER>/search/cometd`

Subscribing to Notifications

Notifications Overview

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.

Usage

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, `http://DDF_HOST:8181/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1w1qlo79j6tscii19jszwp9s2i55` notifications contain the following information:

Parameter Name	Description	Required by DDF Search UI
`application`	"Downloads" for resource retrieval. This is used as a "type" or category of messages.	Yes

Parameter Name	Description	Required by DDF Search UI
`title`	Resource/file name for resource retrieval.	Yes
`message`	Human-readable message containing status and a more detailed message.	Yes
`timestamp`	Timestamp in milliseconds of when event occurs.	Yes
`user`	CometD Session ID or unique User ID.	Yes
`status`	Status of event.	No
`option`	Resource retrieval option.	No
`bytes`	Number of bytes transmitted.	No

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 the Usage section for the data that a notification contains.

Notification Events

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 Channel

/**TODO: fix subscribe link**/ To receive all notifications, subscribe to `/ddf/notifications/*\*`

Notification Message Format

Notifications follow a specific format when they are published to the notifications channel. This message contains a data map that encapsulates the notification information.

Map Key	Description	Value Type	Possible Values	Example Values
application	Name of the application that caused the notification to be sent	String	Any (Downloads is the only application currently implemented)	"Downloads"
id	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.	String	Any	"27ec3222af1144ff827a351b1962a236"
message	User-readable message that explains the notification	String	Any	"The requested product was retrieved successfully and is available for download. "
timestamp	Time that the notification was sent	String	Positive long value (seconds since unix epoch)	"1403734355420"
title	User-readable title for the notification	String	Any String	"Product retrieval successful"
user	User who the notification relates to	String	Any String	"admin"

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"
}
```

2.5.5. Notification Operations

Notification Operations Channel

A notification operation is performed by publishing a list of commands to the CometD endpoint at `/notification/action`

Table 3. Operation Format

Map Key	Description	Value Type	Possible Values	Example Values	Comments
`action`	Type of action to request	String	Any	"remove" (Currently only used action)	If a client publishes with the `remove` action, it removes the notification with the given id from the persistence store.
`id`	ID of the notification to which the action relates	String	Any	"27ec3222af1144ff827a351b1962a236"	This is the id of the notification

Example: Notification Operation Request

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

2.5.6. Activity Events

Activity Events Channel

To receive all activity updates, follow the instructions at Subscribing to Notifications and subscribe to `/ddf/activities/*\*`

Activity Format

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.

Map Key	Description	Value Type	Possible Values	Example Values
`category`	Category of the activity	String	Any String	"Product Retrieval"

Map Key	Description	Value Type	Possible Values	Example Values
`id`	ID that uniquely identifies the activity that sent out the update. Not required to be unique per update.	String	Any String	"b72ccdf6-8ca7-4f53-a0f6-b0ad264393b0"
`message`	User-readable message that explains the current activity status	String	Any String	"Resource retrieval downloading."

`operations`	Map of operations that can be performed on this activity	JSON Map	A map of keys with populated values (that evaluate to 'true' rather than 'null' 'undefined' or 'false') These operations and their values can be used by clients to communicate back to the server by sending a message back on the same channel.
			
			If the value is a URL, the client should invoke the URL as a result of the user invoking the activity operation.
				[source,json,linenums]
				---- 	
				"operations" : {
				"download" : "http://example.com/product"
				}
				---- 	
				
				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.
				
				Note: the DDF UI will interpret several values with special icons:
				* `download`
				* `retry`
				* `cancel`
				* `pause`
				* `remove`
				* `resume`

Map Key	Description	Value Type	Possible Values	Example Values
`progress`	Percentage value of activity completion	String	Integer between 0 - 100 followed by a %	"45%"
`status`	Enumerated value that displays the current state of the activity	String	* `STARTED`, * `RUNNING`, * `FINISHED`, * `STOPPED`, * `PAUSED`, or * `FAILED`	`RUNNING`
`timestamp`	Time that the activity update was sent	String	Positive long value (seconds since unix epoch)	`1403801920875`
`title`	User-readable title for the activity update	String	Any String	"Download Complete"
`subject`	User who started the activity	String	Any String	"admin"
`bytes`	Number of bytes the activity consumed (upload or download)	Integer	Any Integer	11
`session`	The session ID of the user/subject	String	Any String	"6q2lfmmwh9dwv1c4gevboyp297"
`Custom Value`	Additional keys can be inserted by the component sending the activity notification	Any JSON Type	Any JSON	"{}"

Example: Activity update with custom 'bytes' field

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

2.5.7. Activity Operations

Channel

An activity operation is published to the channel `/service/action`

Table 4. Activity Format

Map Key	Description	Value Type	Possible Values	Example Values	Comments
`action`	Requested action	String	Any String.	Common values are * "download", * "retry", * "cancel", * "pause", * "remove", * "resume" * "cancel"	Based on the operations map that comes in from an activity event.
`id`	ID of the activity	String	Any String	"a62f6666-fc41-4a19-91f1-485e73a564b5"	The Activity ID to which the requested operation relates

Example: Activity Operation Request Message

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

2.5.8. Query Service

Query Service Channel

All query requests should be published to the `/service/query` channel.

Query Request Format

When performing a CometD publish command, the data being published must be valid json with 'data' being the key to a map that contains the following values:

Map Key	Description	Value Type	Possible Search UI Values	Example Values	Comments
`count`	Number of entries to return in the response	Number	Positive integer	250	
`format`	Format that the results should be displayed in	String	"geojson"	"geojson"	"geojson" is the recommended format to use
`id`		String		"4303ba5d-21af-4878-9a4c-808e80052e6c"	
`src`	Comma-delimited list of federated sites to search over	String	Any	list of site names.	"DDF-OS,ddf.distribution"

Map Key	Description	Value Type	Possible Search UI Values	Example Values	Comments
`start`	Specifies the number of the first result that should be returned	Number	Positive integer	1 10 would mean the 10th result from the query would be returned as the first one in the response.	cql

Query Request Examples

Enterprise Contextual

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

Multiple Site Temporal Absolute

```
"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

```
"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
}
```

2.5.9. Query Responses

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.

Query Response Message Format

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

Map Key	Description	Value Type	Possible Values	Example Values
`id`	ID that corresponds to the request	String	ID value	
`hits`	Total number of query hits that were found by the server	Number	Integer >= 0	This contains the total amount of items that were found by the query. Depending on the 'count' in the request, not all of the results may be returned.
`results`	Array of metacard results	Array of Maps	GeoJson-formatted value	This format is defined by the GeoJSON Metacard Transformer.

Map Key	Description	Value Type	Possible Values	Example Values
`results/metacard/properties/resource.cached`	A property indicating whether a metacard's associated resource is cached	Boolean	true, false	true, false
`results/metacard/actions`	An array of actions that applies to each metacard, injected into each metacard	Array of Maps	Array of objects, possibly empty if no actions are available	Each Action will contain an id, title, description, and url
`status`	Array of status for each source queried	Array		
`status.state`	Specifies the state of the query	String	SUCCEEDED, FAILED, ACTIVE	
`status.elapsed`	Time in milliseconds that it took for the source to complete the request	Number	Integer >= 0	
`status.hits`	Number of records that were found on the source that matched the query	Number	Integer >= 0	
`status.id`	ID of the federated source	String	Any string value	
`status.results`	Number of results that were returned in this response from the source	Number	Integer >= 0	
`types`	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.	Map of Maps		The Map defining a particular `metacard-type` maps the fields supported by that `metacardtype` to the datatype for that particular field.

Query Response Examples

Example Query Response


```
{
    "data": {
        "hits": 1,
        "metacard-types": {
            "ddf.metacard": {...}
        },
        "id": "857f76c0-da8c-72f5-09cd-536b69e427af",
        "results": [
           {
                "metacard": {
                    "cached": "2016-06-09T22:46:57.258+0000",
                    "geometry": {
                        "coordinates": [
                            -97.03125,
                            35.173808
                        ],
                        "type": "Point"
                    },
                    "type": "Feature",
                    "actions": [ ... ],
                    "properties": {
                        "thumbnail": "..."
                        "metadata": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><metadata>
... </metadata>",
                        "resource-size": "92314",
                        "created": "2016-06-09T22:46:37.707+0000",
                        "metacard-tags": [
                            "resource"
                        ],
                        "resource-uri": "content:1a83781dd56a4050afc420016d078e22",
                        "checksum-algorithm": "Adler32",
                        "metadata-content-type": "image/jpeg",
                        "metacard-type": "ddf.metacard",
                        "title": "fixTheBugs_geotagged.jpg",
                        "resource-download-url":
"https://localhost:8993services/catalog/sources/ddf.distribution/1a83781dd56a4050afc42001
6d078e22?transform=resource",
                        "source-id": "ddf.distribution",
                        "effective": "2016-06-09T22:46:37.707+0000",
                        "point-of-contact": "",
                        "checksum": "edfa1836",
                        "modified": "2016-06-09T22:46:37.707+0000",
                        "id": "1a83781dd56a4050afc420016d078e22"
                    }
                }
            }
```

```
        ],
        "status": [
           {
                "hits": 1,
                "elapsed": 432,
                "reasons": [],
                "id": "ddf.distribution",
                "state": "SUCCEEDED",
                "results": 1
            }
        ],
        "successful": true
    },
    "channel": "/857f76c0-da8c-72f5-09cd-536b69e427af"
}
```

3. Integrating DDF Admin

Version: 2.9.1

The DDF Admin Application is a service that allows components to perform operations on applications. This includes adding, removing, starting, stopping, and viewing status.

3.1. API

The Application service has multiple interfaces which are exposed on to the OSGi runtime for other applications to use. For more information on these interfaces, see Application Service Interfaces.

3.1.1. JMX Managed Bean

Some of the Application service API is exposed via JMX. It can either be accessed using the JMX API or from a REST-based interface created by Jolokia that comes with DDF. Here are the interfaces that are exposed in the Managed Bean:

getApplicationTree

Creates an application hierarchy tree that shows relationships between applications.

startApplication

Starts an application with the given name.

stopApplication

Stops an application with the given name.

addApplications

Adds a list of application that are specified by their URL.

3.1.2. Configuration Files

Support for configuration files was added to allow for an initial installation of applications on first run.

3.1.3. Initial Application Installation

WARNING	This application list configuration file is only read on first start.

To minimize the chance of accidentally installing and uninstalling applications, the configuration file for installing the initial applications is only read the first time that DDF is started. The only way to change what applications are active on startup is to use the console commands. Operations can also be done with the administrator web console that comes with DDF using the Features tab and installing the main feature for the desired application.

The application list file is located at `DDF_HOME/etc/org.codice.ddf.admin.applicationlist.properties`

Applications should be defined in a `<name>=<format>` syntax where location may be empty for applications that have already been added to DDF or were prepackaged with the distribution.

Examples:

```
# Local application:
opendj-embedded

# Application installed into a local maven repository:
opendj-embedded=mvn:org.codice.opendj.embedded/opendj-embedded-app/1.0.1-
SNAPSHOT/xml/features

# Application located on the file system:
opendj-embedded=file:/location/to/opendj-embedded-app-1.0.1-SNAPSHOT.kar
```

Applications will be started in the order they are listed in the file. If an application is listed, DDF will also attempt to install all dependencies for that application.

3.1.4. Interface Details

The Application Service comes with several interfaces to use.

`ApplicationService` **Interface**

The ApplicationService interface is the main class that is used to operate on applications.

`getApplications`

This method returns a set of all applications that are installed on the system. Callers can then use the Application handle to get the name and any underlying features and bundles that this application contains.

`getApplications`

Returns the application that has the given name.

`startApplication`

Starts an application, including any defined dependencies in the application.

`stopApplication`

Stops an application, does not include any external transitive dependencies as they may be needed by other applications.

`addApplication`

Adds a new application to the application list. **NOTE: This does NOT start the application.**

`removeApplication`

Removes an application that has the given URI.

`isApplicationStarted`

This method takes in an application and returns a boolean value relating whether the application is started or not. This method is generally called after retrieving a list of applications in the first method.

`getApplicationStatus`

This method, unlike `isApplicationStarted`, returns the full status of an application. This status contains detailed information about the health of the application and is described in the `ApplicationStatus` interface section.

`getApplicationTree`

Creates a hierarchy tree of application nodes that show the relationship between applications.

`findFeature`

Determine which application contains a certain feature.

`Application` Interface

`getName`

Name of the application. Should be unique among applications.

`getFeatures`

Retrieves all of the features that this application contains regardless if they are required.

`getBundles`

Retrieves all of the bundles that are defined by the features and included in this application.

`ApplicationStatus`

`getApplication`

Sends back the application that is associated with this status.

`getState`

Returns the application's state as defined by ApplicationState.

`getErrorFeatures`

Returns a set of Features that were required for this application but did not start correctly.

`getErrorBundles`

Returns a set of Bundles that were required for this application but did not start correctly.

`ApplicationNode` Interface

`getApplication`

Returns the application this node is referencing.

`getStatus`

Returns the status for the application this node is referencing.

`getParent`

Returns the parent of the application.

`getChildren`

Returns the children of this application. That is, the applications that depend on this application

3.2. Implementation Details

NOTE	A client of this service is provided as an extension to the administrative console. Information about how to use it is available on the Application Commands page.

3.2.1. Imported Services

Registered Interface	Availability	Multiple	Notes
`org.apache.karaf.features.FeaturesService`	required	false	Provided by Karaf Framework
`org.apache.karaf.bundle.core.BundleStateService`	required	true	Installed as part of Platform Status feature.

3.2.2. Exported Services

Registered Interface	Implementation Class	Notes
`org.codice.ddf.admin.application.service.ApplicationService`	`org.codice.ddf.admin.application.service.impl.ApplicationServiceImpl`	

4. Integrating DDF Catalog

Version: 2.9.1

4.1. Design

The Catalog is composed of several components and an API that connects them together. The Catalog API is central to DDF's architectural qualities of extensibility and flexibility. The Catalog API consists of Java interfaces that define Catalog functionality and specify interactions between components. These interfaces provide the ability for components to interact without a dependency on a particular underlying implementation, thus allowing the possibility of alternate implementations that can maintain interoperability and share developed components. As such, new capabilities can be developed independently, in a modular fashion, using the Catalog API interfaces and reused by other DDF installations.

4.1.1. Ensuring Compatibility

The Catalog API will evolve, but great care is taken to retain backwards compatibility with developed components. Compatibility is reflected in version numbers.

This section supports integration of the Catalog Application.

4.2. Integrating Endpoints

Endpoints act as a proxy between the client and the Catalog Framework.

Endpoints expose the Catalog Framework to clients using protocols and formats that they understand.

Endpoint interface formats/protocols can include a variety of formats, including (but not limited to):

- SOAP Web services
- RESTful services
- JMS
- JSON
- OpenSearch

The endpoint may transform a client request into a compatible Catalog format and then transform the response into a compatible client format. Endpoints may use Transformers to perform these transformations. This allows an endpoint to interact with Source(s) that have different interfaces. For example, an OpenSearch Endpoint can send a query to the Catalog Framework, which could then query a federated source that has no OpenSearch interface.

Endpoints are meant to be the only client-accessible components in the Catalog.

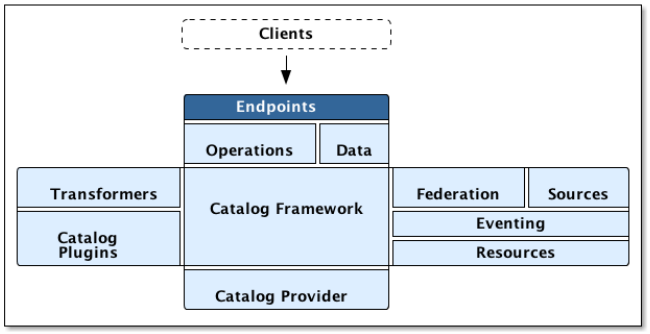


Figure 2. Endpoint Architecture

4.2.1. Existing Endpoints

The following endpoints are provided with the default Catalog out of the box:

DDF Catalog RESTful CRUD 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 http://<HOST>:<PORT>/services/catalog, where `HOST` is the IP address of where the distribution is installed and `PORT` is the port number on which the distribution is listening.

Installing and Uninstalling

The RESTful CRUD Endpoint can be installed and uninstalled using the normal processes described in the Configuring DDF section.

Configuring

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

Using the REST CRUD Endpoint

The RESTful CRUD 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
`create`	HTTP POST	HTTP request body contains the input to be ingested.	http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog

Operation	HTTP Request	Details	Example URL
`update`	HTTP PUT	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.	http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId> where `<metacardId>` is the `Metacard.ID` of the metacard to be updated
`delete`	HTTP DELETE	The ID of the Metacard to be deleted is appended to the end of the URL.	http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId> where `<metacardId>` is the `Metacard.ID` of the metacard to be deleted
`read`	HTTP GET	The ID of the Metacard to be retrieved is appended to the end of the URL. By default, the response body will include the XML representation of the Metacard.	http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId> where `<metacardId>` is the `Metacard.ID` of the metacard to be retrieved
`federated read`	HTTP GET	The SOURCE ID of a federated source is appended in the URL before the ID of the Metacard to be retrieved is appended to the end.	http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/sources/<sourceId>/<metacardId> where `<sourceid>` is the `FEDERATED SOURCE ID` and `<metacardId>` is the `Metacard.ID` of the Metacard to be retrieved
`sources`	HTTP GET	Retrieves information about federated sources, including `sourceid`, `availability`, `contentTypes`,and `version`.	http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/sources/

Sources Operation Example

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

Sources Response Example

```
[
   {
      "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.

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.)

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:

```
http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId>?transform=<T
RANSFORMER_ID>
```

TIP	Transforms also work on read operations for metacards in federated sources. http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/sources/<sourceId>/<metacardId>?transform=<TRANSFORMER_ID>

Metacard Transforms Available in DDF

DDF includes the following Metacard Transformers:

HTML Metacard Transformer

transforms a Metacard into an HTML formatted document

XML Metacard Transformer

transforms a Metacard into an XML formatted document

GeoJSON Metacard Transformer

transforms a Metacard into GeoJSON text

Thumbnail Metacard Transformer

retrieves the thumbnail bytes of a Metacard

Metadata Metacard Transformer

returns the Metacard.METADATA attribute value when given a Metacard.

Resource Metacard Transformer

retrieves the resource bytes of a Metacard product

NOTE	MetacardTransformers can be added to the system at any time. This endpoint can make use of any registered `MetacardTransformers`.

HTML Metacard Transformer

Description

The HTML Metacard Transformer is responsible for translating a Metacard into an HTML formatted document.

Usage

Using the REST Endpoint, for example, request a metacard with the transform option set to the HTML shortname.

```
https://localhost:8993/services/catalog/0123456789abcdef0123456789abcdef?transform=html
```

Installing and Uninstalling

Install the `catalog-transformer-html` feature using the Admin console.

Configuration

None

Implementation Details

Registered Interface	Service Property	Value
`ddf.catalog.transform.MetacardTransformer`	title	View as html...
	description	Transforms query results into html
	shortname (for backwards compatibility)	html

Known Issues

None

XML Metacard Transformer

Description

The XML Metacard Transformer is responsible for translating a Metacard into an XML formatted document. The metacard element that is generated is an extension of `gml:AbstractFeatureType` which makes the output of this transformer GML 3.1.1 compatible.

Usage

Using the REST Endpoint for example, request a Metacard with the transform option set to the XML shortname.

