

Distributed Data Framework Documentation

Version 2.10.3. Copyright (c) Codice Foundation

Table of Contents

1. License	1
Overview	2
2. About DDF	2
2.1. Applications	2
3. Documentation Guide	3
3.1. Documentation Conventions	3
3.2. Support	4
Core Concepts	5
4. Search	5
4.1. Search Types	5
5. Metadata	5
6. Ingest	6
6.1. Populating Metacards During Ingest	6
7. Content	6
8. Catalog Framework	6
9. Federation	7
10. Events and Subscriptions	7
11. Registry	7
11.1. Identity Node	8
11.2. Subscription	8
11.3. Publication	8
12. Endpoints	8
Managing	9
13. Installing	9
13.1. Installation Prerequisites	9
13.2. Installing With the DDF Distribution Zip	11
13.3. Initial Startup	15
14. Configuring	19
14.1. Hardening Checklist	21
14.2. Managing Keystores and Certificates	22
14.3. Configuring from the Admin Console	29
14.4. Configuring from the Command Console	50
14.5. Configuring from Configuration Files	53
14.6. Importing Configurations	66
14.7. Other Configurations	69
15. Quick Start Tutorial	71
15.1. Quick Install Prerequisites	71
15.2. Quick Install of DDF	73

15.3. Configuring (Quick Start)	74
15.4. Ingesting (Quick Start)	74
16. Running	75
16.1. Starting	75
16.2. Maintaining	80
16.3. Monitoring	93
16.4. Troubleshooting	97
17. Data Management	98
17.1. File Formats Supported	98
17.2. Ingesting Data	100
17.3. Removing Expired Records from Catalog	102
Using	104
18. Using the Landing Page	105
18.1. Search DDF Button	105
18.2. Data Source Availability	105
18.3. Announcements	105
19. Using the Catalog Search	105
19.1. Accessing Catalog UI	105
19.2. Workspaces in Catalog UI	106
19.3. Searching with Catalog UI	109
19.4. Viewing Search Results	111
20. Using the Standard Search	115
20.1. Search	116
20.2. Actions	119
20.3. Workspaces	120
20.4. Notifications	121
20.5. Activities	121
20.6. Downloads	121
20.7. Maps	121
21. Using the Simple Search	121
21.1. Search	121
Integrating	123
22. Data and Metadata	124
22.1. Metacards	124
22.2. JSON "Definition" Files	129
22.3. Data Validation Services	130
22.4. Catalog Taxonomy	131
23. Endpoints	140
23.1. Available Endpoints	141
23.2. Endpoint Utility Services	212
23.3. Developing Endpoints	213

24. Eventing.....	214
24.1. Eventing Components.....	215
25. Sources.....	217
25.1. Types of Sources.....	217
25.2. Federation Strategy.....	218
25.3. Federated Sources	221
25.4. Connected Sources.....	243
25.5. Catalog Stores	246
25.6. Catalog Providers.....	247
25.7. Storage Providers.....	253
26. Security Services	253
26.1. Encryption Service	253
26.2. Expansion Service	254
26.3. Security IDP.....	256
26.4. Security STS.....	257
Developing.....	267
27. System Data Flow	267
28. Catalog Framework API.....	268
28.1. Catalog API Design.....	269
28.2. Included Catalog Frameworks, Associated Components, and Configurations	276
28.3. Developing Complementary Catalog Frameworks	280
28.4. Catalog Development Fundamentals	294
29. Developing Sources.....	295
29.1. Developing Sources.....	295
30. Transformers	298
30.1. Input Transformers.....	300
30.2. Developing Input Transformers.....	306
30.3. Metocard Transformers	310
30.4. Developing Metocard Transformers	322
30.5. Query Response Transformers.....	324
30.6. Mime Type Mapper	335
30.7. Mime Type Resolver	335
30.8. Developing Query Response Transformers.....	338
31. Plugins	339
31.1. Types of Plugins	339
31.2. Pre-Ingest Plugins	348
31.3. Pre-Query Plugins	352
31.4. Pre-Resource Plugins	353
31.5. Pre-Subscription Plugins	354
31.6. Pre-Delivery Plugins	354
31.7. Pre-Create Storage Plugins	354

31.8. Pre-Update Storage Plugins	355
31.9. Post-Ingest Plugins.....	355
31.10. Post-Query Plugins	358
31.11. Post-Resource Plugins.....	360
31.12. Post-Create Storage Plugins.....	360
31.13. Post-Update Storage Plugins	361
31.14. Policy Plugins	361
31.15. Access Plugins	368
31.16. Developing Catalog Plugins.....	370
32. Operations.....	378
33. Resources.....	378
33.1. Content Item	380
33.2. Resource Components	382
33.3. Resource Readers.....	382
33.4. Resource Writers	387
33.5. Registry Clients.....	389
33.6. Directory Monitors	389
34. Queries.....	392
34.1. Filters	392
34.2. FilterBuilder API	393
34.3. FilterDelegates	394
34.4. Developing Filters	396
34.5. Query Options.....	407
35. Security Framework.....	409
35.1. Subject	409
35.2. Security Core.....	412
35.3. Security Encryption.....	414
35.4. Security LDAP	416
35.5. Security PDP	422
35.6. Web Service Security Architecture	424
35.7. Expansion Service	455
35.8. Securing SOAP	456
35.9. Security PEP	458
36. Metrics	459
36.1. Metrics Services	459
37. Action Framework	465
37.1. Actions	466
37.2. Action Providers	466
38. Migration API	468
38.1. Migration API	468
39. Application Service API	470

39.1. Application Service API	470
40. General Development Guidelines	473
40.1. Developing for DDF	474
40.2. OSGi Basics	474
40.3. Developing DDF Applications	479
Appendix A: Application Reference	485
A.1. Catalog UI Reference.....	485
A.2. Admin Reference.....	490
A.3. Catalog Reference	493
A.4. GeoWebCache Reference.....	495
A.5. Platform Reference	498
A.6. Registry Reference	499
A.7. Security Reference	506
A.8. Solr Catalog Reference	510
A.9. Spatial Reference.....	511
A.10. Search UI Reference	514
A.11. Resource Management Reference.....	515
Appendix B: Application Whitelists.....	517
B.1. Admin Whitelist	517
B.2. Catalog Whitelist	517
B.3. Platform Whitelist	518
B.4. Registry Whitelist	519
B.5. Security Whitelist	519
B.6. Solr Catalog Whitelist	520
B.7. Spatial Whitelist.....	520
B.8. Search UI Whitelist	521
Appendix C: All File Formats Supported	521
Appendix D: DDF Dependency List	536

1. License

Copyright (c) Codice Foundation.

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

Overview

2. About DDF

Distributed Data Framework (DDF) is a free and open-source common data layer that abstracts services and business logic from the underlying data structures to enable rapid integration of new data sources.

Licensed under [LGPL](#), DDF is an interoperability platform that provides secure and scalable discovery and retrieval from a wide array of disparate sources.

DDF is:

- a flexible and modular integration framework.
- built to "unzip and run" while having the ability to scale to large enterprise systems.
- primarily focused on data integration, enabling clients to insert, query, and transform information from disparate data sources via the DDF Catalog.

2.1. Applications

DDF is comprised of several modular applications, to be installed or uninstalled as needed.

Admin Application

Enhances administrative capabilities when installing and managing DDF. It contains various services and interfaces that allow administrators more control over their systems.

Catalog Application

Provides a framework for storing, searching, processing, and transforming information. Clients typically perform local and/or federated query, create, read, update, and delete (QCRUD) operations against the Catalog. At the core of the Catalog functionality is the **Catalog Framework**, which routes all requests and responses through the system, invoking additional processing per the system configuration.

Platform Application

The Core application of the distribution. The Platform application has fundamental building blocks that the distribution needs to run.

Security Application

Provides authentication, authorization, and auditing services for the DDF. It is both a framework that developers and integrators can extend and a reference implementation that meets security requirements.

Solr Catalog Application

Includes the Solr Catalog Provider, an implementation of the Catalog Provider using [Apache Solr](#) as a data store.

Spatial Application

Provides OGC services, such as [CSW](#), [WCS](#), [WFS](#), and [KML](#).

Search UI

Allows a user to search for records in the local Catalog (provider) and federated sources. Results of the search are returned and displayed on a globe or map, providing a visual representation of where the records were found.

3. Documentation Guide

The DDF documentation is organized by audience.

Core Concepts

This introduction section is intended to give a high level overview of the concepts and capabilities of DDF.

Administrators

[Managing](#) | Administrators will be installing, maintaining, and supporting existing applications. Use this section to [prepare](#), [install](#), [configure](#), [run](#), and [monitor](#) a DDF.

Users

[Using](#) | Users interact with the system to search data stores. Use this section to navigate the various user interfaces available in DDF.