```
https://localhost:8993/services/catalog/ac0c6917d5ee45bfb3c2bf8cd2ebaa67?transform=xml
```

Installation and Uninstallation

This transformer comes installed out of the box and is running on start up. To uninstall or install manually, use the `catalog-transformer-xml` feature using the Admin Console.

Configuration

None

Implementation Details

Table 5. Metacard to XML Mappings

Metacard Variables	XML Element
`id`	`metacard/@gml:id`
`metacardType`	`metacard/type`
`sourceId`	`metacard/source`

all other attributes	`metacard/<AttributeType>[name='<AttributeName>']/value`
	
	For instance, the value for the Metacard Attribute named “title” would be found at:
	
	`metacard/string[@name='title']/value`

Table 6. AttributeTypes

XML Adapted Attributes
boolean
base64Binary
dateTime
double
float
geometry
int
long
object
short
string
stringxml

Known Issues

None

GeoJSON Metacard Transformer

Description

The GeoJSON Metacard Transformer translates a Metacard into GeoJSON.

Usage

The GeoJSON Metacard Transformer can be used programmatically by requesting a

`MetacardTransformer` with the id `geojson`. It can also be used within the REST Endpoint by providing the transform option as geojson.

Example

REST GET method with the GeoJSON MetacardTransformer

```
https://localhost:8993/services/catalog/0123456789abcdef0123456789abcdef?transform=geojson
```

Installation and Uninstallation

Install the `catalog-transformer-json` feature using the Admin Console.

Configuration

None

Implementation Details

Registered Interface	Service Property	Value
`ddf.catalog.transform.MetacardTransformer`	mime-type	application/json
	id	geojson
	shortname (for backwards compatibility)	geojson

Known Issues

None.

Thumbnail Metacard Transformer

Description

The Thumbnail Metacard Transformer retrieves the thumbnail bytes of a Metacard by returning the `Metacard.THUMBNAIL` attribute value.

Usage

Endpoints or other components can retrieve an instance of the Thumbnail Metacard Transformer using its id thumbnail.

Sample Blueprint Reference Snippet

```
 <reference id="metacardTransformer" interface="
ddf.catalog.transform.MetacardTransformer" filter="(id=thumbnail)"/>
```

The Thumbnail Metacard Transformer returns a `BinaryContent` object of the `Metacard.THUMBNAIL` bytes and a MIME Type of `image/jpeg`.

Installation and Uninstallation

This transformer is installed by default. To uninstall the transformer, you must stop or uninstall the bundle.

Configuration

None

Implementation Details

Service Property	Value
id	thumbnail
shortname	thumbnail
mime-type	image/jpeg

Known Issues

None

Metadata Metacard Transformer

Description

The Metadata Metacard Transformer returns the `Metacard.METADATA` attribute value when given a Metacard. The MIME Type returned is `text/xml`.

Usage

The Metadata Metacard Transformer can be used programmatically by requesting a `MetacardTransformer` with the id metadata. It can also be used within the REST Endpoint by providing the transform option as metadata.

Example

REST GET method with the Metadata MetacardTransformer

```
https://localhost:8993/services/catalog/0123456789abcdef0123456789abcdef?transform=metadata
```

Installation and Uninstallation

The Catalog Transformers App will install this feature when deployed. This transformer's feature, `catalog-transformer-metadata`, can be uninstalled or installed.

Configuration

None

Implementation Details

Registered Interface	Service Property	Value
`ddf.catalog.transform.MetacardTransformer`	mime-type	`text/xml`
	id	`metadata`
	shortname (for backwards compatibility)	`metadata`

Known Issues

None.

Resource Metacard Transformer

Description

The Resource Metacard Transformer retrieves the resource bytes of a Metacard by returning the product associated with the metacard.

Usage

Endpoints or other components can retrieve an instance of the Resource Metacard Transformer using its id `resource`.

Sample Blueprint Reference Snippet

```
 <reference id="metacardTransformer" interface="
ddf.catalog.transform.MetacardTransformer" filter="(id=resource)"/>
```

Installation and Uninstallation

This transformer is installed by installing the feature associated with the transformer `catalog-`

`transformer-resource`. To uninstall the transformer, you must uninstall the feature `catalog-transformer-resource`.

Configuration

None

Implementation Details

Service Property	Value
id	`resource`
shortname	`resource`
mime-type	`application/octet-stream`
title	Get Resource ...

Known Issues

None

`InputTransformers`

The REST Endpoint uses `InputTransformers` to create Metacards from the `metacard` endpoint. Using the REST Endpoint `create` or a `HTTP POST` operation the Catalog Framework will also use `InputTransformers` to create Metacards and associate those Metacards with the provided resource. The REST Endpoint and Catalog Framework will dynamically find `InputTransformers` that support the mime type stated in the HTTP header of a `HTTP POST`. `InputTransformers` register as Services with a list of Content-Type mime-types. The REST Endpoint and Catalog Framework receive a list of `InputTransformers` that match the Content-Type and one-by-one call the `InputTransformers` until a transformer is successful and creates a Metacard. For instance, if GeoJSON was in the body of the `HTTP POST`, then the HTTP header would need to include `application/json` in order to match the mime-type GeoJSON Input Transformer supports. The Catalog Framework will attempt to "guess" the mime-type of a resource, if it is not provided. This functionality is provided as a best effort and it is recommended to always include the mime-type if possible.

NOTE | InputTransformers can be added to the system at any time.

Resources (Content)

The Catalog Framework can interface with Storage providers to provide storage of resources to specific types of storage, e.g., file system, relational database, XML database. A default file system provider is provided by default. Storage plugins provide pluggable functionality that can be executed either immediately before or immediately after content has been stored or updated. Storage providers act as a proxy between the Catalog Framework and the mechanism storing the content, e.g., file system, relational database. Storage providers expose the storage mechanism to the Catalog Framework.

Storage providers provide the capability to the Catalog Framework to create, read, update, and delete content in the content repository.

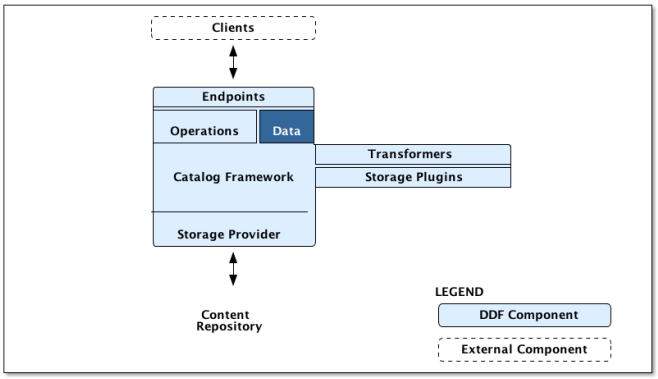


Figure 3. Content Data Component Architecture

Content Item

ContentItem is the domain object populated by the Storage Provider that represents the information about the content to be stored or content that has been stored in the Storage Provider. A ContentItem encapsulates the content's globally unique ID, mime type, and input stream (i.e., the actual content). The unique ID of a ContentItem will always correspond to a Metacard ID.

Storage Plugins

The Catalog Framework calls Storage Plugins to process each request both immediately before and immediately after an item is created or updated in the content repository.

Types of Storage Plugins available out of the box:

- Video Thumbnail Plugin, which is both a PostCreateStoragePlugin and a PostUpdateStoragePlugin and is used to generate thumbnails for video files stored in the content repository.

Video Thumbnail Plugin

The Video Thumbnail Plugin provides the ability to generate thumbnails for video files stored in the Content Repository.

It is an implementation of both the PostCreateStoragePlugin and PostUpdateStoragePlugin interfaces. When installed, it is invoked by the Catalog Framework immediately after a content item has been created or updated by the Storage Provider.

This plugin uses a custom 32-bit LGPL build of FFmpeg (a video processing program) to generate thumbnails. When this plugin is installed, it places the FFmpeg executable appropriate for the current operating system in `<DDF_INSTALL_DIR>/bin_third_party/ffmpeg`. When invoked, this plugin runs the FFmpeg binary in a separate process to generate the thumbnail. The `<DDF_INSTALL_DIR>/bin_third_party/ffmpeg` directory is deleted when the plugin is uninstalled.

NOTE Prebuilt FFmpeg binaries are provided for Linux, Mac, and Windows only.

Directory Monitor

The Catalog Content Directory Monitor allows files placed in a monitored directory to be ingested into the DDF Catalog Repository and/or the Metadata Catalog (MDC). A monitored directory is a directory configured to be polled by DDF periodically (typically once per second) for any new files added to the directory that should be ingested into the Catalog Framework.

The typical execution flow of the Directory Monitor is:

1. A new file is detected in the monitored directory,
2. The file's contents are passed on to the Catalog Framework and processed based on whether the monitored directory's processing directive was:
   a. configured to just store the file in the DDF Catalog Repository,
   b. configured to just process the file's metadata and ingest it into the MDC, or
   c. configured to both store the file in the Content Repository and ingest it into the MDC.
3. If the response from the Catalog Framework is successful, indicating the content was stored and/or processed, the file in the monitored directory is either deleted (default behavior) or copied to a sub-directory called `.ingested` (see below for how to configure this behavior). If the response from the Catalog Framework was unsuccessful or a failure occurred, the file is moved from the monitored directory to a sub-folder named `.errors`, allowing easy identification of the ingested files that had problems.

Multiple monitored directories can be configured, each monitoring different directories.

Using

The Content Directory Monitor provides the capability to easily create content in the DDF Catalog Repository and metacards in the MDC by simply placing a file in a directory that has been configured to be monitored by DDF. For example, this would be useful for copying files from a hard drive (or directory) in a batch-like operation to the monitored directory and having all of the files processed by the Catalog Framework.

Sample Usage Scenarios

Scenario 1: Monitor single directory for storage and processing, with no file backup

- The Content Directory Monitor has the following configurations.
  - The **relative** path of `inbox` for the directory path.
  - The Processing Directive is set to Store and Process.

- The **Copy Ingested Files** option is not checked.

- As files are placed in the monitored directory `<DDF_INSTALL_DIR>/inbox`, the files are ingested into the Catalog Framework.
  - The Catalog Framework generates a GUID for the create request for this ingested file.
  - Since the Store and Process directive was configured the ingested file is passed on to the Content File System Storage Provider, which creates a sub-directory in the Content Repository using the GUID and places the ingested file into this GUID sub-directory using the file name provided in the request.
  - The Catalog Framework then invokes the Catalog Content Plugin, which looks up the Input Transformer associated with the ingested file's mime type and invokes the Catalog Framework, which inserts the metacard into the MDC. This Input Transformer creates a metacard based on the contents of the ingested file.
  - The Catalog Framework sends back a successful status to the Camel route that was monitoring the directory.
  - Camel route completes and deletes the file from the monitored directory.

Scenario 2: Monitor single directory for storage with file backup

- The Content Directory Monitor has the following configurations.
  - The **absolute** path of `/usr/my/home/dir/inbox` for the directory path.
  - The Processing Directive is set to store only.
  - The **Copy Ingested Files** option is checked.

- As files are placed in the monitored directory `/usr/my/home/dir/inbox`, the files are ingested into the Catalog Framework.
  - The Catalog Framework generates a GUID for the create request for this ingested file.
  - Since the Store directive was configured, the ingested file is passed on to the Content File System Storage Provider, which creates a sub-directory in the Content Repository using the GUID and places the ingested file into this GUID sub-directory using the file name provided in the request.
  - The Catalog Framework sends back a successful status to the Camel route that was monitoring the directory.
  - The Camel route completes and moves the file from the monitored directory to its sub-directory `/usr/my/home/dir/inbox/.ingested`.

Scenario 3: Monitor multiple directories for processing only with file backup - errors encountered on some ingests

- Two different Content Directory Monitors have the following configurations.
  - The **relative** path of `inbox` and `inbox2` for the directory path.
  - The Processing Directive on both directory monitors is set to Process.
  - The Copy Ingested Files option is checked for both directory monitors.
- As files are placed in the monitored directory `<DDF_INSTALL_DIR>/inbox`, the files are ingested into the Catalog Framework.
  - The Catalog Framework generates a GUID for the create request for this ingested file.
  - Since the Process directive was configured, the ingested file is passed on to the Catalog Content Plugin, which looks up the Input Transformer associated with the ingested file's mime type (but no Input Transformer is found) and an exception is thrown.
  - The Catalog Framework sends back a failure status to the Camel route that was monitoring the directory.
  - The Camel route completes and moves the file from the monitored directory to the `.errors` sub-directory.
- As files are placed in the monitored directory `<DDF_INSTALL_DIR>/inbox2`, the files are ingested into the Catalog Framework.
  - The Catalog Framework generates a GUID for the create request for this ingested file.
  - The Catalog Framework then invokes the Catalog Content Plugin, which looks up the Input Transformer associated with the ingested file's mime type and invokes the Catalog Framework, which inserts the metacard into the MDC. This Input Transformer creates a metacard based on the contents of the ingested file.
  - The Catalog Framework sends back a successful status to the Camel route that was monitoring the directory.
  - The Camel route completes and moves the file from the monitored directory to its `.ingested` sub-directory.

Configuring

The configurable properties for the Content Directory Monitor are accessed from the **Content Directory Monitor** Configuration in the Admin Console.

Configuring Content Directory Monitors

Managed Service Factory PID: `org.codice.ddf.catalog.content.monitor.ContentDirectoryMonitor`

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Directory Path	`monitoredDirectoryPath`	String	Specifies the directory to be monitored. Can be a fully-qualified directory or a relative path (which is relative to the DDF installation directory).	N/A	Yes
Processing Directive	`directive`	String	One of three possible values from a drop down box: * Store only - indicates to only store content in Content Repository * Process only - indicates to only create metacard and insert into MDC * Store and Process - do both	Store and Process	Yes
Copy Files to Backup Directory	`copyIngestedFiles`	Boolean	Checking this option indicates that a backup of the file placed in the monitored directory should be made upon successful processing of the file. The file is moved into the `.ingested` sub-directory of the monitored directory.	False	No

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`ddf.mime.MimeTypeToTransformerMapper`	required	false
`ddf.catalog.CatalogFramework`	required	false
`ddf.catalog.filter.FilterBuilder`	required	false

Exported Services

Registered Interface	Service Property	Value
ddf.action.ActionProvider	id	catalog.data.metacard.view

Known

None.

4.2.2. OpenSearch Endpoint

The OpenSearch Endpoint provides a CDR REST Search v3.0 and CDR REST Brokered Search 1.1 compliant DDF endpoint that a client accesses 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.

This endpoint is a `JAX-RS` RESTful service and is compliant with the CDR IPT BrokeredSearch, CDR IPT OpenSearch, and OpenSearch specifications.

Installing and Uninstalling

The OpenSearch Endpoint can be installed and uninstalled using the normal processes described in the Configuring DDF section.

Configuring

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

Using the OpenSearch Endpoint

Once installed, the OpenSearch endpoint is accessible from `http://<DDF_HOST>:<DDF_PORT>/services/catalog/query`.

Using the endpoint

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.

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://<ddf_host>:8181/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.

Parameter List

Main OpenSearch Standard

OS Element	HTTP Parameter	Possible Values	Comments
`searchTerms`	`q`	URL-encoded string	Complex contextual search string.
`count`	`count`	integer >= 0	Maximum # of results to retrieve default: 10
`startIndex`	`start`	integer >= 1	Index of first result to return. default: 1 This value uses a one based index for the results.

`format`	`format`	requires a transformer shortname as a string, possible values include, when available
		
		`atom`
		
		`html`
		
		`kml`
		
		see Included Query Response Transformers for more possible values.
			default: `atom`

Temporal Extension

OS Element	HTTP Parameter	Possible Values	Comments
`start`	`dtstart`	RFC-3399-defined value	`yyyy-MM-dd T HH:mm:ss.SSSZZ`
`end`	`dtend`	RFC-3399-defined value	`yyyy-MM-dd T HH:mm:ss.SSSZZ`

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` = geometries that are not `DISJOINT` of the given geospatial parameter.

OS Element	HTTP Parameter	Possible Values	Comments
`lat`	`lat`	`EPSG:4326` decimal degrees	Expects a latitude and a radius to be specified.
`lon`	`lon`	`EPSG:4326` decimal degrees	Expects a longitude and a radius to be specified.
`radius`	`radius`	Meters along the Earth's surface > 0	Used in conjunction with lat and lon query parameters.
`polygon`	`polygon`	clockwise `lat lon` pairs ending at the first one	example: `-80, -170, 0, -170, 80, -170, 80, 170, 0, 170, -80, 170, -80, -170` According to the OpenSearch Geo Specification this is **deprecated**. Use geometry instead.
`box`	`bbox`	4 comma-separated `EPSG:4326` decimal degrees	west, south, east, north

OS Element	HTTP Parameter	Possible Values	Comments
`geometry`	`geometry`	WKT Geometries: `POINT`, `POLYGON`, `MULTIPOINT`, `MULTIPOLYGON`	Examples: `POINT(10 20)` where 10 is the longitude and 20 is the latitude. `POLYGON ( ( 30 10, 10 20, 20 40, 40 40, 30 10 ) )`. 30 is longitude and 10 is latitude for the first point. Make sure to repeat the starting point as the last point to close the polygon.

Table 7. Extensions

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

Table 8. Federated Search

OS Element	HTTP Parameter	Possible Values	Comments
routeTo	src	(varies depending on the names of the sites in the federation)	comma delimited list of site names to query. Also can specify src=local to query the local site. If src is not provided, the default behavior is to execute an enterprise search to the entire federation.

DDF Extensions

OS Element	HTTP Parameter	Possible Values	Comments
dateOffset	dtoffset	integer > 0	Specifies an offset, backwards from the current time, to search on the modified time field for entries. Defined in milliseconds.
type	type	nitf	Specifies the type of data to search for.
version	version	20,30	Comma-delimited list of version values to search for.
selector	selector	//namespace:example,//example	Comma-delimited list of XPath string selectors that narrow down the search.

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.

The following EBNF describes the grammar used for the contextual query format.

OpenSearch Complex Contextual Query `EBNF`

```
keyword query expression = optional whitespace, term, {boolean operator, term}, optional
whitespace;
boolean operator = or | not | and;
and = (optional whitespace, "AND", optional whitespace) | mandatory whitespace;
or = (optional whitespace, "OR", optional whitespace);
not = (optional whitespace, "NOT", optional whitespace);
term = group | phrase | keyword;
phrase = optional whitespace, '"', optional whitespace, keyword, { optional whitespace,
keyword}, optional whitespace, '"';
group = optional whitespace, '(', optional whitespace, keyword query expression,
optional whitespace, ')';
optional whitespace = {' '};
mandatory whitespace = ' ', optional whitespace;
valid character = ? any printable character ? - ('"' | '(' | ')' | " ");
keyword = valid character, {valid character};
OpenSearch Description Document
```

The OpenSearch Description Document is an XML file is found inside of the OpenSearch Endpoint bundle and is named `ddf-os.xml`.

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`ddf.catalog.CatalogFramework`	required	false
`ddf.catalog.filter.FilterBuilder`	required	false

Exported Services

None.

Known Issues

None

4.2.3. 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.

Installing and Uninstalling

Use the Admin Menu to install the FTP feature in the DDF Catalog. The FTP Endpoint is not installed by default.

Configuring

The configurable properties for the FTP Endpoint are accessed from the **FTP Endpoint** Configuration in the Admin Console under the DDF Catalog application.

Using the FTP Endpoint

Using the endpoint

The FTP endpoint supports the PUT and MPUT operations. Any other operation, such as DELETE, will return a 550 action not taken response.

From Code:

Custom Ftplets can be implemented by extending the `DefaultFtplet` class provided by Apache FTP Server. Doing this will allow custom handling of various FTP commands by overriding the methods of the `DefaultFtplet`. Refer to `https://mina.apache.org/ftpserver-project/ftplet.html` for available methods that can be overridden. After creating a custom Ftplet, it needs to be added to the FTP server's Ftplets before the server is started. Any Ftplets that are registered to the FTP server will execute the FTP command in the order that they were registered.

From an FTP client:

The FTP endpoint can be accessed from any FTP client of choice. Some common clients are FileZilla, PuTTY, or the FTP client provided in the terminal. The default port number is **8021**. Valid usernames and password are stored in the `<INSTALL_DIR>/etc/users.properties` file.

Implementation Details

The FTP endpoint is implemented using the Apache FTP Server and Apache Mina. Apache Mina provides an API for transport protocols such as TCP and UDP. The FTP server is built on top of Apache Mina to transport its messages. DDF implements a custom Ftplet by overriding the `DefaultFtplet` class provided by Apache FTP Server.

Known Issues

None

4.2.4. Resource Download Endpoint

The Resource Download Endpoint provides a REST API to trigger downloading resources to the DDF cache only. The resource is not returned to the client.

Configuring

The Resource Download Endpoint has no configurable properties.

Using the Resource Download Endpoint

Using the endpoint

The Resource Download Endpoint WADL is accessible from `http://<DDF_HOST>:<DDF_PORT>/services/catalog/downloads?_wadl`.

NOTE	If resource caching is not enabled in DDF, the endpoint will return a `400 BAD REQUEST` response.

Operation	HTTP Request	Details	Example URL
`cache only`	HTTP GET	Resource caching must be enabled	http://<DDF_HOST>:<DDF_PORT>/services/catalog/downloads/<sourceId>/<metacardId>

Known Issues

None

4.3. Developing a New Endpoint

Complete the following procedure to create an endpoint.

1. Create a Java class that implements the endpoint's business logic. Example: Creating a web service that external clients can invoke.
2. Add the endpoint's business logic, invoking `CatalogFramework` calls as needed.
3. Import the DDF packages to the bundle's manifest for run-time (in addition to any other required packages):
   `Import-Package: ddf.catalog, ddf.catalog.*`
4. Retrieve an instance of `CatalogFramework` from the OSGi registry. (Refer to the Working with OSGi - Service Registry section for examples.)

Deploy the packaged service to DDF. (Refer to the Working with OSGi - Bundles section.)

NOTE	It is recommended to use the maven bundle plugin to create the Endpoint bundle's manifest as opposed to directly editing the manifest file.

TIP

No implementation of an interface is required

Unlike other DDF components that require you to implement a standard interface, no implementation of an interface is required in order to create an endpoint.

4.3.1. Common Endpoint Business Logic

Methods	Use
`Ingest`	Add, modify, and remove metadata using the ingest-related `CatalogFramework` methods: create, update, and delete.
`Query`	Request metadata using the `query` method.
`Source`	Get available `Source` information.
`Resource`	Retrieve products referenced in Metacards from Sources.
`Transform`	Convert common Catalog Framework data types to and from other data formats.

4.4. DDF Data Migration

Data migration is the process of moving metadata from one catalog provider to another. It is also the process of translating metadata from one format to another. Data migration is necessary when a user decides to use metadata from one catalog provider in another catalog provider. The following steps define the procedure for transferring metadata from one catalog provider to another catalog provider. In addition, the procedures define the steps for converting metadata to different data formats.

4.4.1. Set Up

Set up DDF as instructed in Starting DDF section.

4.4.2. Move Metadata from One Catalog Provider to Another

Export Metadata Out of Catalog Provider

1. Configure a desired catalog provider.
2. From the command line of DDF console, use the command to export all metadata from the catalog provider into serialized data files dump. The following example shows a command for running on Linux and a command for running on Windows.

ddf@local

```
dump "/myDirectory/exportFolder"
or
dump "C:/myDirectory/exportFolder"
```

Ingest Exported Metadata into Catalog Provider

1. Configure a different catalog provider.
2. From the command line of DDF console, use the **ingest** command to import exported metadata from serialized data files into catalog provider. The following example shows a command for running on Linux and a command for running on Windows.

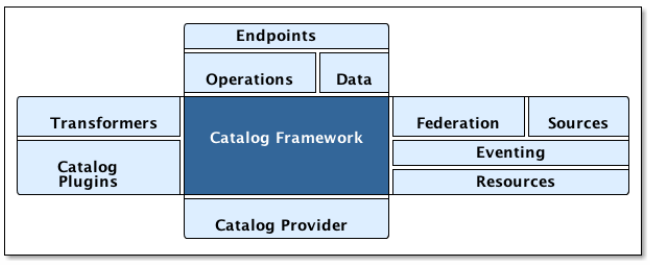
ddf@local

```
ingest -p "/myDirectory/exportFolder"
or
ingest -p "C:/myDirectory/exportFolder"
```

Translate Metadata from One Format to Another

Metadata can be converted from one data format to another format. Only the data format changes, but the content of the metadata does not, as long as `option -p` is used with the ingest command. The process for converting metadata is performed by ingesting a data file into a catalog provider in one format and dumping it out into a file in another format.

4.5. Integrating Catalog Framework



4.5.1. Catalog Framework

The Catalog Framework wires all Catalog components together. It is responsible for routing Catalog requests and responses to the appropriate target. Endpoints send Catalog requests to the Catalog Framework. The Catalog Framework then invokes Catalog Plugins, Transformers, and Resource Components as needed before sending requests to the intended destination, such as one or more Sources.

Example Catalog Frameworks

The Catalog comes with the following Catalog Frameworks out of the box:

- Catalog Framework
- Catalog Fanout Framework

Catalog Framework Sequence Diagrams

Because the Catalog Framework plays a central role to Catalog functionality, it interacts with many different Catalog components. To illustrate these relationships, high level sequence diagrams with notional class names are provided below. These examples are for illustrative purposes only and do not necessarily represent every step in each procedure.

Ingest

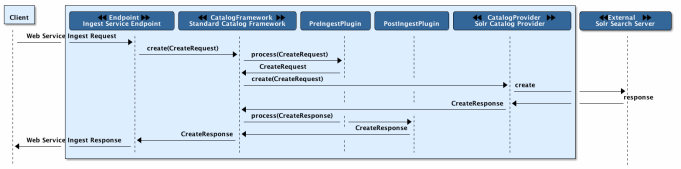


Figure 4. Ingest Request

The Ingest Service Endpoint, the Catalog Framework, and the Catalog Provider are key components of the Reference Implementation. The Endpoint bundle implements a Web service that allows clients to create, update, and delete metacards. The Endpoint calls the `CatalogFramework` to execute the operations of its specification. The `CatalogFramework` routes the request through optional `PreIngest` and `PostIngest` Catalog Plugins, which may modify the ingest request/response before/after the Catalog Provider executes the ingest request and provides the response. Note that a `CatalogProvider` must be present for any ingest requests to be successfully processed, otherwise a fault is returned.

This process is similar for updating catalog entries, with update requests calling the `update(UpdateRequest)` methods on the Endpoint, `CatalogFramework`, and Catalog Provider. Similarly, for deletion of catalog entries, the delete requests call the `delete(DeleteRequest)` methods on the `Endpoint`, `CatalogFramework`, and `CatalogProvider`.

Error Handling

Any ingest attempts that fail inside the Catalog Framework (whether the failure comes from the Catalog Framework itself, pre-ingest plugin failures, or issues with the Catalog Provider) will be logged to a separate log file for ease of error handling. The file is located at `data/log/ingest_error.log` and will log the Metacards that fail, their ID and Title name, and the stack trace associated with their failure. By default, successful ingest attempts are not logged. However, that functionality can be achieved by setting the log level of the `ingestLogger` to DEBUG (note that enabling DEBUG can cause a non-trivial performance hit).

TIP

To turn off logging failed ingest attempts into a separate file, execute the following via the command line console

```
log:set
 ERROR ingestLogger
```

Query

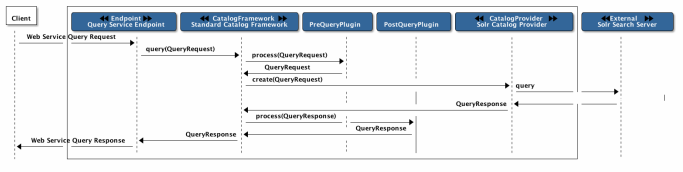


Figure 5. Ingest Request

The Query Service Endpoint, the Catalog Framework, and the `CatalogProvider` are key components for processing a query request as well. The Endpoint bundle contains a Web service that exposes the interface to query for `Metacards`. The Endpoint calls the `CatalogFramework` to execute the operations of its specification. The `CatalogFramework` relies on the `CatalogProvider` to execute the actual query. Optional PreQuery and PostQuery Catalog Plugins may be invoked by the `CatalogFramework` to modify the query request/response prior to the Catalog Provider processing the query request and providing the query response. If a `CatalogProvider` is not configured and no other remote Sources are configured, a fault will be returned. It is possible to have only remote Sources configured and no local `CatalogProvider` configured and be able to execute queries to specific remote Sources by specifying the site name(s) in the query request.

4.5.2. Product Retrieval

The Query Service Endpoint, the Catalog Framework, and the `CatalogProvider` are key components for processing a retrieve product request. The Endpoint bundle contains a Web service that exposes the interface to retrieve products, also referred to as Resources. The Endpoint calls the `CatalogFramework` to execute the operations of its specification. The `CatalogFramework` relies on the Sources to execute the actual product retrieval. Optional `PreResource` and `PostResource` Catalog Plugins may be invoked by the `CatalogFramework` to modify the product retrieval request/response prior to the Catalog Provider processing the request and providing the response. It is possible to retrieve products from specific remote Sources by specifying the site name(s) in the request.

Product Caching

The Catalog Framework optionally provides caching of products, so future requests to retrieve the same product will be serviced much quicker. If caching is enabled, each time a retrieve product request is received, the Catalog Framework will look in its cache (default location `<INSTALL_DIR>/data/product-cache`) to see if the product has been cached locally. If it has, the product is retrieved from the local site and returned to the client, providing a much quicker turnaround because remote product retrieval and network traffic was avoided. If the requested product is not in the cache,

the product is retrieved from the Source (local or remote) and cached locally while returning the product to the client. The caching to a local file of the product and the streaming of the product to the client are done simultaneously so that the client does not have to wait for the caching to complete before receiving the product. If errors are detected during the caching, caching of the product will be abandoned, and the product will be returned to the client.

The Catalog Framework attempts to detect any network problems during the product retrieval, e.g., long pauses where no bytes are read implying a network connection was dropped. (The amount of time defined as a "long pause" is configurable, with the default value being five seconds.) The Catalog Framework will attempt to retrieve the product up to a configurable number of times (default = three), waiting for a configurable amount of time (default = 10 seconds) between each attempt, trying to successfully retrieve the product. If the Catalog Framework is unable to retrieve the product, an error message is returned to the client.

If the admin has enabled the **Always Cache When Canceled** option, caching of the product will occur even if the client cancels the product retrieval so that future requests will be serviced quickly. Otherwise, caching is canceled if the user cancels the product download.

Product Download Status

As part of the caching of products, the Catalog Framework also posts events to the OSGi notification framework. Information includes when the product download started, whether the download is retrying or failed (after the number of retrieval attempts configured for product caching has been exhausted), and when the download completes. These events are retrieved by the Search UI and presented to the user who initiated the download.

4.6. DDF Schematron

Custom schematron rulesets can be used to validate metacard metadata. Multiple services can be created, and each service can have multiple rule sets 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.

4.6.1. Configuring

Schematron validation services are configured with a namespace and one or more schematron rule sets. Additionally, warnings may be suppressed so that only errors are reported. To create a new service, ensure that catalog-schematron-plugin is started and then click Schematron Validation Services.

4.7. Adding New Attribute Types, Metacard Types, and Validators Using JSON 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.

4.7.1. Definition File Format

Metacard types, attribute types, global attribute validators, and default attribute values can be defined within a definition file. A definition file follows the JSON format as specified in ECMA-404. All definition files must be valid JSON in order to be parsed. There are four main types that can be defined in a definition file.

- Metacard Types
- Attribute Types
- Global Attribute Validators
- Default Attribute Values

Within a definition file you may define as many of the four types as you wish. This means that types can be defined across multiple files for grouping or clarity.

4.7.2. Deploying

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.

4.7.3. Attribute Type Definition

To define Attribute Types, your definition file must have an `attributeTypes` key in the root object.

```
{
    "attributeTypes": {...}
}
```

The value of `attributeTypes` must be a map where each key is the attribute type's name and each value is a map that includes the data type and whether the attribute type is stored, indexed, tokenized, or multi-valued.

```
{
    "attributeTypes": {
        "temperature": {
            "type": "DOUBLE_TYPE",
            "stored": true,
            "indexed": true,
            "tokenized": false,
            "multivalued": false
        }
    }
}
```

The attributes `stored`, `indexed`, `tokenized`, and `multivalued` must be included and must have a boolean value. The `type` attribute must also be included and must have one of the following values:

- `DATE_TYPE`
- `STRING_TYPE`
- `XML_TYPE`
- `LONG_TYPE`
- `BINARY_TYPE`
- `GEO_TYPE`
- `BOOLEAN_TYPE`
- `DOUBLE_TYPE`
- `FLOAT_TYPE`
- `INTEGER_TYPE`
- `OBJECT_TYPE`
- `SHORT_TYPE`

An example with multiple attributes defined:

```
{
    "attributeTypes": {
        "resolution": {
            "type": "STRING_TYPE",
            "stored": true,
            "indexed": true,
            "tokenized": false,
            "multivalued": false
        },
        "target-areas": {
            "type": "GEO_TYPE",
            "stored": true,
            "indexed": true,
            "tokenized": false,
            "multivalued": true
        }
    }
}
```

4.7.4. Metacard Type Definition

To define Metacard Types, your definition file must have a `metacardTypes` key in the root object.

```
{
    "metacardTypes": [...]
}
```

The value of `metacardTypes` must be an array of Metacard Type Objects, which are composed of the `type` and `attributes` keys.

```
{
    "metacardTypes": [
        {
            "type": "my-metacard-type",
            "attributes": {...}
        }
    ]
}
```

The value of the `type` key is the name of the metacard type being defined.

The value of the `attributes` key is a map where each key is the name of an attribute type to include in

this metacard type and each value is a map with a single key named `required` and a boolean value. Required attributes are used for metacard validation - metacards that lack required attributes will be flagged with validation errors.

```
{
    "metacardTypes": [
        {
            "type": "my-metacard-type",
            "attributes": {
                "resolution": {
                    "required": true
                },
                "target-areas": {
                    "required": false
                },
                "expiration": {
                    "required": false
                },
                "point-of-contact": {
                    "required": true
                }
            }
        }
    ]
}
```

NOTE The DDF basic metacard attribute types are added to custom metacard types by default. If you wish to make any of them required by your metacard type, just include them in your `attributes` map and set `required` to `true`, as shown in the above example with `point-of-contact`.

You can define multiple metacard types in a single file:

```
{
    "metacardTypes": [
        {
            "type": "my-metacard-type",
            "attributes": {
                "resolution": {
                    "required": true
                },
                "target-areas": {
                    "required": false
                }
            }
        },
        {
            "type": "another-metacard-type",
            "attributes": {
                "effective": {
                    "required": true
                },
                "resolution": {
                    "required": false
                }
            }
        }
    ]
}
```

4.7.5. Validator Definition

To define Validators, your definition file must have a `validators` key in the root object.

```
{
    "validators": {...}
}
```

The value of `validators` is a map of the attribute name to a list of validators for that attribute.

```
{
    "validators": {
        "point-of-contact": [...]
    }
}
```

Each object in the list of validators is the validator name and list of arguments for that validator.

```
{
    "validators": {
        "point-of-contact": [
            {
                "validator": "pattern",
                "arguments": [".*regex.+\\s"]
            }
        ]
    }
}
```

WARNING	The value of the `arguments` key must always be an array of strings, even for numeric arguments, e.g. `["1", "10"]`

The `validator` key must have a value of one of the following:

- `size` (validates the size of Strings, Arrays, Collections, and Maps)
  - `arguments`: (2) [integer: lower bound (inclusive), integer: upper bound (inclusive)]
- `pattern`
  - `arguments`: (1) [regular expression]
- `pastdate`
  - `arguments`: (0) [NO ARGUMENTS]
- `futuredate`
  - `arguments`: (0) [NO ARGUMENTS]
- `range`
  - (2) [number (decimal or integer): inclusive lower bound, number (decimal or integer): inclusive upper bound]
    - uses a default epsilon of 1E-6 on either side of the range to account for floating point representation inaccuracies
  - (3) [number (decimal or integer): inclusive lower bound, number (decimal or integer): inclusive upper bound, decimal number: epsilon (the maximum tolerable error on either side of the range)]
- `enumeration`
  - `arguments`: (unlimited) [list of strings: each argument is one case-sensitive, valid enumeration value]

Examples:

```
{
    "validators": {
        "title": [
            {
                "validator": "size",
                "arguments": ["1", "50"]
            },
            {
                "validator": "pattern",
                "arguments": ["\\D+"]
            }
        ],
        "created": [
            {
                "validator": "pastdate",
                "arguments": []
            }
        ],
        "expiration": [
            {
                "validator": "futuredate",
                "arguments": []
            }
        ],
        "page-count": [
            {
                "validator": "range",
                "arguments": ["1", "500"]
            }
        ],
        "temperature": [
            {
                "validator": "range",
                "arguments": ["12.2", "19.8", "0.01"]
            }
        ],
        "resolution": [
            {
                "validator": "enumeration",
                "arguments": ["1080p", "1080i", "720p"]
            }
        ]
    }
}
```

4.7.6. Default Attribute Values

To define default attribute values, your definition file must have a `defaults` key in the root object.

```
{
    "defaults": [...]
}
```

The value of `defaults` is a list of objects where each object contains the keys `attribute`, `value`, and optionally `metacardTypes`.