Integrators

[Integrating](#) | Integrators will use the existing applications to support their external frameworks. This section will provide details for finding, accessing and using the components of DDF.

Developers

[Developing](#) | Developers will build or extend the functionality of the applications.

3.1. Documentation Conventions

The following conventions are used within this documentation:

3.1.1. Customizable Values

Many values used in descriptions are customizable and should be changed for specific use cases. These values are denoted by `< >`, and by `[[]]` when within XML syntax. When using a real value, the placeholder characters should be omitted.

3.1.2. Code Values

Java objects, lines of code, or file properties are denoted with the [Monospace](#) font style. Example: `ddf.catalog.CatalogFramework`

3.1.3. Hyperlinks

Some hyperlinks (e.g., [/admin](#)) within the documentation assume a locally running installation of DDF. Simply change the hostname if accessing a remote host.

3.2. Support

Questions about DDF should be posted to the [ddf-users forum](#) or [ddf-developers forum](#), where they will be responded to quickly by a member of the DDF team.

3.2.1. Documentation Updates

The most current DDF documentation is available at [DDF Documentation](#).

Core Concepts

This introduction section is intended to give a high level overview of the concepts and capabilities of DDF.

4. Search

DDF provides the capability to search the Catalog for metadata. There are a number of different types of searches that can be performed on the Catalog, 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.

4.1. Search Types

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

4.1.1. Text Search

A text search is used when searching for textual information. It searches all textual metadata fields by default, although it is possible to refine searches to a text search on a single attribute. It is similar to a Google search over the metadata contained in the Catalog. Text searches may use wildcards, logical operators, and approximate matches.

4.1.2. Spatial Search

A spatial search is used for Area of Interest (AOI) searches. Polygon and point radius searches are supported.

4.1.3. 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 use the `created` or `modified` date attributes.

4.1.4. Datatype Search

A datatype search is used to search for metadata based on the datatype of the resource. 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.

5. Metadata

In DDF, `resources` are the data products, files, reports, or documents of interest to users of the

system.

Metadata is information about those resources, organized into a schema to make search possible. The Catalog stores this metadata and allows access to it. Metacards are single instances of metadata, representing a single resource, in the Catalog. Metacards follow one of several schemas to ensure reliable, accurate, and complete metadata. Essentially, Metacards function as containers of metadata.

6. Ingest

Ingest is the process of bringing data products, metadata, or both into the catalog to enable search, sharing, and discovery. Ingested files are [transformed](#) into a neutral format that can be searched against as well as migrated to other formats and systems. See [Ingesting Data](#) for the various methods of ingesting data.

6.1. Populating Metacards During Ingest

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

7. 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 implementation is provided by default.

Storage providers act as a proxy between the Catalog Framework and the mechanism storing the content. Storage providers expose the storage mechanism to the Catalog Framework. Storage plugins provide pluggable functionality that can be executed either immediately before or immediately after content has been stored or updated.

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

8. Catalog Framework

The Catalog Framework wires all the Catalog components together.

It is responsible for routing Catalog requests and responses to the appropriate source, destination, federated system, etc.

[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](#).

The Catalog Framework decouples clients from service implementations and provides integration

points for Catalog Plugins and convenience methods for Endpoint developers.

9. Federation

Federation is the ability of the DDF to query other data sources, including other DDFs. By default, the DDF is able to federate using [OpenSearch](#) and [CSW](#) protocols. The minimum configuration necessary to configure those federations is to supply a query address.

Federation enables constructing dynamic networks of data sources that can be queried individually, or aggregated into specific configuration to enable a wider range of accessibility for data and data products.

Federation provides the capability to extend the DDF enterprise to include [Remote Sources](#), which may include other instances of DDF. The Catalog handles all aspects of federated queries as they are sent to the Catalog Provider and Remote Sources, as they are processed, and as the query results are returned. Queries can be scoped to include only the local Catalog Provider (and any Connected Sources), only specific Federated Sources, or the entire enterprise (which includes all local and Remote Sources). If the query is supposed to be federated, the Catalog Framework passes the query to a Federation Strategy, which is responsible for querying each federated source that is specified. The Catalog Framework is also responsible for receiving the query results from each federated source and returning them to the client in the order specified by the particular federation strategy used. After the federation strategy handles the results, the Catalog returns them to the client through the Endpoint. Query results returned from a federated query are a list of metacards. The source ID in each metacard identifies the Source from which the metacard originated.