```
{
    "defaults": [
        {
            "attribute": ...,
            "value": ...,
            "metacardTypes": [...]
        }
    ]
}
```

The value corresponding to the `attribute` key is the name of the attribute to which the default value will be applied. The value corresponding to the `value` key is the default value of the attribute.

NOTE

The attribute's default value must be of the same type as the attribute, but it has to be written as a string (i.e., enclosed in quotation marks) in the JSON file.

Dates must be UTC datetimes in the ISO 8601 format, i.e., `yyyy-MM-ddTHH:mm:ssZ`

The `metacardTypes` key is optional. If it is left out, then the default attribute value will be applied to every metacard that has that attribute. It can be thought of as a 'global' default value. If the `metacardTypes` key is included, then its value must be a list of strings where each string is the name of a metacard type. In this case, the default attribute value will be applied only to metacards that match one of the types given in the list.

NOTE

In the event that an attribute has a 'global' default value as well as a default value for a specific metacard type, the default value for the specific metacard type will be applied (i.e., the more specific default value wins).

Example:

```
{
    "defaults": [
        {
            "attribute": "title",
            "value": "Default Title"
        },
        {
            "attribute": "description",
            "value": "Default video description",
            "metacardTypes": ["video"]
        },
        {
            "attribute": "expiration",
            "value": "2020-05-06T12:00:00Z",
            "metacardTypes": ["video", "nitf"]
        },
        {
            "attribute": "frame-rate",
            "value": "30"
        }
    ]
}
```

5. Integrating DDF Platform

Version: 2.9.1

This section supports integration of this application with external frameworks.

5.1. Platform UI Settings

The Platform UI Settings are the system-wide configuration settings used throughout DDF to customize certain aspects of the DDF UI.

5.1.1. Configuration

The configurable properties for the platform-wide configuration are accessed from **DDF Platform Configuration** **Platform UI Configuration** in the Admin Console.

5.1.2. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Enable System Usage Message	`systemUsageEnabled`	Boolean	Turns on a system usage message, which is shown when the Search Application is opened		yes
System Usage Message Title	`systemUsageTitle`	String	A title for the system usage Message when the application is opened		yes
System Usage Message	`systemUsageMessage`	String	A system usage message to be displayed to the user each time the user opens the application		yes

Title	Property	Type	Description	Default Value	Required
Show System Usage Message once per session	systemUsageOncePerSession	Boolean	With this selected, the system usage message will be shown once for each browser session. Uncheck this to have the usage message appear every time the search window is opened or refreshed.	true	yes
Header	header	String	Specifies the header text to be rendered on all pages.		yes
Footer	footer	String	Specifies the footer text to be rendered on all pages.		yes
Text Color	color	String	Specifies the Text Color of the Header and Footer. Use html css colors or #rrggbb.		yes
Background Color	background	String	Specifies the Background Color of the Header and Footer. Use html css colors or #rrggbb.		yes

5.2. Landing Page

The DDF landing page offers a starting point and general information for a DDF node. It is accessible at /(index|home|landing(.htm|html)).

5.2.1. Configuration

The configurable properties for the landing page configuration are accessed from **DDF Platform Landing Page** in the Admin UI.

5.2.2. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Description	`description`	String	Specifies the description to display on the landing page.	As a common data layer, DDF provides secure enterprise-wide data access for both users and systems.	yes
Phone Number	`phone`	String	Specifies the phone number to display on the landing page.		yes
Email Address	`email`	String	Specifies the email address to display on the landing page.		yes
External Web Site	`externalUrl`	String	Specifies the external web site URL to display on the landing page.		yes
Announcements	`announcements`	String	Announcements that will be displayed on the landing page. Can be prefixed with a date of the form `mm/dd/yy`, leading zeroes not required.		yes

Title	Property	Type	Description	Default Value	Required
Branding Background	`background`	String	Specifies the landing page Background Color. Use html css colors or `#rrggbb`.		yes
Branding Foreground	`foreground`	String	Specifies the landing page Foreground Color. Use html css colors or `#rrggbb`.		yes
Branding Logo	`logo`	String	Specifies the landing page Logo. Use a base64 encoded image. You can use openssl to encode an image. `openssl base64 -in <infile> -out <outfile>` The contents of `<outfile>` should be pasted into this field.		yes

5.3. DDF Mime Framework

5.3.1. Mime Type Mapper

The MimeTypeMapper is the entry point in DDF for resolving file extensions to mime types, and vice versa.

`MimeTypeMappers` are used by the `ResourceReader` to determine the file extension for a given mime type in aid of retrieving a product. `MimeTypeMappers` are also used by the `FileSystemProvider` in the Catalog Framework to read a file from the content file repository.

The `MimeTypeMapper` maintains a list of all of the `MimeTypeResolvers` in DDF.

The `MimeTypeMapper` accesses each `MimeTypeResolver` according to its priority until the provided file extension is successfully mapped to its corresponding mime type. If no mapping is found for the file extension, `null` is returned for the mime type. Similarly, the `MimeTypeMapper` accesses each `MimeTypeResolver` according to its priority until the provided mime type is successfully mapped to its corresponding file extension. If no mapping is found for the mime type, `null` is returned for the file extension.

5.3.2. Included Mime Type Mappers

DDF Mime Type Mapper

The DDF Mime Type Mapper is the core implementation of the DDF Mime API. It provides access to all `MimeTypeResolvers` within DDF, which provide mapping of mime types to file extensions and file extensions to mime types.

Installing and Uninstalling

The DDF Mime Type Mapper is bundled in the `mime-core` feature, which is installed by default, as part of the `mime-core-app` application.

Configuring

There is no configuration needed for this feature.

5.3.3. Mime Type Resolver

A `MimeTypeResolver` is a DDF service that can map a file extension to its corresponding mime type and, conversely, can map a mime type to its file extension.

`MimeTypeResolvers` are assigned a priority (0-100, with the higher the number indicating the higher priority). This priority is used to sort all of the `MimeTypeResolvers` in the order they should be checked for mapping a file extension to a mime type (or vice versa). This priority also allows custom `MimeTypeResolvers` to be invoked before default `MimeTypeResolvers` if the custom resolver's priority is set higher than the default's.

`MimeTypeResolvers` are not typically invoked directly. Rather, the `MimeTypeMapper` maintains a list of `MimeTypeResolvers` (sorted by their priority) that it invokes to resolve a mime type to its file extension (or to resolve a file extension to its mime type).

Tika Mime Type Resolver

The `TikaMimeTypeResolver` is a `MimeTypeResolver` that is implemented using the Apache Tika open source product.

Using the Apache Tika content analysis toolkit, the `TikaMimeTypeResolver` provides support for resolving over 1300 mime types.

The `TikaMimeTypeResolver` is assigned a default priority of `-1` to insure that it is always invoked last by the `MimeTypeMapper`. This insures that any custom `MimeTypeResolvers` that may be installed will be invoked before the `TikaMimeTypeResolver`.

Using

The `TikaMimeTypeResolver` provides the bulk of the default mime type support for DDF.

Installing and Uninstalling

The `TikaMimeTypeResolver` is bundled as the `mime-tika-resolver` feature in the `mime-tika-app` application.

This feature is installed by default.

Configuring

There are no configuration properties for the `mime-tika-resolver`.

Implementation Details

Exported Services

Registered Interface	Service Property	Value
`ddf.mime.MimeTypeResolver`		`tika-mimetypes.xml`

Custom Mime Type Resolver

The Custom Mime Type Resolver is a `MimeTypeResolver` that defines the custom mime types that DDF will support out of the box. These are mime types not supported by the default `TikaMimeTypeResolver`.

Currently, the custom mime types supported by the Custom Mime Type Resolver that are configured for DDF out-of-the-box are:

File Extension	Mime Type
nitf	image/nitf
ntf	image/nitf
json	json=application/json;id=geojson

New custom mime type resolver mappings can be added using the Admin Console.

As a `MimeTypeResolver`, the Custom Mime Type Resolver will provide methods to map the file extension to the corresponding mime type, and vice versa.

Using

The Custom Mime Type Resolver is used when mime types need to be added that are not supported

by DDF out of the box. By adding custom mime type resolvers to DDF, new content with that mime type can be processed by DDF.

Installing and Uninstalling

One Custom Mime Type Resolver is configured and installed out of the box for the `image/nitf` mime type. This custom resolver is bundled in the `mime-core-app` application and is part of the `mime-core` feature.

Additional Custom Mime Type Resolvers can be added for other custom mime types.

Configuring

The configurable properties for the Custom Mime Type Resolver are accessed from the **MIME Custom Types** configuration in the Admin Console.

Managed Service Factory PID

- `DDF_Custom_Mime_Type_Resolver`

Table 9. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Resolver Name	`name`	String	Unique name for the custom mime type resolver.	N/A	Yes
Priority	`priority`	Integer	Execution priority of the resolver. Range is 0 to 100, with 100 being the highest priority.	10	Yes
File Extensions to Mime Types	`customMimeTypes`	String	Comma-delimited list of key/value pairs where key is the file extension and value is the mime type, e.g., `nitf=image/nitf`.	N/A	Yes

Implementation Details

Table 10. Imported Services

Registered Interface	Availability	Multiple
`ddf.catalog.transform.InputTransformer`	optional	true
`ddf.catalog.transform.QueryResponseTransformer`	optional	true
`ddf.mime.MimeTypeResolver`	optional	true

Table 11. Exported Services

Registered Interface	Service Property	Value
ddf.mime.MimeTypeToTransformerMapper		
ddf.mime.MimeTypeMapper		

5.4. Metrics Collection

The Metrics Collection collects data for all of the pre-configured metrics in DDF and stores them in custom JMX Management Bean (MBean) attributes. Samples of each metric's data is collected every 60 seconds and stored in the `<DDF_INSTALL_DIR>/data/metrics` directory with each metric stored in its own `.rrd` file. Refer to the Metrics Reporting Application for how the stored metrics data can be viewed.

WARNING

Do not remove the `<DDF_INSTALL_DIR>/data/metrics` directory or any files in it. If this is done, all existing metrics data will be permanently lost.

Also note that if DDF is uninstalled/re-installed that all existing metrics data will be permanently lost.

The metrics currently being collected by DDF are:

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Exceptions	ddf.metrics.catalog:name=Exceptions	Count	A count of the total number of exceptions, of all types, thrown across all catalog queries executed.
Catalog Exceptions Federation	ddf.metrics.catalog:name=Exceptions.Federation	Count	A count of the total number of Federation exceptions thrown across all catalog queries executed.
Catalog Exceptions Source Unavailable	ddf.metrics.catalog:name=Exceptions.SourceUnavailable	Count	A count of the total number of `SourceUnavailable` exceptions thrown across all catalog queries executed. These exceptions occur when the source being queried is currently not available.
Catalog Exceptions Unsupported Query	ddf.metrics.catalog:name=Exceptions.UnsupportedQuery	Count	A count of the total number of `UnsupportedQuery` exceptions thrown across all catalog queries executed. These exceptions occur when the query being executed is not supported or is invalid.

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Ingest Created	`ddf.metrics.catalog:name=Ingest.Created`	Count	A count of the number of catalog entries created in the Metadata Catalog.
Catalog Ingest Deleted	`ddf.metrics.catalog:name=Ingest.Deleted`	Count	A count of the number of catalog entries updated in the Metadata Catalog.
Catalog Ingest Updated	`ddf.metrics.catalog:name=Ingest.Updated`	Count	A count of the number of catalog entries deleted from the Metadata Catalog.
Catalog Queries	`ddf.metrics.catalog:name=Queries`	Count	A count of the number of queries attempted.
Catalog Queries Comparison	`ddf.metrics.catalog:name=Queries.Comparison`	Count	A count of the number of queries attempted that included a string comparison criteria as part of the search criteria, e.g., `PropertyIsLike`, `PropertyIsEqualTo`, etc.
Catalog Queries Federated	`ddf.metrics.catalog:name=Queries.Federated`	Count	A count of the number of federated queries attempted.
Catalog Queries Fuzzy	`ddf.metrics.catalog:name=Queries.Fuzzy`	Count	A count of the number of queries attempted that included a string comparison criteria with fuzzy searching enabled as part of the search criteria.
Catalog Queries Spatial	`ddf.metrics.catalog:name=Queries.Spatial`	Count	A count of the number of queries attempted that included a spatial criteria as part of the search criteria.
Catalog Queries Temporal	`ddf.metrics.catalog:name=Queries.Temporal`	Count	A count of the number of queries attempted that included a temporal criteria as part of the search criteria.
Catalog Queries Total Results	`ddf.metrics.catalog:name=Queries.TotalResults`	Mean	An average of the total number of results returned from executed queries. This total results data is averaged over the metric's sample rate.
Catalog Queries Xpath	`ddf.metrics.catalog:name=Queries.Xpath`	Count	A count of the number of queries attempted that included a Xpath criteria as part of the search criteria.

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Resource Retrieval	`ddf.metrics.catalog:name=Resource`	Count	A count of the number of products retrieved.
Services Latency	`ddf.metrics.services:name=Latency`	Mean	The response time (in milliseconds) from receipt of the request at the endpoint until the response is about to be sent to the client from the endpoint. This response time data is averaged over the metric's sample rate.

5.4.1. Source Metrics

Metrics are also collected on a per source basis for each configured Federated Source and Catalog Provider. When the source is configured, the metrics listed in the table below are automatically created. With each request that is either an enterprise query or a query that lists the source(s) to query these metrics are collected. When the source is deleted (or renamed), the associated metrics' MBeans and Collectors are also deleted. However, the RRD file in the `data/metrics` directory containing the collected metrics remain indefinitely and remain accessible from the Metrics tab in the Admin Console.

In the table below, the metric name is based on the Source's ID (indicated by `<sourceId>`).

Metric	JMX MBean Name	MBean Attribute Name	Description
Source <sourceId> Exceptions	`ddf.metrics.catalog.source:name=<sourceId>.Exceptions`	Count	A count of the total number of exceptions, of all types, thrown from catalog queries executed on this source.
Source <sourceId> Queries	`ddf.metrics.catalog.source:name=<sourceId>.Queries`	Count	A count of the number of queries attempted on this source.
Source <sourceId> Queries Total Results	`ddf.metrics.catalog.source:name=<sourceId>.Queries.TotalResults`	Mean	An average of the total number of results returned from executed queries on this source. This total results data is averaged over the metric's sample rate.

For example, if a Federated Source was created with a name of `fs-1`, then the following metrics would be created for it:

- `Source Fs1 Exceptions`
- `Source Fs1 Queries`
- `Source Fs1 Queries Total Results`

If this federated source is then renamed to `fs-1-rename`, the MBeans and Collectors for the `fs-1` metrics are deleted, and new MBeans and Collectors are created with the new names:

- `Source Fs1 Rename Exceptions`
- `Source Fs1 Rename Queries`
- `Source Fs1 Rename Queries Total Results`

Note that the metrics with the previous name remain on the Metrics tab because the data collected while the Source had this name remains valid and thus needs to be accessible. Therefore, it is possible to access metrics data for sources renamed months ago, i.e., until DDF is reinstalled or the metrics data is deleted from the `<DDF_INSTALL_DIR>/data/metrics` directory. Also note that the source metrics' names are modified to remove all non-alphanumeric characters and renamed in camelCase.

5.4.2. Usage

The Metrics Collection is used when collection of historical metrics data, such as catalog query metrics, message latency, or individual sources' metrics type of data, is desired.

5.4.3. Install and Uninstall

The Metrics Collecting application is installed by default.

The catalog level metrics (packaged as the `catalog-core-metricsplugin` feature) can be installed and uninstalled using the normal processes described in the Configuration section.

Similarly, the source-level metrics (packaged as the `catalog-core-sourcemetricsplugin` feature) can be installed and uninstalled using the normal processes described in the Configuration section.

5.4.4. Configuration

No configuration is made for the Metrics Collecting application. All of the metrics that it collects data on are either pre-configured in DDF out of the box or dynamically created as sources are created or deleted.

5.4.5. Known Issues

None

5.5. Metrics Reporting Application

The DDF Metrics Reporting application provides access to historical data in a graphic, a comma-separated values file, a spreadsheet, a PowerPoint file, XML, and JSON formats for system metrics collected while DDF is running. Aggregate reports (weekly, monthly, and yearly) are also provided where all collected metrics are included in the report. Aggregate reports are available in Excel and PowerPoint formats.

5.5.1. Usage

The DDF Metrics Reporting application provides a plugin that adds a new tab to the Admin Console with the title of Metrics. When selected, the Metrics tab displays a list of all of the metrics being collected by DDF, e.g., Catalog Queries, Catalog Queries Federated, Catalog Ingest Created, etc.

With each metric in the list, a set of hyperlinks is displayed under each column. Each column's header is displayed with the available time ranges. The time ranges currently supported are 15 minutes, 1 hour, 1 day, 1 week, 1 month, 3 months, 6 months, and 1 year, measured from the time that the hyperlink is clicked.

All metrics reports are generated by accessing the collected metric data stored in the `<DDF_INSTALL_DIR>/data/metrics` directory. All files in this directory are generated by the JmxCollector using RRD4J, a Round Robin Database for a Java open source product. All files in this directory will have the `.rrd` file extension and are binary files, hence they cannot be opened directly. These files should only be accessed using the Metrics tab's hyperlinks. There is one RRD file per metric being collected. Each RRD file is sized at creation time and will never increase in size as data is collected. One year's worth of metric data requires approximately 1 MB file storage.

WARNING

Do not remove the `<DDF_INSTALL_DIR>/data/metrics` directory or any files in the directory. If this is done, all existing metrics data will be permanently lost.

Also note that if DDF is uninstalled/re-installed, all existing metrics data will be permanently lost.

There is a hyperlink per format in which the metric's historical data can be displayed. For example, the PNG hyperlink for 15m for the Catalog Queries metric maps to \http://<DDF_HOST>:<DDF_PORT>/services/internal/metrics/catalogQueries.png?dateOffset=900, where the `dateOffset=900` indicates the previous 900 seconds (15 minutes) to be graphed.

Note that the date format will vary according to the regional/locale settings for the server.

All of the metric graphs displayed are in PNG format and are displayed on their own page. The user may use the back button in the browser to return to the Admin Console, or, when selecting the hyperlink for a graph, they can use the right mouse button in the browser to display the graph in a separate browser tab or window, which will keep the Admin console displayed. The screen shot below is a sample graph of the Catalog Queries metrics data for the previous 15 minutes from when the link was selected. Note that the y-axis label and the title use the metrics name (Catalog Queries) by default. The average min and max of all of the metrics data is summarized in the lower left corner of the graph.

The user can also specify custom time ranges by adjusting the URL used to access the metric's graph.

The Catalog Queries metric data may also be graphed for a specific time range by specifying the `startDate` and `endDate` query parameters in the URL.

WARNING	Note that the Metrics endpoint URL says "internal." This indicates that this endpoint is intended for internal use by the DDF code. This endpoint is likely to change in future versions; therefore, any custom applications built to make use of it, as described below, should be made with caution.

For example, to map the Catalog Queries metric data for March 31, 6:00 am, to April 1, 2013, 11:00 am, (Arizona timezone, which is -07:00) the URL would be:

```
http://<DDF_HOST><DDF_PORT>/services/internal/metrics/catalogQueries.png?startDate=2013-03-31T06:00:00-07:00&endDate=2013-04-01T11:00:00-07:00
```

Or to view the last 30 minutes of data for the Catalog Queries metric, a custom URL with a `dateOffset=1800` (30 minutes in seconds) could be used:

```
http://<DDF_HOST>:<DDF_PORT>/services/internal/metrics/catalogQueries.png?dateOffset=1800
```

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

Parameter	Description	Example
`startDate`	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. *This parameter cannot be used with the `dateOffset` parameter.*	`startDate=2013-03-31T06:00:00-07:00`
`endDate`	Specifies the endof 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. *This parameter cannot be used with the `dateOffset` parameter.*	`endDate=2013-04-01T11:00:00-07:00`

Parameter	Description	Example
`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. *This parameter cannot be used with the `startDate` or `endDate` parameters.*	`dateOffset=1800`
`yAxisLabel`	(optional) the label to apply to the graph's y-axis. Will default to the metric's name, e.g., Catalog Queries. _This parameter is only applicable for the metric's graph display format.	Catalog Query Count
`title`	(optional) the title to be applied to the graph. Will default to the metric's name plus the time range used for the graph. *This parameter is only applicable for the metric's graph display format.*	Catalog Query Count for the last 15 minutes

5.5.2. Metric 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://<DDF_HOST>:<DDF_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 \http://<DDF_HOST>:<DDF_PORT> which is omitted in the table for brevity.)

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=my%20label&title=my%20graph%20title`
			
			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=my%20label&title=my%20graph%20title`
			
			Note that the `yAxisLabel` *and* `title` *parameters are optional.*

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 are the column headers and the remaining rows are 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`

Display Format	Description	How To Access	Example URL

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`

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`

Display Format	Description	How To Access	Example URL

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`
			
			Sample XML-formatted output would look like:
			
			[source,xml,linenums]
			---- 	
			<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>
			---- 	

Display Format	Description	How To Access	Example URL

JSON	Displays the metric's data as an JSON-formatted file.	via custom URL only	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`
			
			.Sample JSON-formatted Output
			[source,json,linenums]
			---- 	
			{
			"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
			}
			]
			}
			---- 	

5.5.3. Metrics Aggregate Reports

The Metrics tab also provides aggregate reports for the collected metrics. These are reports that include data for all of the collected metrics for the specified time range.

The aggregate reports provided are:

- Weekly reports for each week up to the past four **complete** weeks from current time. A complete week is defined as a week from Monday through Sunday. For example, if current time is Thursday, April 11, 2013, the past complete week would be from April 1 through April 7.

- Monthly reports for each month up to the past 12 **complete** months from current time. A complete month is defined as the full month(s) preceding current time. For example, if current time is

Thursday, April 11, 2013, the past complete 12 months would be from April 2012 through March 2013.

- Yearly reports for the past **complete** year from current time. A complete year is defined as the full year preceding current time. For example, if current time is Thursday, April 11, 2013, the past complete year would be 2012.

An aggregate report in XLS format would consist of a single workbook (spreadsheet) with multiple worksheets in it, where a separate worksheet exists for each collected metric's data. Each worksheet would display:

- the metric's name and the time range of the collected data,
- two columns: Timestamp and Value, for each sample of the metric's data that was collected during the time range, and
- a total count (if applicable) at the bottom of the worksheet.

An aggregate report in PPT format would consist of a single slideshow with a separate slide for each collected metric's data. Each slide would display:

- a title with the metric's name,
- the PNG graph for the metric's collected data during the time range, and
- a total count (if applicable) at the bottom of the slide.

Hyperlinks are provided for each aggregate report's time range in the supported display formats, which include Excel (XLS) and PowerPoint (PPT). Aggregate reports for custom time ranges can also be accessed directly via the URL:

```
http://<DDF_HOST>:<DDF_PORT>/services/internal/metrics/report.<format>?startDate=<start_date_value>&endDate=<end_date_value>
```

where `<format>` is either `xls` or `ppt` and the `<start_date_value>` and `<end_date_value>` specify the custom time range for the report.

The table below list several examples for custom aggregate reports. (NOTE: all example URLs begin with http://<DDF_HOST>:<DDF_PORT> which is omitted in the table for brevity.)

Description	**URL**
XLS aggregate report for March 15, 2013 to April 15, 2013	`/services/internal/metrics/report.xls?startDate=2013-03-15T12:00:00-07:00&endDate=2013-04-15T12:00:00-07:00`
XLS aggregate report for last 8 hours	`/services/internal/metrics/report.xls?dateOffset=28800`

Description	URL
PPT aggregate report for March 15, 2013 to April 15, 2013	`/services/internal/metrics/report.ppt?startDate=2013-03-15T12:00:00-07:00&endDate=2013-04-15T12:00:00-07:00`
PPT aggregate report for last 8 hours	`/services/internal/metrics/report.ppt?dateOffset=28800`

5.5.4. 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.

5.5.5. Install and Uninstall

The Metrics Reporting application can be installed and uninstalled using the normal processes described in the Configuring DDF section.

5.5.6. Configuration

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

The `metrics-reporting` feature can only be installed and uninstalled. It is installed by default.

5.5.7. Known Issues

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.

5.6. Security Core API

The Security Core API contains all of the DDF Security Framework APIs that are used to perform security operations within DDF.

5.6.1. Install and Uninstall

The Security Services App installs this bundle by default. Do not uninstall the Security Core API as it is integral to system function and all of the other security services depend upon it.

5.6.2. Configuration

None

5.6.3. Implementation Details

Imported Services

None

Exported Services

None

5.7. Compression Services

The compression services offer CXF-based message encoding that allows for compression of outgoing and incoming messages.

5.7.1. Install and Uninstall

The compression services are not installed by default within the platform application. Installing them can be done by doing:

```
feature:install compression-[DESIRED COMPRESSION SERVICE]
```

Where [DESIRED COMPRESSION SERVICE] is one of the following:

Compression Type	Description
`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. More information is available at EXI.
`gzip`	Adds GZip compression to in and outgoing messages through CXF components. Code comes with CXF.

WARNING	Due to the way CXF features work, 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.

5.7.2. Configuration

None

5.7.3. Implementation Details

Imported Services

None

Exported Services

Registered Interface	Implemented Class(es)	Service Property	Value

`org.apache.cxf.feature.Feature`	`ddf.compression.exi.EXIFeature`
	
	`org.apache.cxf.transport.common.gzip.GZIPFeature`
		N/A	N/A

6. Integrating DDF Security

Version: 2.9.1

This section supports integration of this application with external frameworks.

6.1. Security CAS

The Security CAS app contains all of the services and implementations needed to integrate with the *Central Authentication Server* (CAS).

Information on setting up and configuring the CAS server is located on the CAS SSO Configuration page.

6.1.1. Components

Bundle Name	Feature Located In	Description/Link to Bundle Page
`security-cas-client`	`security-cas-client`	Security CAS Client
`security-cas-impl`	`security-cas-client`	Security CAS Implementation
`security-cas-tokenvalidator`	`security-cas-tokenvalidator`	Security CAS Token Validator
`security-cas-cxfservletfilter`	`security-cas-cxfservletfilter`	Security CAS CXF Servlet Filter
`security-cas-server`		Security CAS Server

6.1.2. Security CAS Client

The Security CAS client bundle contains client files needed by components that are performing authentication with CAS. This includes setting up the CAS SSO servlet filters and starting a callback service that is needed to request proxy tickets from CAS.

Installing CAS Client

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

Configuring CAS Client

Table 12. Settings

Configuration Name	Default Value	Additional Description
Server Name	https://server:8993	This is the name of the server that is calling CAS. The URL is used during CAS redirection to redirect back to the calling server.
CAS Server URL	https://cas:8443/cas	The main URL to the CAS Web application.
CAS Server Login URL	https://cas:8443/cas/login	URL to the login page of CAS (generally ends in /login)
Proxy Callback URL	https://server:8993/sso	Full URL of the callback service that CAS hits to create proxy tickets.
Proxy Receptor URL	`/sso`	

Implementation Details

Imported Services

None

Exported Services

Registered Interface	Implementation Class	Properties Set
`javax.servlet.Filter`	`ddf.security.cas.client.ProxyFilter`	CAS Filters

6.1.3. Security CAS Implementation

The Security CAS implementation bundle contains CAS-specific implementations of classes from the Security Core API. Inside this bundle is the `ddf.security.service.impl.cas.CasAuthenticationToken` class. It is an implementation of the `AuthenticationToken` class that is used to pass Authentication Credentials to the Security Framework.

Configuring

None.

Implementation Details

Imported Services

None

Exported Services

None

6.1.4. Security CAS Server

The Security CAS Server project creates a web application (.war) file that is configured to be deployed to a tomcat application server.

Configuring Security CAS Server

N/A - Not a bundle

Implementation Details

N/A - Not a bundle

6.1.5. Security CAS Token Validator

The Security CAS `TokenValidator` bundle exports a `TokenValidator` service that is called by the STS to validate CAS proxy tickets.

Installing

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

Configuring

Settings

Configuration Name	Default Value	Additional Description
CAS Server URL	https://localhost:8443/cas/	The hostname in the URL should match the hostname alias defined within the certificate that CAS is using for SSL communication.

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`ddf.security.encryption.EncryptionService`	required	false

Exported Services

Registered Interfaces	Implementation Class	Properties Set
`org.apache.cxf.sts.token.validator.TokenValidator`	`ddf.security.cas.WebSSOTokenValidator`	CAS Server URL and Encryption Service reference

6.1.6. Security CAS CXF Servlet Filter

The Security CAS CXF Servlet Filter bundle binds a list of CAS servlet filters to the CXF servlet. The servlet filters are defined by by the `security-cas-client` bundle.

Installing

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

Configuring

Settings

Configuration Name	Default Value	Additional Description
URL Pattern	/services/catalog/*	This defines the servlet URL that should be binded to the CAS filter. By default, they will bind to the REST and OpenSearch endpoints. The REST endpoint is called by the SearchUI when accessing individual metadata about a metacard and when accessing the metacard's thumbnail. An example of just securing the OpenSearch endpoint would be the value: `/services/catalog/query`.

WARNING	Endpoints that are secured by the CXF Servlet Filters will not currently work with federation. With the default settings, REST and OpenSearch federation to the site with this feature installed will not work. Federation from this site, however, will work normally.

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`javax.servlet.Filter`	required	false

Exported Services

None (filter is exported inside the code and not via configuration)

6.2. Security Core

The Security Core app contains all of the necessary components that are used to perform security operations (authentication, authorization, and auditing) required in the framework.

6.2.1. Components

Bundle Name	Located in Feature	Description / Link to Bundle Page
`security-core-api`	`security-core`	Security Core API
`security-core-impl`	`security-core`	Security Core Implementation
`security-core-commons`	`security-core`	Security Core Commons

6.2.2. Security Core Commons

The Security Core Commons bundle contains helper and utility classes that are used within DDF to help with performing common security operations. Most notably, this bundle contains the `ddf.security.common.audit.SecurityLogger` class that performs the security audit logging within DDF.

Configuring

None

Implementation Details

Imported Services

None

Exported Services

None

6.2.3. Security Core Implementation

The Security Core Implementation contains the reference implementations for the Security Core API interfaces that come with the DDF distribution.

Configuring

None

Install and Uninstall

The Security Core app installs this bundle by default. It is recommended to use this bundle as it contains the reference implementations for many classes used within the DDF Security Framework.

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`org.apache.shiro.realm.Realm`	optional	true

Exported Services

Registered Interface	Implementation Class	Properties Set
`ddf.security.service.SecurityManager`	`ddf.security.service.impl.SecurityManagerImpl`	None

6.2.4. Security Encryption

The DDF Security Encryption application offers an encryption framework and service implementation for other applications to use. This service is commonly used to encrypt and decrypt default passwords that are located within the metatype and Administration Web Console.

Components

Bundle Name	Feature Located In	Description/Link to Bundle Page
`security-encryption-api`	`security-encryption`	Security Encryption API
`security-encryption-impl`	`security-encryption`	Security Encryption Implementation
`security-encryption-commands`	`security-encryption`	Security Encryption Commands

6.3. Security Encryption API

The Security Encryption API bundle provides the framework for the encryption service. Applications that use the encryption service should import this bundle and use the interfaces defined within it instead of calling an implementation directly.

6.3.1. Installing Security Encryption API

This bundle is installed by default as part of the `security-encryption` feature. Many applications that come with DDF depend on this bundle and it should not be uninstalled.

6.3.2. Configuring

None

6.3.3. Implementation Details

Imported Services

None

Exported Services

None

6.3.4. Security Encryption Commands

The Security Encryption Commands bundle enhances the DDF system console by allowing administrators and integrators to encrypt and decrypt values directly from the console. More information and sample commands are available on the Encryption Service page.

Installing Security Encryption Commands

This bundle is installed by default by the `security-encryption` feature. This bundle is tied specifically to the DDF console and can be uninstalled without causing any issues to other applications. When uninstalled, however, administrators will not be able to encrypt and decrypt data from the console.

Configuring

None

Implementation Details

Imported Services

None

Exported Services

None

6.3.5. Security Encryption Implementation

The Security Encryption Implementation bundle contains all of the service implementations for the Encryption Framework and exports those implementations as services to the OSGi service registry.

Installing Security Encryption Implementation

This bundle is installed by default as part of the `security-encryption` feature. Other projects are dependent on the services this bundle exports and it should not be uninstalled unless another security service implementation is being added.

Configuring

None

Implementation Details

Imported Services

None

Exported Services

Registered Interface	Implementation Class	Properties Set
`ddf.security.encryption.EncryptionService`	`ddf.security.encryption.impl.EncryptionServiceImpl`	Key

6.4. Security LDAP

The DDF LDAP application allows the user to configure either an embedded or a standalone LDAP server. The provided features contain a default set of schemas and users loaded to help facilitate authentication and authorization testing.

6.4.1. Components

Bundle Name	Feature Located In	Description/Link to Bundle Page
`opendj-embedded-server`	`opendj-embedded`	Embedded LDAP Configuration

6.5. Installing the Embedded LDAP Server

The embedded OpenDJ LDAP server can be installed for testing purposes.

6.5.1. Run the Embedded LDAP Instance

1. Unzip and install DDF
2. Run the installer at https://localhost:8993/admin
3. After the install is complete, click the **Manage** button in the upper right hand corner
4. Click the **Play** icon on the Opendj Embedded application tile
5. The embedded LDAP is now installed.

6.5.2. Configuration

The configuration options are located on the Admin Console under **Opendj Embedded Configuration** **LDAP Server**. It currently contains three configuration options.

The configuration options are located on the standard DDF configuration Admin Console under the

title **LDAP Server**. It currently contains three configuration options.

Configuration Name	Description
LDAP Port	Sets the port for LDAP (plaintext and startTLS). 0 will disable the port.
LDAPS Port	Sets the port for LDAPS. 0 will disable the port.
Base LDIF File	Location on the server for a LDIF file. This file will be loaded into the LDAP and overwrite any existing entries. This option should be used when updating the default groups/users with a new LDIF file for testing. The LDIF file being loaded may contain any LDAP entries (schemas, users, groups, etc.). If the location is left blank, the default base LDIF file will be used that comes with DDF.

6.5.3. Trust Certificates

For LDAPS or startTLS to function correctly, it is important that the LDAP server is configured with a keystore file that trusts the clients it is connecting to and vice versa. Complete the following procedure to provide your own keystore information for the LDAP.

1. The embedded LDAP server is not recommended for production use as it has hardcoded keystores and truststores with localhost certificates. These cannot be changed unless the embedded server bundle is re-built with new stores.

6.5.4. Connect to Standalone LDAP Servers

DDF instances can connect to external LDAP servers by installing and configuring the `security-sts-ldaplogin` and `security-sts-ldapclaimshandler` features detailed here.

In order to connect to more than one LDAP server, configure these features for each LDAP server.

6.5.5. Embedded LDAP Configuration

The Embedded LDAP application contains an LDAP server (OpenDJ version 2.6.2) that has a default set of schemas and users loaded to help facilitate authentication and authorization testing.

6.5.6. Default Settings

Ports

Protocol	Default Port
`LDAP`	1389
`LDAPS`	1636

Protocol	Default Port
StartTLS	1389

Users

LDAP Users

Username	Password	Groups	Description
testuser1	password1		General test user for authentication
testuser2	password2		General test user for authentication
nromanova	password1	avengers	General test user for authentication
lcage	password1	admin, avengers	General test user for authentication, Admin user for karaf
jhowlett	password1	admin, avengers	General test user for authentication, Admin user for karaf
pparker	password1	admin, avengers	General test user for authentication, Admin user for karaf
jdrew	password1	admin, avengers	General test user for authentication, Admin user for karaf
tstark	password1	admin, avengers	General test user for authentication, Admin user for karaf
bbanner	password1	admin, avengers	General test user for authentication, Admin user for karaf
srogers	password1	admin, avengers	General test user for authentication, Admin user for karaf
admin	admin	admin	Admin user for karaf

LDAP Admin

Username	Password	Groups	Attributes	Description
admin	secret			Administrative User for LDAP

Schemas

The default schemas loaded into the LDAP instance are the same defaults that come with OpenDJ.

Schema File Name	Schema Description
00-core.ldif	This file contains a core set of attribute type and objectclass definitions from several standard LDAP documents, including draft-ietf-boreham-numsubordinates, draft-findlay-ldap-groupofentries, draft-furuseth-ldap-untypedobject, draft-good-ldap-changelog, draft-ietf-ldup-subentry, draft-wahl-ldap-adminaddr, RFC 1274, RFC 2079, RFC 2256, RFC 2798, RFC 3045, RFC 3296, RFC 3671, RFC 3672, RFC 4512, RFC 4519, RFC 4523, RFC 4524, RFC 4530, RFC 5020, and X.501.
01-pwpolicy.ldif	This file contains schema definitions from draft-behera-ldap-password-policy, which defines a mechanism for storing password policy information in an LDAP directory server.
02-config.ldif	This file contains the attribute type and objectclass definitions for use with the directory server configuration.
03-changelog.ldif	This file contains schema definitions from draft-good-ldap-changelog, which defines a mechanism for storing information about changes to directory server data.
03-rfc2713.ldif	This file contains schema definitions from RFC 2713, which defines a mechanism for storing serialized Java objects in the directory server.
03-rfc2714.ldif	This file contains schema definitions from RFC 2714, which defines a mechanism for storing CORBA objects in the directory server.
03-rfc2739.ldif	This file contains schema definitions from RFC 2739, which defines a mechanism for storing calendar and vCard objects in the directory server. Note that the definition in RFC 2739 contains a number of errors, and this schema file has been altered from the standard definition in order to fix a number of those problems.
03-rfc2926.ldif	This file contains schema definitions from RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.
03-rfc3112.ldif	This file contains schema definitions from RFC 3112, which defines the authentication password schema.
03-rfc3712.ldif	This file contains schema definitions from RFC 3712, which defines a mechanism for storing printer information in the directory server.
03-uddiv3.ldif	This file contains schema definitions from RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.
04-rfc2307bis.ldif	This file contains schema definitions from the draft-howard-rfc2307bis specification, used to store naming service information in the directory server.
05-rfc4876.ldif	This file contains schema definitions from RFC 4876, which defines a schema for storing Directory User Agent (DUA) profiles and preferences in the directory server.

Schema File Name	Schema Description
`05-samba.ldif`	This file contains schema definitions required when storing Samba user accounts in the directory server.
`05-solaris.ldif`	This file contains schema definitions required for Solaris and OpenSolaris LDAP naming services.
`06-compat.ldif`	This file contains the attribute type and `objectclass` definitions for use with the directory server configuration.

6.5.7. Configuration

Start and Stop

The embedded LDAP application installs a feature with the name `ldap-embedded`. Installing and uninstalling this feature will start and stop the embedded LDAP server. This will also install a fresh instance of the server each time. If changes need to persist, stop then start the `embedded-ldap-opendj` bundle (rather than installing/uninstalling the feature).

All settings, configurations, and changes made to the embedded LDAP instances are persisted across DDF restarts. If DDF is stopped while the LDAP feature is installed and started, it will automatically restart with the saved settings on the next DDF start.

Settings

The configuration options are located on the standard DDF configuration Admin Console under the title LDAP Server. It currently contains three configuration options.

Configuration Name	Description
LDAP Port	Sets the port for LDAP (`plaintext` and `StartTLS`). 0 will disable the port.
LDAPS Port	Sets the port for LDAPS. 0 will disable the port.
Base LDIF File	Location on the server for a LDIF file. This file will be loaded into the LDAP and overwrite any existing entries. This option should be used when updating the default groups/users with a new ldif file for testing. The LDIF file being loaded may contain any ldap entries (schemas, users, groups..etc). If the location is left blank, the default base LDIF file will be used that comes with DDF.

6.5.8. Limitations

Current limitations for the embedded LDAP instances include:

- Inability to store the LDAP files/storage outside of the DDF installation directory. This results in any

LDAP data (i.e., LDAP user information) being lost when the `ldap-embedded` feature is uninstalled.

- Cannot be run standalone from DDF. In order to run `embedded-ldap`, the DDF must be started.

6.5.9. External Links

Location to the default base LDIF file in the DDF source code.

OpenDJ documentation

6.5.10. LDAP Administration

OpenDJ provides a number of tools for LDAP administration. Refer to the OpenDJ Admin Guide.

Download the Admin Tools

Download OpenDJ (Version 2.4.6) and the included tool suite.

Use the Admin Tools

The admin tools are located in `<opendj-installation>/bat` for Windows and `<opendj-installation>/bin` for **`nix.` `These tools can be used to administer both local and remote LDAP servers by setting the *host` and **port** parameters appropriately.

Example Commands for Disabling/Enabling a User's Account

In this example, the user **Bruce Banner (uid=bbanner)** is disabled using the **manage-account** command on Windows. Run **manage-account --help** for usage instructions.