10. Events and Subscriptions

DDF can be configured to receive metacards whenever metadata is created, updated, or deleted in any federated sources. Creations, updates, and deletions are collectively called **Events**, and the process of registering to receive them is called **Subscription**.

The behavior of these subscriptions is consistent, but the method of configuring them is specific to the [Endpoint](#) used.

11. Registry

The Registry Application serves as an index of registry nodes and their information, including service bindings, configurations and supplemental details.

Each registry has the capability to serve as an index of information about a network of registries which, in turn, can be used to connect across a network of DDFs and other data sources. Registries communicate with each other through the CSW endpoint and each registry node is converted into a registry metacard to be stored in the catalog. When a registry is subscribed to or published from, it sends the details of one or more nodes to another registry.

11.1. Identity Node

The Registry is initially comprised of a single registry node, referred to as the **identity**, which represents the registry's primary configuration.

11.2. Subscription

Subscribing to a registry is the act of retrieving its information, specifically its identity information and any other registries it knows about. By default, subscriptions are configured to check for updates every 30 seconds.

11.3. Publication

Publishing is the act of sending a registry's information to another registry. Once publication has occurred, any updates to the local registry will be pushed out to the registries that have been published to.

12. Endpoints

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

Endpoint interface formats encompass a variety of protocols, 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.

Managing

Administrators will be installing, maintaining, and supporting existing applications. Use this section to prepare, install, configure, run, and monitor a DDF.

13. Installing

Set up a complete, secure instance of DDF. For simplified steps used for a testing, development, or demonstration installation, see the [DDF Quick Start](#).

IMPORTANT

Although DDF can be installed by any user, it is recommended for security reasons to have a non-**root** user execute the DDF installation.

NOTE

Hardening guidance assumes a Standard installation.

Adding other components does not have any security/hardening implications.

13.1. Installation Prerequisites

These are the system/environment requirements to configure *prior* to an installation.

Hardware Requirements

- At least 4096MB of memory for DDF.
 - This amount can be increased to support memory-intensive applications. See [Memory Considerations](#).

Java Requirements

- Supported platforms are *NIX - Unix/Linux/OSX, Solaris, and Windows.
- [JDK8](#) must be installed.
- The **JAVA_HOME** environment variable must be set to the location where the JDK is installed.
 1. Install/Upgrade to Java 8 [J2SE 8 SDK](#)
 - a. The recommended version is [8u60](#) or later.
 - b. Java version must contain only number values.
 2. Install/Upgrade to [JDK8](#).
 3. Set the **JAVA_HOME** environment variable to the location where the JDK is installed.

Setting **JAVA_HOME** variable

*NIX

WARNING Unlink `JAVA_HOME` if it is already linked to a previous version of the JRE:
`unlink JAVA_HOME`

Replace `<JAVA_VERSION>` with the version and build number installed.

1. Open a terminal window(*NIX) or command prompt (Windows) with administrator privileges.
2. Determine Java Installation Directory (This varies between operating system versions).

*Find Java Path in *NIX*

```
which java
```

Find Java Path in Windows

The path to the JDK can vary between versions of Windows, so manually verify the path under:

```
C:\Program Files\Java\jdk<M.m.p_build>
```

3. Copy path of Java installation to clipboard. (example: `/usr/java/<JAVA_VERSION>`)
4. Set `JAVA_HOME` by replacing `<PATH_TO_JAVA>` with the copied path in this command:

*Setting JAVA_HOME on *NIX*

```
JAVA_HOME=<PATH_TO_JAVA><JAVA_VERSION>
export JAVA_HOME
```

Setting JAVA_HOME on Windows

```
set JAVA_HOME=<PATH_TO_JAVA><JAVA_VERSION>
setx JAVA_HOME "<PATH_TO_JAVA><JAVA_VERSION>" /m
```

Adding JAVA_HOME to PATH Environment Variable on Windows

```
setx PATH "%PATH%;%JAVA_HOME%\bin" /m
```

5. Restart Terminal (shell) or Command Prompt.
 - Verify that the `JAVA_HOME` was set correctly.

*NIX

```
echo $JAVA_HOME
```

Windows

```
echo %JAVA_HOME%
```

File Descriptor Limit on Linux

- For Linux systems, increase the file descriptor limit by editing [/etc/sysctl.conf](#) to include:

```
fs.file-max = 6815744
```

NOTE

- For the change to take effect, a restart is required.

**Nix Restart Command*

```
init 6
```

13.2. Installing With the DDF Distribution Zip

Check System Time

WARNING

Prior to installing DDF, ensure the system time is accurate to prevent federation issues.