```
D:\OpenDJ-2.4.6\bat>manage-account set-account-is-disabled -h localhost -p 4444 -O true
-D "cn=admin" -w secret -b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Account Is Disabled:  true
```

Verify the Account is Disabled

Notice `Account Is Disabled: true` in the listing.

```
D:\OpenDJ-2.4.6\bat>manage-account get-all -h localhost -p 4444  -D "cn=admin" -w secret
-b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Password Policy DN:  cn=Default Password Policy,cn=Password Policies,cn=config
Account Is Disabled:  true
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

Enable the Account

```
D:\OpenDJ-2.4.6\bat>manage-account clear-account-is-disabled  -h localhost -p 4444  -D
"cn=admin" -w secret -b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Account Is Disabled:  false
```

Verify the Account is Enabled

Notice `Account Is Disabled: false` in the listing.

```
D:\OpenDJ-2.4.6\bat>manage-account get-all -h localhost -p 4444  -D "cn=admin" -w secret
-b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Password Policy DN:  cn=Default Password Policy,cn=Password Policies,cn=config
Account Is Disabled:  false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

6.6. Security PEP

The DDF Security Policy Enforcement Point (PEP) application contains bundles and services that enable service and metacard authorization. These two types of authorization can be installed separately and extended with custom services.

6.6.1. Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
security-pep-interceptor	security-pep-serviceauthz	Security PEP Interceptor

6.6.2. Security PEP Interceptor

The Security PEP Interceptor bundle contains the

`ddf.security.pep.interceptor.EPAuthorizingInterceptor` class. This class uses CXF to intercept incoming SOAP messages and enforces service authorization policies by sending the service request to the security framework.

Installing Security PEP Interceptor

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

WARNING	To perform service authorization within a default install of DDF, this bundle MUST be installed.

Configuring

None

Implementation Details

Imported Services

None

Exported Services

None

6.7. 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.

6.7.1. Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
`security-sts-realm`	`security-sts-realm`	Security STS Realm
`security-sts-ldaplogin`	`security-sts-ldaplogin`	Security STS LDAP Login
`security-sts-ldapclaimshandler`	`security-sts-ldapclaimshandler`	Security STS LDAP Claims Handler
`security-sts-server`	`security-sts-server`	Security STS Server
`security-sts-samlvalidator`	`security-sts-server`	Contains the default CXF SAML validator, exposes it as a service for the STS.

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

6.7.2. Security STS Client Config

The DDF 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.

Installing Security STS Client Config

This bundle is installed by default.

Configuring

Settings can be found in the Admin Console under **DDF Security Configuration Security STS Client**.

Configuration Name	Default Value	Additional Information
SAML Assertion Type	SAML v2.0	The version of SAML to use. Most services require SAML v2.0. Changing this value from the default could cause services to stop responding.
SAML Key Size	256	The key type to use with SAML. Most services require Bearer. Changing this value from the default could cause services to stop responding.
Use Key	true	Signals whether or not the STS Client should supply a public key to embed as the proof key. Changing this value from the default could cause services to stop responding.
STS WSDL Address	https://localhost:8993/services/SecurityTokenService?wsdl	The hostname of the remote server should match the certificate that the server is using.
STS Endpoint Name	`{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}STS_Port`	
STS Service Name	`{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}SecurityTokenService`	

Configuration Name	Default Value	Additional Information
Signature Properties	`etc/ws-security/server/signature.properties`	Path to Signature crypto properties. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.
Encryption Properties	`etc/ws-security/server/encryption.properties`	Path to Encryption crypto properties file. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.
STS Properties	`etc/ws-security/server/signature.properties`	Path to STS crypto properties file. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.
Claims	<List of Claims>	List of claims that should be requested by the STS Client.

6.7.3. Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`org.osgi.service.cm.ConfigurationAdmin`	required	false

Exported Services

None

6.7.4. External/WS-S STS Support

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.

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).

6.7.5. Security STS LDAP Claims Handler

The DDF 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.

Installing Security STS LDAP Claims Handler

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

Configuring

Settings

Settings can be found in the Admin Console under **DDF Security Configuration Security STS LDAP and Roles Claims Handler**.

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

Configuration Name	Default Value	Additional Information
LDAP Membership Attribute	`member`	Attribute used to designate the user's name as a member of the group in LDAP. Usually this is member or uniqueMember
LDAP Base Group DN	`ou=groups,dc=example,dc=com`	
User Attribute Map File	`etc/ws-security/attributeMap.properties`	Properties file that contains mappings from Claim=LDAP attribute.

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`ddf.security.encryption.EncryptionService`	optional	false

Exported Services

Registered Interface	Implementation Class	Properties Set
`org.apache.cxf.sts.claims.ClaimsHandler`	`ddf.security.sts.claimsHandler.LdapClaimsHandler`	Properties from the settings
`org.apache.cxf.sts.claims.claimsHandler`	`ddf.security.sts.claimsHandler.RoleClaimsHandler`	Properties from the settings

6.7.6. Security STS LDAP Login

The DDF 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.

Installing Security STS LDAP Claims Handler

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

Configuring

Configuration settings can be found in the Admin Console under **DDF Security Configuration Security STS LDAP Login**.

Configuration Name	Default Value	Additional Information
LDAP URL	`ldaps://${org.codice.ddf.system.hostname}:1636`	

Configuration Name	Default Value	Additional Information
StartTLS	`false`	Ignored if the URL uses ldaps.
LDAP Bind User DN	`cn=admin`	This user should have the ability to verify passwords and read attributes for any user.
LDAP Bind User Password	`secret`	This password value is encrypted by default using the Security Encryption application.
LDAP Username Attribute	`uid`	
LDAP Base User DN	`ou=users,dc=example,dc=com`	
LDAP Base Group DN	`ou=groups,dc=example,dc=com`	

Implementation Details

Imported Services

None

Exported Services

None

6.8. Security STS Service

The DDF 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.

Installing Security STS Realm

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

Configuring

None

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`ddf.security.encryption.EncryptionService`	optional	false

Exported Services

Registered Interfaces	Implementation Class	Properties Set
`org.apache.shiro.realm.Realm`	`ddf.security.realm.sts.StsRealm`	None

6.8.4. Security STS Server

The DDF 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/wsdl/ws-trust-1.4-service.wsdl` within the source code.

6.8.5. Installing Security STS Server

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

6.8.6. Configuring

Settings

Configuration settings can be found in the Admin Console under **Configuration** **Security STS Server**.

Configuration Name	Default Value	Additional Information
SAML Assertion Lifetime	1800	
Token Issuer	`localhost`	The name of the server issuing tokens. Generally this is the cn or hostname of this machine on the network.
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.

6.8.7. Implementation Details

Imported Services

Registered Interface	Availability	Multiple
`org.apache.cxf.sts.claims.ClaimsHandler`	optional	true
`org.apache.cxf.sts.token.validator.TokenValidator`	optional	true

Exported Services

None

6.9. Security PDP

The DDF Security Policy Decision Point (PDP) module contains services that are able to perform authorization decisions based on configurations and policies. In the DDF Security Framework, these components are called realms, and they implement the `org.apache.shiro.realm.Realm` and `org.apache.shiro.authz.Authorizer` interfaces. Although these components perform decisions on access control, enforcement of this decision is performed by components within the notional PEP application.

6.9.1. Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
`security-pdp-authzrealm`	`security-pdp-authz`	Security PDP Java Realm

6.10. Security PDP AuthZ Realm

The DDF Security PDP AuthZ Realm exposes a realm service that makes decisions on authorization requests using the attributes stored within the metacard to determine if access should be granted. This realm can use XACML and will delegate decisions to an external processing engine if internal processing fails. Decisions are first made based on the "match-all" and "match-one" logic. Any attributes listed in the "match-all" or "match-one" sections will not be passed to the XACML processing engine and they will be matched internally. It is recommended to list as many attributes as possible in these sections to avoid going out to the XACML processing engine for performance reasons. If it is desired that all decisions be passed to the XACML processing engine, remove all of the "match-all" and "match-one" configurations. The configuration below provides the mapping between user attributes and the attributes being asserted - one map exists for each type of mapping (each map may contain multiple values).

Match-All Mapping

This mapping is used to guarantee that all values present in the specified metacard attribute exist in the corresponding user attribute.

Match-One Mapping

This mapping is used to guarantee that at least one of the values present in the specified metacard attribute exists in the corresponding user attribute.

- Match-One Mapping: This mapping is used to guarantee that at least one of the values present in the specified metacard attribute exists in the corresponding user attribute.

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

This bundle is installed by default and can be added by installing the `security-pdp-authz` feature if it was uninstalled previously.

6.10.1. Configuring

Settings can be found in the Admin Console (https://localhost:8993/admin) under **DDF Security Configuration Security AuthZ Realm**.

Configuration Name	Default Value	Additional Description
Match-All Mappings		These map user attributes to metacard security attributes to be used in "Match All" checking. All the values in the metacard attribute must be present in the user attributes in order to "pass" and allow access. These attribute names are case-sensitive.
Match-One Mappings		These map user attributes to metacard security attributes to be used in "Match One" checking. At least one of the values from the metacard attribute must be present in the corresponding user attribute to "pass" and allow access. These attribute names are case-sensitive.

Implementation Details

Imported Services

None

Exported Services

Registered Interfaces	Implementation Class	Properties Set

`org.apache.shiro.realm.Realm`
 	
`org.apache.shiro.authz.Authorizer`
	`ddf.security.pdp.realm.AuthzRealm`	None

6.10.2. Guest Interceptor

The goal of the `GuestInterceptor` is to allow non-secure clients (SOAP requests without security headers) to access secure service endpoints.

All requests to secure endpoints must include, as part of the incoming message, a user's credentials in the form of a SAML assertion or a reference to a SAML assertion. For REST/HTTP requests, either the assertion itself or the session reference (that contains the assertion) is included. For SOAP requests, the assertion is included in the SOAP header.

Rather than reject requests without user credentials, the guest interceptor detects the missing credentials and inserts an assertion that represents the "guest" user. The attributes included in this guest user assertion are configured by the administrator to represent any unknown user on the current network.

Installing Guest Interceptor

The `GuestInterceptor` is installed by default with DDF Security Application.

Configuring Guest Interceptor

6.10.3. Configuring via the Admin Console

1. Navigate to the Admin Console at https://localhost:8993/admin
2. Click the DDF Security application tile
3. Click the **Configuration** tab
4. Click the **Security STS Guest Claims Handler** configuration
5. Click the + next to Attributes to add a new attribute
6. Add any addtional attributes that you want every user to have
7. Click **Save changes**

Once these configurations have been added, the GuestInterceptor is ready for use. Both secure and non-secure requests will be accepted by all secure DDF service endpoints.

6.11. 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).

6.11.1. Components

Bundle Name	Located in Feature	Description
`security-idp-sp`	`security-idp`	IdP Service Provider
`security-idp-server`	`security-idp`	IdP Server

6.11.2. Installing Security IdP

These bundles are not installed by default but can be started by installing the `security-idp` feature.

6.11.3. Security IdP Service Provider

The IdP client that interacts with the specified Identity Provider.

6.11.4. Security IdP Server

An internal Identity Provider solution.

6.11.5. 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.

7. Integrating DDF Solr

Version: 2.9.1

This section supports integration of this application with external frameworks.

Some notable features of the Solr Catalog Provider include:

- Supports extensible metacards
- Fast, simple contextual searching
- Indexes XML Attributes as well as CDATA sections and XML text elements
- Full XPath support.
- Works with an embedded, local Solr Server (all-in-one Catalog)
  - No configuration necessary on a single-node distribution
  - Data directory of Solr indexes are configurable
- Works with a standalone Solr Server

7.1. Embedded Solr Catalog Provider Pros and Cons

Feature	Pro	Con
Scalability		* Does not scale. Only runs one single server instance. * Does not allow the middle tier to be scaled.
Flexibility	* Can be embedded in Java easily. * Requires no HTTP connection. * Uses the same interface as the Standalone Solr Server uses under the covers. * Allows for full control over the Solr Server. No synchronous issues on startup; i.e., the Solr Server will synchronously start up with the Solr Catalog Provider * Runs within the same JVM * Setup and Installation is simple: unzip and run.	* Can only be interfaced using Java

Feature	Pro	Con
(Administrative)Tools	* External open source tools like Luke will work to allow admins to check index contents * JMX metrics reporting is enabled	* No Web Console. * No easy way to natively access (out of the box) what is in the index files or health of server at the data store level.
Security	* Does not open any ports which means no ports have to be secured.	
Performance	* Requires no HTTP or network overhead * Near real-time indexing * Can understand complex queries	
Backup/Recovery	* Can manually or through custom scripts back up the indexes	* Must copy files when server is shutdown

7.1.1. When to Use

Use the local, embedded Solr Catalog Provider when only one DDF instance is necessary and scalability is not an issue. The local, embedded Solr Catalog Provider requires no installation and little to no configuration. It is great for demonstrations, training exercises, or for sparse querying and ingesting.

7.2. Solr Catalog External Provider Pros and Cons

Feature	Pro	Con
Scalability	* Allows the middle-tier to be scaled by pointing various middle-tier instances to one server facade. * Possible data tier scalability with Solr Cloud. Solr Cloud allows for "high scale, fault tolerant, distributed indexing and search capabilities."	* Solr Cloud Catalog Provider not implemented yet.
Flexibility	* REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. * Ability to run in separate or same JVM of middle tier.	

Feature	Pro	Con
(Administrative) Tools	* Contains Solr Admin GUI, which allows admins to query, check health, see metrics, see configuration files and preferences, etc. * External open source tools like Luke will work to allow admins to check index contents. * JMX metrics reporting is enabled.	
Security	* Inherits app server security.	* Web Console must be secured and is openly accessible. * REST-like HTTP/XML and JSON APIs must be secured. * Current Catalog Provider implementation requires sending unsecured messages to Solr. Without a coded solution, requires network or firewall restrictions in order to secure.
Performance	* If scaled, high performance. * Near real-time indexing.	* Possible network latency impact * Extra overhead when sent over HTTP. Extra parsing for XML, JSON, or other interface formats. * Possible limitations upon requests and queries dependent on HTTP server settings.
Backup/Recovery	* Built-in recovery tools that allow in-place backups (does not require server shutdown). * Backup of Solr indexes can be scripted.	* Recovery is performed as an HTTP request.

7.2.1. When to Use

Use the Solr External Provider when the Standalone Solr Server is being used on a separate machine. Refer to the Standalone Solr Server recommended configuration.

7.2.2. Implementation Details

Indexing Text

When storing fields, the Solr Catalog Provider will analyze and tokenize the text values of `STRING_TYPE` and `XML_TYPE AttributeTypes`. These types of fields are indexed in at least three ways: in raw form, analyzed with case sensitivity, and analyzed without concern to case sensitivity. Concerning XML, the Solr Catalog Provider will analyze and tokenize XML CDATA sections, XML element text values, and XML attribute values.

8. Integrating DDF Spatial

Version: 2.9.1

The DDF Spatial Application provides KML transformer and a KML network link endpoint that allows a user to generate a View-based KML Query Results Network Link.

This guide supports integration of this application with external frameworks.

8.1. Integrating DDF with CSW

Catalog Services for Web (*CSW*) is an Open Geospatial Consortium (OGC) standard.

8.1.1. CSW v2.0.2 Endpoint

The CSW endpoint provides an XML-RPC endpoint that a client accesses to search collections of descriptive information (metadata) about geospatial data and services. The CSW endpoint implements version 2.0.2 of the CSW specification. The CSW endpoint also supports the Application Profile based on ISO 19115/ISO19119

Using the CSW Endpoint

Once installed, the CSW endpoint is accessible from http://<DDF_HOST>:<DDF_PORT>/services/csw.

`GetCapabilities` Operation

The `GetCapabilites` 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 Encoding*

```
http://<DDF_HOST>:<DDF_PORT>/services/csw?service=CSW&version=2.0.2&request=GetCapabiliti
es
```

`GetCapabilities HTTP POST`

The `HTTP POST` form of `GetCapabilities` operates on the root CSW endpoint URL (http://<DDF_HOST>:<DDF_PORT>/services/csw) with an XML message body that is defined by the

`GetCapabilities` element of the http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd [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` **Response**

The following is an example of an `application/xml` response to the `GetCapabilities` operation:

Sample Response

```
<?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://localhost:8993/services/csw"/>
                    <ows:Post ns2:href="https://localhost:8993/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://localhost:8993/services/csw"/>
                    <ows:Post ns2:href="https://localhost:8993/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
```

```
                </ows:Post>
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="typeName">
            <ows:Value>csw:Record</ows:Value>
            <ows:Value>gmd:MD_Metadata</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="OutputFormat">
            <ows:Value>application/xml</ows:Value>
            <ows:Value>application/json</ows:Value>
            <ows:Value>application/atom+xml</ows:Value>
            <ows:Value>text/xml</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="schemaLanguage">
            <ows:Value>http://www.w3.org/XMLSchema</ows:Value>
            <ows:Value>http://www.w3.org/XML/Schema</ows:Value>
            <ows:Value>http://www.w3.org/2001/XMLSchema</ows:Value>
            <ows:Value>http://www.w3.org/TR/xmlschema-1/</ows:Value>
        </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetRecords">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get ns2:href="https://localhost:8993/services/csw"/>
                <ows:Post ns2:href="https://localhost:8993/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://localhost:8993/services/csw"/>
                    <ows:Post ns2:href="https://localhost:8993/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://localhost:8993/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="typeNames">
                <ows:Value>xml</ows:Value>
                <ows:Value>appxml</ows:Value>
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
                <ows:Value>tika</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ConstraintLanguage">
                <ows:Value>Filter</ows:Value>
                <ows:Value>CQL_Text</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Parameter name="service">
            <ows:Value>CSW</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="version">
            <ows:Value>2.0.2</ows:Value>
        </ows:Parameter>
    </ows:OperationsMetadata>
    <ogc:Filter_Capabilities>
        <ogc:Spatial_Capabilities>
            <ogc:GeometryOperands>
                <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
                <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
                <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
            </ogc:GeometryOperands>
            <ogc:SpatialOperators>
                <ogc:SpatialOperator name="BBOX"/>
                <ogc:SpatialOperator name="Beyond"/>
                <ogc:SpatialOperator name="Contains"/>
                <ogc:SpatialOperator name="Crosses"/>
                <ogc:SpatialOperator name="Disjoint"/>
                <ogc:SpatialOperator name="DWithin"/>
                <ogc:SpatialOperator name="Intersects"/>
                <ogc:SpatialOperator name="Overlaps"/>
                <ogc:SpatialOperator name="Touches"/>
                <ogc:SpatialOperator name="Within"/>
            </ogc:SpatialOperators>
```

```
        </ogc:Spatial_Capabilities>
        <ogc:Scalar_Capabilities>
            <ogc:LogicalOperators/>
            <ogc:ComparisonOperators>
                <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
            </ogc:ComparisonOperators>
        </ogc:Scalar_Capabilities>
        <ogc:Id_Capabilities>
            <ogc:EID/>
        </ogc:Id_Capabilities>
    </ogc:Filter_Capabilities>
</csw:Capabilities>
```

`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 Encoding*

```
http://<DDF_HOST>:<DDF_PORT>/services/csw?service=CSW&version=2.0.2&request=DescribeRecor
d&NAMESPACE=xmlns(http://www.opengis.net/cat/csw/2.0.2)&outputFormat=application/xml&sche
maLanguage=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` **Response**

The following is an example of an application/xml response to the `DescribeRecord` operation.

DescribeRecord Response

```
<?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:id
entifier>

                </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_ResolutionType.
        </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=
"gmd:CI_Citation_PropertyType"/>
                            <xs:element name="abstract" type=
"gco:CharacterString_PropertyType"/>
                            <xs:element minOccurs="0" name="purpose" type=
"gco:CharacterString_PropertyType"/>
                            <xs:element maxOccurs="unbounded" minOccurs="0" name="credit"
type="gco:CharacterString_PropertyType"/>
                            <xs:element maxOccurs="unbounded" minOccurs="0" name="status"
type="gmd:MD_ProgressCode_PropertyType"/>
                            <xs:element maxOccurs="unbounded" minOccurs="0" name=
"pointOfContact" type="gmd:CI_ResponsibleParty_PropertyType"/>
                            <xs:element maxOccurs="unbounded" minOccurs="0" name=
"resourceMaintenance" type="gmd:MD_MaintenanceInformation_PropertyType"/>
                            <xs:element maxOccurs="unbounded" minOccurs="0" name=
"graphicOverview" type="gmd: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: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="intelligenceMilitary"/>
                    <xs:enumeration value="inlandWaters"/>
                    <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` Response

The following is an example of an application/xml response to the `DescribeRecord` operation for a csw:Record.

DescribeRecord Response

```
<?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:id
entifier>

                </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 req`uestId (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

This is 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).

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.

`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 Encoding*

```
http://<DDF_HOST>:<DDF_PORT>/services/csw?service=CSW&version=2.0.2&request=GetRecords&o
utputFormat=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 specifc source.

NOTE The `DistributedSearch` element must be specific with a `hopCount` greater than 1 to identify the is 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:PropertyIsEqualToLike wildCard="*" singleChar="#" escapeChar="!">
                  <ns2:PropertyName>source-id</ns2:PropertyName>
                  <ns2:Literal>Source1</ns2:Literal>
                </ns2:PropertyIsLike>
                <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` Response

The following is an example of an `application/xml` response to the `GetRecords` operation.

GetRecords *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="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>
        <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:SummaryRecord>
        <dc:identifier>c607440db9b0407e92000d9260d35444</dc:identifier>
        <dc:title>Product03</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>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>
        <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>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>
        <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>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` Response

The following is an example of an `application/xml` response to the `GetRecords` operation with the GMD Output Schema.

GetRecords *XML Response*

```
<?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:CharacterStrin
g>
            </fileIdentifier>
            <hierarchyLevel>
                <MD_ScopeCode codeListValue="nitf" 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.ntf</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>
```

`GetRecordById` Operation

The `GetRecords` 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

This is 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). Additionally the output schema `http://www.iana.org/assignments/media-types/application/octet-stream` is recognized and used to retrieve a product.

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`

The following is an example of a `HTTP GET` request:

`GetRecords` *KVP Encoding*

```
http://<DDF_HOST>:<DDF_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=fd7ff1535dfe47db8793
b550d4170424,ba908634c0eb439b84b5d9c42af1f871
```

`GetRecordById` `HTTP POST`

The following is an example of a `HTTP POST` request:

`GetRecordById` *XML Request*

```
<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`

The following is an example of an `application/xml` response to the `GetRecordById` operation:

Sample - GetRecordByIdResponse

```
<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 13. 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			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

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dct:educationLevel				0-n
dct:extent				0-n
dct:hasFormat				0-n
dct:hasPart				0-n
dct:hasVersion				0-n
dct:isFormatOf				0-n
dct:isPartOf				0-n
dct:isReferencedBy				0-n
dct:isReplacedBy				0-n
dct:isRequiredBy				0-n
dct:issued	modified			0-n
dct:isVersionOf				0-n
dct:license				0-n
dct:mediator				0-n
dct:medium				0-n
dct:modified	modified		0-n	0-n
dct:provenance				0-n
dct:references				0-n
dct:replaces				0-n
dct:requires				0-n
dct:rightsHolder				0-n
dct:spatial	location		0-n	0-n
dct:tableOfContents				0-n

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

Transaction Operation

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 operations.

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>
```

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"`.

Sample XML Transaction Insert Response

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

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. A property is removed only 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 `<csw: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">
    <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`.

Sample XML Transaction Update Request with CQL filter constraint

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

Transaction Update Response

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

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>
```

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>
```

Transaction Delete Response

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

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>
```

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.

Subscription `GetRecords` `HTTP GET`

`GetRecords` *KVP Encoding*

```
http://<DDF_HOST>:<DDF_PORT>/services/csw/subscription?service=CSW&version=2.0.2&request=
GetRecords&o
utputFormat=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&ResponseHandler=https%3A%2F%2Fsome.ddf%2Fservices%2Fcsw%2Fsubscripti
on%2Fevent&
ConstraintLanguage=CQL_TEXT&constraint=Text Like '%25'
```

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

Subscription `GetRecords` `HTTP PUT`

The `HTTP PUT` request GetRecords is the exact same as the `POST` but is used to update an existing subscription but the requestid urn tacked on the end of the url.

Subscription `GetRecords` XML Request

```
http://<DDF_HOST>:<DDF_PORT>/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

Subscription `GetRecords` Response

The following is an example of an `application/xml` response to the `GetRecords` operation.

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/XMLSch
ema-instance"
                                                    xsi:schemaLocation="http://www.opengis.n
et/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</ResponseHandle
r>
        <Query typeNames="Record">
            <ElementSetName>summary</ElementSetName>
            <Constraint version="1.1.0">
                <ogc:Filter>
                    <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                        <ogc:PropertyName>xml</ogc:PropertyName>
                        <ogc:Literal>%</ogc:Literal>
                    </ogc:PropertyIsLike>
                </ogc:Filter>
            </Constraint>
        </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

Subscription `GetRecords` event 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>
```

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`

```
http://<DDF_HOST>:<DDF_PORT>/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

Subscription `HTTP GET`or `HTTP DELETE` Response

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

Subscription `HTTP GET` or `HTTP DELETE` 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/XMLSch
ema-instance"
                                                    xsi:schemaLocation="http://www.opengis.n
et/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</ResponseHandle
r>
        <Query typeNames="Record">
            <ElementSetName>summary</ElementSetName>
            <Constraint version="1.1.0">
                <ogc:Filter>
                    <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                        <ogc:PropertyName>xml</ogc:PropertyName>
                        <ogc:Literal>%</ogc:Literal>
                    </ogc:PropertyIsLike>
                </ogc:Filter>
            </Constraint>
        </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

8.1.2. Install and Uninstall

The CSW endpoint can be installed and uninstalled using the normal processes described in the Configuration section.

8.1.3. Configuration

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

8.1.4. Known Issues

None

8.2. CSW v2.0.2 Source

The CSW source supports the ability to search collections of descriptive information (metadata) for data, services, and related information objects.

8.2.1. Using

Use the CSW source if querying a CSW version 2.0.2 compliant service.

8.2.2. Installing and Uninstalling

The CSW source can be installed and uninstalled using the normal processes described in the Configuring DDF section.

8.2.3. Configuring

The configurable properties for the CSW source are accessed from the CSW Federated Source Configuration in the Admin Console.

Configure the CSW Source

Table 14. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	`id`	String	Unique Name of this Source.	CSW	Yes
CSW URL	`cswUrl`	String	URL to the Catalogue Services for the Web site that will be queried by this source		Yes
Event Service Address	`eventServiceAddress`	String	DDF Event Service endpoint.		No
Register for Events	`registerForEvents`	Boolean	Check to register for events from this source.	false	No

Title	Property	Type	Description	Default Value	Required
Username	`username`	String	Username to log into the CSW service		No
Password	`password`	String	Password to log into the CSW service		No
Disable CN Check	`disableCnCheck`	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	Yes
Force Longitude/Latitude coordinate order	`isLonLatOrder`	Boolean	Force Longitude/Latitude coordinate order	false	Yes
Use `posList` in `LinearRing`	`usePosList`	Boolean	Use a `<posList>` element rather that a series of `<pos>` elements when issuing geospatial queries containing a `LinearRing`	false	Yes
Metacard Mappings	`metacardMappings`	String	Mapping of the Metacad Attribute names to their CSW property names. The format should be 'title=dc:title'.	`effective=created,created=dateSubmitted,modified=modified,thumbnail=references,content-type=type,id=identifier,resource-uri=source`	No
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1)	5	Yes

Title	Property	Type	Description	Default Value	Required
Connection Timeout	`connectionTimeout`	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	30000	Yes
Receive Timeout	`receiveTimeout`	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	60000	Yes
Output schema	`outputSchema`	String	Output Schema	http://www.opengis.net/cat/csw/2.0.2	Yes
Query Type Name	`queryTypeName`	String	Qualified Name for the Query Type used in the CSW GetRecords request	csw:Record	Yes
Query Type Namespace	`queryTypeNamespace`	String	Namespace for the Query Type used in the CSW GetRecords request	http://www.opengis.net/cat/csw/2.0.2	Yes
Force CQL Text as the Query Language	`isCqlForced`	Boolean	Force CQL Text	false	Yes
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected None No Spatial Filter Type as the only available Spatial Filter.		

8.2.4. Known Issues

- The CSW Source does not support text path searches.
- All contextual searches are case sensitive; case-insensitive searches are not supported.
- Nearest neighbor spatial searches are not supported.
- Fuzzy contextual searches are not supported.

8.3. GMD CSW APISO v2.0.2 Source

The GMD CSW source supports the ability to search collections of descriptive information (metadata) for data, services, and related information objects, based on the Application Profile ISO 19115/ISO19119.

8.3.1. Using

Use the GMD CSW source if querying a GMD CSW APISO compliant service.

8.3.2. Installing and Uninstalling

The GMD CSW source can be installed and uninstalled using the normal processes described in the Configuring DDF section.

8.3.3. Configuring

The configurable properties for the GMD CSW source are accessed from the GMD CSW ISO Federated Source Configuration in the Admin Console.

Configure the CSW Source

Table 15. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	`id`	String	Unique Name of this Source.	CSW	Yes
CSW URL	`cswUrl`	String	URL to the Catalogue Services for the Web site that will be queried by this source		Yes

Title	Property	Type	Description	Default Value	Required
Event Service Address	`eventServiceAddress`	String	DDF Event Service endpoint.		No
Register for Events	`registerForEvents`	Boolean	Check to register for events from this source.	false	No
Username	`username`	String	Username to log into the CSW service		No
Password	`password`	String	Password to log into the CSW service		No
Disable CN Check	`disableCnCheck`	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	Yes
Force Longitude/Latitude coordinate order	`isLonLatOrder`	Boolean	Force Longitude/Latitude coordinate order	false	Yes
Use `posList` in `LinearRing`	`usePosList`	Boolean	Use a `<posList>` element rather that a series of `<pos>` elements when issuing geospatial queries containing a `LinearRing`	false	Yes

Title	Property	Type	Description	Default Value	Required
Metacard Mappings	`metacardMappings`	String	Mapping of the Metacad Attribute names to their CSW property names. The format should be 'title=dc:title'.	`id=apiso:Identifier,effective=apiso:PublicationDate,created=apiso:CreationDate,modified=apiso:RevisionDate,title=apiso:AlternateTitle,AnyText=apiso:AnyText,ows:BoundingBox=apiso:BoundingBox`	No
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1)	5	Yes
Connection Timeout	`connectionTimeout`	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	30000	Yes
Receive Timeout	`receiveTimeout`	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	60000	Yes
Output schema	`outputSchema`	String	Output Schema	http://www.isotc211.org/2005/gmd	Yes

Title	Property	Type	Description	Default Value	Required
Query Type Name	`queryTypeName`	String	Qualified Name for the Query Type used in the CSW GetRecords request	gmd:MD_Metadata	Yes
Query Type Namespace	`queryTypeNamespace`	String	Namespace for the Query Type used in the CSW GetRecords request	http://www.isotc211.org/2005/gmd	Yes
Force CQL Text as the Query Language	`isCqlForced`	Boolean	Force CQL Text	false	Yes
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected None No Spatial Filter Type as the only available Spatial Filter.		

8.4. Integrating DDF with KML

Keyhole Markup Language (*KML*) is an XML notation for describing geographic annotation and visualization for 2- and 3- dimensional maps.

8.5. KML Network Link Endpoint

The KML Network Link endpoint allows a user to generate a view-based KML Query Results Network Link. This network link can be opened with Google Earth, establishing a dynamic connection between Google Earth and DDF. The root network link will create a network link for each configured source, including the local catalog. The individual source network links will perform a query against the OpenSearch Endpoint periodically based on the current view in the KML client. The query parameters for this query are obtained by a bounding box generated by Google Earth. The root network link will refresh every 12 hours or can be forced to refresh. As a user changes their current view, the query will be re-executed with the bounding box of the new view. (This query gets re-executed two seconds after the user stops moving the view.)

8.5.1. Using

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

```
http://<DDF_HOST>:<DDF_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

```
<?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:xsdschema: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>
```

When configured to do so, the KML endpoint can serve up a KML style document. The request below will return the configured KML style document.

```
http://<DDF_HOST>:<DDF_PORT>/services/catalog/kml/style
```

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.

```
http://<DDF_HOST>:<DDF_PORT>/services/catalog/kml/icons?<icon-name>
#NOTE: <icon-name> must be the name of an icon contained in the directory being served
up like:
http://<DDF_HOST>:<DDF_PORT>/services/catalog/kml/icons?sample-icon.png
```

8.5.2. Installing and Uninstalling

The `spatial-kml-networklinkendpoint` feature is installed by default with the Spatial App.

8.5.3. Configuring

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.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Style Document	`styleUrl`	String	KML document containing custom styling. This will be served up by the KmlEndpoint (e.g., file:///path/to/kml/style/doc.kml).		No
Icons Location	`iconLoc`	String	Location of icons for `KmlEndpoint`.		No
Description	`description`	String	Description of this `NetworkLink`. Enter a short description of what this `NetworkLink` provides.		No

8.5.4. Known Issues

None.

8.6. KML Query Response Transformer

The KML Query Response Transformer is responsible for translating a query response into a KML-formatted document. The KML will contain an HTML description for each metacard that will display in the pop-up bubble in Google Earth. The HTML contains links to the full metadata view as well as the product.

8.6.1. Using

Using the OpenSearch Endpoint, for example, query with the format option set to the KML shortname: `kml`.

```
http://localhost:8181/services/catalog/query?q=schematypesearch&format=kml
```

Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns:ns2="http://www.google.com/kml/ext/2.2" xmlns="http://www.opengis.net/kml/2.2"
xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:ns3=
"http://www.w3.org/2005/Atom">
  <Document id="f0884d8c-cf9b-44a1-bb5a-d3c6fb9a96b6">
    <name>Results (1)</name>
    <open xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">false</open>
    <Style id="bluenormal">
      <LabelStyle>
        <scale>0.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>33ff0000</color>
        <width>3.0</width>
      </LineStyle>
      <PolyStyle>
        <color>33ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
        <text><h3><b>$[name]</b></h3><table><tr><td width="400">$[description]</td>
</tr></table></text>
      </BalloonStyle>
    </Style>
    <Style id="bluehighlight">
      <LabelStyle>
        <scale>1.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>99ff0000</color>
        <width>6.0</width>
      </LineStyle>
      <PolyStyle>
        <color>99ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
        <text><h3><b>$[name]</b></h3><table><tr><td width="400">$[description]</td>
</tr></table></text>
      </BalloonStyle>
    </Style>
    <StyleMap id="default">
      <Pair>
```

```
        <key>normal</key>
        <styleUrl>#bluenormal</styleUrl>
      </Pair>
      <Pair>
        <key>highlight</key>
        <styleUrl>#bluehighlight</styleUrl>
      </Pair>
    </StyleMap>
    <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
      <name>MultiPolygon</name>
      <description><!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=windows-1252" http-equiv="content-type">
    <style media="screen" type="text/css">
      .label {
        font-weight: bold
      }
      .linkTable {
width: 100% }
      .thumbnailDiv {
        text-align: center
} img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    </style>
  </head>
  <body>
        <div class="thumbnailDiv"><a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f
48fab939da528e?transform=resource"><img alt="Thumnail"
src=" CA=="></a></div>
    <table>
      <tr>
        <td class="label">Source:</td>
        <td>ddf.distribution</td>
      </tr>
      <tr>
        <td class="label">Created:</td>
        <td>Wed Oct 30 09:46:29 MDT 2013</td>
      </tr>
      <tr>
              <td class="label">Effective:</td>
        <td>2014-01-07T14:48:47-0700</td>
      </tr>
    </table>
```

```
    <table class="linkTable">
      <tr>
        <td><a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f
48fab939da528e?transform=html">View Details...</a></td>
        <td><a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f
48fab939da528e?transform=resource">Download...</a></td>
      </tr>
    </table>
  </body>
</html>
</description>
      <TimeSpan>
        <begin>2014-01-07T21:48:47</begin>
      </TimeSpan>
      <styleUrl>#default</styleUrl>
      <MultiGeometry>
        <Point>
          <coordinates>102.0,2.0</coordinates>
        </Point>
        <MultiGeometry>
          <Polygon>
            <outerBoundaryIs>
              <LinearRing>
                <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0
102.0,2.0</coordinates>
              </LinearRing>
100.8,0.2
  </outerBoundaryIs>
</Polygon>
<Polygon>
  <outerBoundaryIs>
    <LinearRing>
      <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2
        100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
      </MultiGeometry>
    </MultiGeometry>
  </Placemark>
</Document>
</kml>
```

8.6.2. Installing and Uninstalling

The `spatial-kml-transformer` feature is installed by default with the Spatial App.

```
http://localhost:8181/services/catalog/0103c77e66d9428d8f48fab939da528e?transform=kml
```

8.6.3. Configuring

None.