To install the DDF distribution zip, perform the following:

1. Download the DDF [zip file](#).
2. After the [prerequisites](#) have been met, change the current directory to the desired install directory, creating a new directory if desired. This will be referred to as [`<DDF_HOME>`](#).

Windows Pathname Warning

WARNING

Do not use spaces in directory or file names of the [`<DDF_HOME>`](#) path. For example, do not install in the default [Program Files](#) directory.

*Example: Create a Directory (Windows and *NIX)*

```
mkdir new_installation
```

- a. Use a Non-[root](#) User on *NIX. (Windows users skip this step)

It is recommended that the **root** user create a new install directory that can be owned by a non-**root** user (e.g., DDF_USER). This can be a new or existing user. This DDF_USER can now be used for the remaining installation instructions.

- b. Create a new group or use an existing group (e.g., DDF_GROUP) (Windows users skip this step)

*Example: Add New Group on *NIX*

```
groupadd DDF_GROUP
```

*Example: Switch User on *NIX*

```
chown DDF_USER:DDF_GROUP new_installation  
su - DDF_USER
```

3. Change the current directory to the location of the zip file (ddf-2.10.3.zip).

**NIX (Example assumes DDF has been downloaded to a CD/DVD)*

```
cd /home/user/cdrom
```

Windows (Example assumes DDF has been downloaded to the D drive)

```
cd D:\
```

4. Copy ddf-2.10.3.zip to <DDF_HOME>.

**NIX*

```
cp ddf-2.10.3.zip <DDF_HOME>
```

Windows

```
copy ddf-2.10.3.zip <DDF_HOME>
```

5. Change the current directory to the desired install location.

**NIX or Windows*

```
cd <DDF_HOME>
```

6. The DDF zip is now located within the <DDF_HOME>. Unzip ddf-2.10.3.zip.

*NIX

```
unzip ddf-2.10.3.zip
```

Windows Zip Utility Warning

DO NOT use the windows zip utility bundled with windows to unzip DDF. Unzipping DDF using the windows zip utility will cause unexpected behavior and errors in DDF!

WARNING

Use Java to Unzip in Windows

```
"%JAVA_HOME%\bin\jar.exe" xf ddf-2.10.3.zip
```

The unzipping process may take time to complete. The command prompt will stop responding to input during this time.

13.2.1. Controlling File System Access

Restrict access to sensitive files by ensuring that the only users with access privileges are administrators.

Within the <DDF_HOME>, a directory is created named ddf-2.10.3. This directory will be referred to in the documentation as <DDF_HOME>.

1. Do not assume the deployment is from a trusted source; verify its origination.
2. Check the available storage space on the system to ensure the deployment will not exceed the available space.
3. Set maximum storage space on the <DDF_HOME>/deploy and <DDF_HOME>/system directories to restrict the amount of space used by deployments.

Setting Directory Permissions

- **Required Step for Security Hardening**

DDF relies on the Directory Permissions of the host platform to protect the integrity of the DDF during operation. System administrators MUST perform the following steps prior to deploying bundles added to the DDF.

IMPORTANT

The system administrator must restrict certain directories to ensure that the application (user) cannot access restricted directories on the system. For example the **DDF_USER** should have read-only access to <DDF_HOME>, except for the sub-directories **etc**, **data** and **instances**.

Setting Directory Permissions on Windows

Set directory permissions on the `<DDF_HOME>`; all sub-directories except `etc`, `data`, and `instances`; and any directory intended to interact with the DDF to protect from unauthorized access.

1. Right-click on the `<DDF_HOME>` directory.
2. Select **Properties** → **Security** → **Advanced**.
3. Under **Owner**, select **Change**.
4. Enter `Creator Owner` into the **Enter the Object Name...** field.
5. Select **Check Names**.
6. Select **Apply**.
 - a. If prompted **Do you wish to continue**, select **Yes**.
7. Remove all Permission Entries for any groups or users with access to `<DIB_HOME>` other than **System**, **Administrators**, and **Creator Owner**.
 - a. Note: If prompted with a message such as: **You can't remove X because this object is inheriting permissions from its parent**. when removing entries from the Permission entries table:
 - i. Select **Disable Inheritance**.
 - ii. Select **Convert Inherited Permissions into explicit permissions on this object**.
 - iii. Try removing the entry again.
8. Select the option for **Replace all child object permission entries with inheritable permission entries from this object**.
9. Close the **Advanced Security Settings** window.