8.6.4. Implementation Details

Transformer Shortname	MIME Type
`kml`	`application/vnd.google-earth.kml+xml`

8.6.5. Known Issues

None.

8.7. KML Metacard Transformer

The KML Metacard Transformer is responsible for translating a metacard into a KML-formatted document. The KML will contain an HTML description that will display in the pop-up bubble in Google Earth. The HTML contains links to the full metadata view as well as the product.

8.7.1. Using

Using the REST Endpoint for example, request a metacard with the transform option set to the KML shortname.

Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns:ns2="http://www.google.com/kml/ext/2.2" xmlns="http://www.opengis.net/kml/2.2"
xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:ns3=
"http://www.w3.org/2005/Atom">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description><!DOCTYPE html>
    <html>
      <head>
        <meta content="text/html; charset=windows-1252" http-equiv="content-type">
        <style media="screen" type="text/css">
          .label {
            font-weight: bold
          }
          .linkTable {
width: 100% }
          .thumbnailDiv {
            text-align: center
          }
          img {
            max-width: 100px;
               max-height: 100px;
            border-style:none
          }
    </style>
  </head>
  <body>
        <div class="thumbnailDiv"><a href=
"http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab9
39da528e?transform=resource"><img alt="Thumnail" src=" CA=="></a></div>
    <table>
      <tr>
        <td class="label">Source:</td>
        <td>ddf.distribution</td>
      </tr>
      <tr>
        <td class="label">Created:</td>
        <td>Wed Oct 30 09:46:29 MDT 2013</td>
      </tr>
      <tr>
        <td class="label">Effective:</td>
        <td>2014-01-07T14:58:16-0700</td>
      </tr>
    </table>
    <table class="linkTable">
```

```
      <tr>
        <td><a href=
"http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab9
39da528e?transform=html">View Details...</a></td>
        <td><a href=
"http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab9
39da528e?transform=resource">Download...</a></td>
      </tr>
    </table>
  </body>
</html>
</description>
    <TimeSpan>
      <begin>2014-01-07T21:58:16</begin>
    </TimeSpan>
    <Style id="bluenormal">
      <LabelStyle>
        <scale>0.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>33ff0000</color>
        <width>3.0</width>
      </LineStyle>
      <PolyStyle>
        <color>33ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
<text><h3><b>$[name]</b></h3><table><tr><td
width="400">$[description]</td></tr></table></text>
      </BalloonStyle>
    </Style>
    <Style id="bluehighlight">
      <LabelStyle>
        <scale>1.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>99ff0000</color>
        <width>6.0</width>
      </LineStyle>
      <PolyStyle>
        <color>99ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
        <text><h3><b>$[name]</b></h3><table><tr><td width="400">$[description]</td>
```

```
</tr></table></text>
      </BalloonStyle>
    </Style>
    <StyleMap id="default">
      <Pair>
        <key>normal</key>
        <styleUrl>#bluenormal</styleUrl>
      </Pair>
      <Pair>
        <key>highlight</key>
        <styleUrl>#bluehighlight</styleUrl>
      </Pair>
    </StyleMap>
    <MultiGeometry>
      <Point>
        <coordinates>102.0,2.0</coordinates>
      </Point>
      <MultiGeometry>
        <Polygon>
          <outerBoundaryIs>
            <LinearRing>
              <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0 102.0,2.0</
coordinates>
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
        <Polygon>
100.8,0.2
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2 100.8
,0.8 100.2,0.8 100.2,0.2</coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</MultiGeometry>
</Placemark>
</kml>
```

8.7.2. Installing and Uninstalling

The `spatial-kml-transformer` feature is installed by default with the Spatial App.

8.7.3. Configuring

None.

8.7.4. Implementation Details

Transformer Shortname	MIME Type
kml	application/vnd.google-earth.kml+xml

8.7.5. Known Issues

None.

8.8. KML Style Mapper

The KML Style Mapper provides the ability for the `KmlTransformer` to map a KML Style URL to a metacard based on that metacard's attributes. For example, if a user wanted all JPEGs to be blue, the KML Style Mapper provides the ability to do so. This would also allow an administrator to configure metacards from each source to be different colors.

The configured style URLs are expected to be HTTP URLs. For more information on style URL's, refer to the KML Reference.

The KML Style Mapper supports all basic and extended metacard attributes. When a style mapping is configured, the resulting transformed KML contain a `<styleUrl>` tag pointing to that style, rather than the default KML style supplied by the `KmlTransformer`.

8.8.1. Configuring

The properties below describe how to configure a Style Mapping. The configuration name is `Spatial KML Style Map Entry`.

Table 16. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Attribute Name	`attributeName`	The name of the metacard attribute to match against (e.g., `title`, `metadata-content-type`).	String		Yes
Attribute Value	`attributeValue`	String	The value of the metacard attribute.		Yes

Title	Property	Type	Description	Default Value	Required
Style URL	`styleUrl`	String	The full qualified URL to the KML style (e.g., http://example.com/styles#myStyle).		Yes

8.8.2. Example Values

```
xmlns="http://www.opengis.net/kml/2.2"
  xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xmlns:ns3="http://www.w3.org/2005/Atom">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description><!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=windows-1252" http-equiv="content-type">
    <style media="screen" type="text/css">
      .label {
        font-weight: bold
      }
      .linkTable {
width: 100% }
      .thumbnailDiv {
        text-align: center
} img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    </style>
  </head>
  <body>
        <div class="thumbnailDiv"><a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f4
8fab939da528e?transform=resource"><img alt="Thumnail"
src=" CA=="></a></div>
    <table>
      <tr>
        <td class="label">Source:</td>
        <td>ddf.distribution</td>
      </tr>
      <tr>
        <td class="label">Created:</td>
        <td>Wed Oct 30 09:46:29 MDT 2013</td>
      </tr>
      <tr>
        <td class="label">Effective:</td>
        <td>2014-01-07T14:58:16-0700</td>
      </tr>
    </table>
    <table class="linkTable">
      <tr>
        <td><a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f4
```

```
8fab939da528e?transform=html">View Details...</a></td>
        <td><a href=
"http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab9
39da528e?transform=resource">Download...</a></td>
      </tr>
    </table>
  </body>
</html>
</description>
    <TimeSpan>
      <begin>2014-01-07T21:58:16</begin>
    </TimeSpan>
 <styleUrl>http://example.com/kml/style#sampleStyle</styleUrl>
    <MultiGeometry>
      <Point>
        <coordinates>102.0,2.0</coordinates>
      </Point>
      <MultiGeometry>
        <Polygon>
          <outerBoundaryIs>
            <LinearRing>
              <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0
102.0,2.0</coordinates>
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
        <Polygon>
100.8,0.2
<outerBoundaryIs>
  <LinearRing>
    <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2
      100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
  </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</MultiGeometry>
</MultiGeometry>
</Placemark>
</kml>
```

8.8.3. Installing and Uninstalling

The KML Style Mapper is included in the `spatial-kml-transformer` feature and is installed by default with the Spatial App.

8.8.4. Implementation Details

Transformer Shortname	MIME Type
`kml`	`application/vnd.google-earth.kml+xml`

8.8.5. Known Issues

None.

8.9. Integrating DDF with WFS

The Web Feature Service (WFS) is an Open Geospatial Consortium (OGC) Specification. DDF supports the ability to integrate WFS 1.0 and WFS 2.0 Web Services.

NOTE DDF does not include a supported WFS Web Service (Endpoint) implementation. Therefore, federation for 2 DDF instances is not possible via WFS.

8.10. Working with WFS Sources

A Web Feature Service (WFS) source is an implementation of the FederatedSource interface provided by the DDF Framework. A WFS source provides capabilities for querying an Open Geospatial Consortium (OGC) WFS 1.0.0-compliant server. The results are made available to DDF clients.

8.10.1. WFS Features

When a query is issued to a WFS server, the output of the query is an XML document that contains a collection of feature member elements. Each WFS server can have one or more feature types with each type being defined by a schema that extends the WFS featureMember schema. The schema for each type can be discovered by issuing a `DescribeFeatureType` request to the WFS server for the feature type in question. The WFS source handles WFS capability discovery and requests for feature type description when an instance of the WFS source is configured and created.

See the WFS v1.0.0 Source for more information about how to configure a WFS source.

Convert a WFS Feature

In order to expose WFS features to DDF clients, the WFS feature must be converted into the common data format of the DDF, a metacard. The OGC package contains a `GenericFeatureConverter` that attempts to populate mandatory metacard fields with properties from the WFS feature XML. All properties will be mapped directly to new attributes in the metacard. However, the `GenericFeatureConverter` may not be able to populate the default metacard fields with properties from the feature XML.

Create a Custom Converter

To more accurately map WFS feature properties to fields in the metacard, a custom converter can be

created. The OGC package contains an interface, `FeatureConverter`, which extends the [] http://xstream.codehaus.org/javadoc/com/thoughtworks/xstream/converters/Converter.htmlConverter interface provided by the XStream project. XStream is an open source API for serializing XML into Java objects and vice-versa. Additionally, a base class, `AbstractFeatureConverter`, has been created to handle the mapping of many fields to reduce code duplication in the custom converter classes.

- Create the `CustomConverter` class extending the `ogc.catalog.common.converter.AbstractFeatureConverter` class.

```
public class CustomConverter extends ogc.catalog.common.converter
.AbstractFeatureConverter
```

- Implement the `FeatureConverterFactory` interface and the `createConverter()` method for the `CustomConverter`.

```
public class CustomConverterFactory implements FeatureConverterFactory {
 private final featureType;
 public CustomConverterFactory(String featureType) {
  this.featureType = featureType;
 }
 public FeatureConverter createConverter() {
  return new CustomConverter();
 }
 public String getFeatureType() {
  return featureType;
} }
```

- Implement the `unmarshal` method required by the `FeatureConverter` interface. The `createMetacardFromFeature(reader, metacardType)` method implemented in the `AbstractFeatureConverter` is recommended.

```
public Metacard unmarshal(HierarchicalStreamReader reader, UnmarshallingContext ctx) {
  MetacardImpl mc = createMetacardFromFeature(reader, metacardType);
  //set your feature specific fields on the metacard object here
  //
  //if you want to map a property called "beginningDate" to the Metacard.createdDate
field
  //you would do:
  mc.setCreatedDate(mc.getAttribute("beginningDate").getValue());
}
```

- Export the `ConverterFactory` to the OSGi registry by creating a blueprint.xml file for its bundle. The bean id and argument value must match the WFS Feature type being converted.

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:cm=
"http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.1.0">
  <bean id="custom_type" class="com.example.converter.factory.CustomConverterFactory">
    <argument value="custom_type"/>
  </bean>
  <service ref="custom_type" interface=
"ogc.catalog.common.converter.factory.FeatureConverterFactory"/>
</blueprint>
```

8.11. WFS v1.0.0 Source

The WFS Source allows for requests for geographical features across the web using platform-independent calls.

8.11.1. Using

Use the WFS Source if querying a WFS version 1.0.0 compliant service.

8.11.2. Installing and Uninstalling

The WFS Source can be installed and uninstalled using the normal processes described in the Configuring DDF section.

8.11.3. Configuring

The configurable properties for the WFS Source are accessed from the WFS Federated Source Configuration in the Admin Console.

Configuring WFS Source

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	`id`	String	Unique name of the Source.	WFS_v1_0_0	Yes
WFS URL	`wfsUrl`	String	URL to the Web Feature Service (WFS) that will be queried by this source (see below).		Yes

Title	Property	Type	Description	Default Value	Required
Disable CN Check	`disableCnCheck`	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	Yes
Username	`username`	String	Username to log in to the WFS service. Password to log in to the WFS service.		No
Password	`password`	String	Password to log in to the WFS service.		No
Forced Feature Type	`forcedFeatureType`	String	Force only a specific FeatureType to be queried instead of all featureTypes		No
Non Queryable Properties	`nonQueryableProperties`	List of Strings	Multivalued list of properties in the feature XML that should not be used as filters.		No
Poll Interval	`pollInterval`	Integer	Poll interval to check if the source is available (in minutes; minimum = 1).	5	Yes
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force the selected Spatial Filter Type to be the only available Spatial Filter.	None	No

Title	Property	Type	Description	Default Value	Required
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds	30000	Yes
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	Yes

8.11.4. WFS URL

The WFS URL must match the endpoint for the service being used. The type of service and version are added automatically, so they do not need to be included. Some servers will throw an exception if they are included twice, so do not include those.

The syntax depends on the server. However, in most cases, the syntax will be everything before the `?` character in the URL that corresponds to the `GetCapabilities` query.

Example GeoServer 2.5 Syntax

```
http://www.example.org:8080/geoserver/ows?service=wfs&version=1.0.0&request=GetCapabilities
```

In this case, the WFS URL would be

```
http://www.example.org:8080/geoserver/ows
```

8.11.5. Known Issues

None.

8.12. WFS v2.0.0 Source

The WFS 2.0 Source allows for requests for geographical features across the web using platform-independent calls.

8.12.1. Using

Use the WFS Source if querying a WFS version 2.0.0 compliant service. Also see Working with WFS Sources.

8.12.2. Installing and Uninstalling

The WFS Source can be installed and uninstalled using the normal processes described in the Configuring DDF section.

8.12.3. Configuring

The configurable properties for the WFS 2.0.0 Source are accessed from the WFS 2.0.0 Federated Source Configuration in the Admin Console.

Configuring WFS 2.0.0 Source

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	`id`	String	Unique name of the source	WFS_v2_0_0	Yes
WFS URL	`wfsUrl`	String	URL to the endpoint implementing the Web Feature Service (WFS) 2.0.0 spec.		Yes
Disable CN Check	`disableCnCheck`	Boolean	Disable CN Check for the server certificate. This should only be used when testing.	false	Yes
Coordinate Order	`coordinateOrder`	String	Coordinate order that remote source expects and returns spatial data in.	Lat/Lon	Yes

Title	Property	Type	Description	Default Value	Required
Disable Sorting	`disableSorting`	Boolean	When selected, the system will not specify sort criteria with the query. This should only be used if the remote source is unable to handle sorting even when the capabilities states `ImplementsSorting` is supported.	false	Yes
Username	`username`	String	Username for the WFS service.		No
Password	`password`	String	Password for the WFS service.		No
Forced Feature Type	`forcedFeatureType`	String	Force only a specific FeatureType to be queried instead of all featureTypes		No
Non Queryable Properties	`nonQueryableProperties`	List of Strings	Properties listed here will NOT be queryable and any attempt to filter on these properties will result in an exception.		No
Poll Interval	`pollInterval`	Integer	Poll interval to check if the source is available (in minutes; minimum = 1).	5	Yes

Title	Property	Type	Description	Default Value	Required
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.		No
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds	30000	Yes
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	Yes

8.12.4. WFS URL

The WFS URL must match the endpoint for the service being used. The type of service and version is added automatically, so they do not need to be included. Some servers will throw an exception if they are included twice, so do not include those.

The syntax depends on the server. However, in most cases, the syntax will be everything before the `?` character in the URL that corresponds to the `GetCapabilities` query.

Example GeoServer 2.5 Syntax

```
http://www.example.org:8080/geoserver/ows?service=wfs&version=2.0.0&request=GetCapabilities
```

In this case, the WFS URL would be

```
http://www.example.org:8080/geoserver/ows
```

8.12.5. Known Issues

None.

8.12.6. Mapping WFS Feature Properties to Metacard Attributes

The WFS 2.0 Source allows for virtually any schema to be used to describe a feature. A feature is relatively equivalent to a metacard. The `MetacardMapper` was added to allow an administrator to configure which feature properties map to which metacard attributes.

8.12.7. Using

Use the WFS `MetacardMapper` to configure which feature properties map to which metacard attributes when querying a WFS version 2.0.0 compliant service. When feature collection responses are returned from WFS sources, a default mapping occurs which places the feature properties into metacard attributes, which are then presented to the user via DDF. There can be situations where this automatic mapping is not optimal for your solution. Custom mappings of feature property responses to metacard attributes can be achieved through the `MetacardMapper`. The `MetacardMapper` is set by creating a feature file configuration which specifies the appropriate mapping. The mappings are specific to a given feature type.

8.12.8. Installing and Uninstalling

The WFS `MetacardMapper` can be installed and uninstalled using the normal processes described in the Configuring DDF section.

8.12.9. Configuring

This component can be configured using the normal processes described in the Configuring DDF section.

The configurable properties for the WFS MetacardMapper are accessed from the Metacard to WFS Feature Map Configuration in the Admin Console.

Configuring WFS MetacardMapper

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Feature Type	`featureType`	String	Feature Type. Format is `{URI}local-name`		Yes

Title	Property	Type	Description	Default Value	Required
Metacard Attribute to WFS Feature Property Mapping	`metacardAttrToFeaturePropMap`	String	Metacard Attribute to WFS Feature Property Mapping. Format is `metacardAttribute=featureProperty`		Yes
Temporal Sort By Feature Property	`sortByTemporalFeatureProperty`	String	When Sorting Temporally, Sort By This Feature Property.		No
Relevance Sort By Feature Property	`sortByRelevanceFeatureProperty`	String	When Sorting By Distance, Sort By This Feature Property.		No
Distance Sort By Feature Property	`sortByDistanceFeatureProperty`	String	When Sorting By Relevance, Sort By This Feature Property.		No

Example Configuration

There are two ways to configure the `MetcardMapper`, one is to use the Configuration Admin available via the Admin Console. Additionally, a `feature.xml` file can be created and copied into the "deploy" directory. The following shows how to configure the `MetacardMapper` to be used with the sample data provided with GeoServer. This configuration shows a custom mapping for the feature type ‘states’. For the given type, we are taking the feature property ‘states.STATE_NAME’ and mapping it to the metacard attribute ‘title’. In this particular case, since we mapped the state name to title in the metacard, it will now be fully searchable. More mappings can be added to the `featurePropToMetacardAttrMap` line through the use of comma as a delimiter.

Example `MetacardMapper` *Configuration Within a* `feature.xml` *file:*

```
<feature name="geoserver-states" version="2.9.1"
    description="WFS Feature to Metacard mappings for GeoServer Example
{http://www.openplans.org/topp}states">
    <config name="org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetacardMapper-
geoserver.http://www.openplans.org/topp.states">
        featureType = {http://www.openplans.org/topp}states
        service.factoryPid = org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetacardMapper
        featurePropToMetacardAttrMap = states.STATE_NAME=title
    </config>
</feature>
```

Known Issues

None.

9. Integrating DDF Search UI

Version: 2.9.1

The DDF Search UI application allows a user to search for records in the local Catalog (provider) and federated sources. Results of the search are displayed on a globe, providing a visual representation of where the records were found.

This page supports integration of this application with external frameworks.

9.1. CometD

The Standard Search UI utilizes CometD to communicate with the DDF. This 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/`

9.1.1. Query

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

Table 17. Query Parameters

Parameter Name	Description	Required
`src`	Catalog Source	No
`cql`	CQL query	Yes
`sort`	Sort Type	No
`id`	Query ID (Should be a uuid), This determines the channel that the query results will be returned on.	Yes

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

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

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

9.2. Notifications

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.

9.2.1. Persistence

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.