Setting Directory Permissions on *NIX

Set directory permissions to protect the DDF from unauthorized access.

- Change ownership of <DDF_HOME>
 - `chown -R ddf-user <DDF_HOME>`
- Create instances sub-directory if does not exist
 - `mkdir -p <DDF_HOME>/instances <DDF_HOME>/solr`
- Change group ownership on sub-directories
 - `chgrp -R DDF_GROUP <DDF_HOME>/etc <DDF_HOME>/data <DDF_HOME>/instances`
- Change group permissions
 - `chmod -R g-w <DDF_HOME>/etc <DDF_HOME>/data <DDF_HOME>/instances`
- Remove permissions for other users
 - `chmod -R o-rwx <DDF_HOME>/etc <DDF_HOME>/data <DDF_HOME>/instances`

13.3. Initial Startup

Run the DDF using the appropriate script.

*NIX

```
<DDF_HOME>/bin/ddf
```

Windows

```
<DDF_HOME>/bin/ddf.bat
```

The distribution takes a few moments to load depending on the hardware configuration.

TIP To run DDF as a service, see [Starting as a Service](#).

13.3.1. Verifying Startup

At this point, DDF should be configured and running with a Solr Catalog Provider. New features (endpoints, services, and sites) can be added as needed.

Verification is achieved by checking that all of the DDF bundles are in an **Active** state (excluding fragment bundles which remain in a **Resolved** state).

NOTE It may take a few moments for all bundles to start so it may be necessary to wait a few minutes before verifying installation.

Execute the following command to display the status of all the DDF bundles:

View Status

```
ddf@local>list | grep -i ddf
```

WARNING

Entries in the **Resolved** state are expected, they are OSGi bundle fragments. Bundle fragments are distinguished from other bundles in the command line console list by a field named **Hosts**, followed by a bundle number. Bundle fragments remain in the **Resolved** state and can never move to the **Active** state.

Example: Bundle Fragment in the Command Line Console

```
96 | Resolved | 80 | 2.10.0.SNAPSHOT | DDF :: Platform :: PaxWeb :: Jetty Config,  
Hosts: 90
```

After successfully completing these steps, the DDF is ready to be configured.

13.3.2. DDF Directory Contents after Installation and Initial Startup

During DDF installation, the major directories and files shown in the table below are created, modified, or replaced in the destination directory.

Table 1. DDF Directory Contents

Directory Name	Description
<code>bin</code>	Scripts to start, stop, and connect to DDF.
<code>data</code>	The working directory of the system – installed bundles and their data
<code>data/log/ddf.log</code>	Log file for DDF, logging all errors, warnings, and (optionally) debug statements. This log rolls up to 10 times, frequency based on a configurable setting (default=1 MB)
<code>data/log/ingest_error.log</code>	Log file for any ingest errors that occur within DDF.
<code>data/log/security.log</code>	Log file that records user interactions with the system for auditing purposes.
<code>data/log/solr.log</code>	Log file for any info coming from Solr.
<code>deploy</code>	Hot-deploy directory – KARs and bundles added to this directory will be hot-deployed (Empty upon DDF installation)
<code>documentation</code>	HTML and PDF copies of DDF documentation.
<code>etc</code>	Directory monitored for addition/modification/deletion of <code>.config</code> configuration files or third party <code>.cfg</code> configuration files.
<code>etc/failed</code>	If there is a problem with any of the <code>.config</code> files, such as bad syntax or missing tokens, they will be moved here.

Directory Name	Description
etc/processed	All successfully processed .config files will be moved here.
etc/templates	Template .config files for use in configuring DDF sources, settings, etc., by copying to the etc directory.
lib	The system's bootstrap libraries. Includes the ddf-branding.jar file which is used to brand the system console with the DDF logo.
licenses	Licensing information related to the system.
system	Local bundle repository. Contains all of the JARs required by DDF, including third-party JARs.

13.3.3. Completing Installation from the Admin Console

Upon startup, complete installation by navigating to the Admin Console at <https://localhost:8993/admin>.

Internet Explorer 10 TLS Warning

Internet Explorer 10 users may need to enable TLS 1.2 to access the Admin Console in the browser.

WARNING

Enabling TLS1.2 in IE10

1. Go to **Tools** → Internet Options → Advanced → Settings → Security.
2. Enable TLS1.2.

- Default user/password: **admin/admin**.

On the initial startup of the Admin Console, a series of prompts walks through essential configurations. These configurations can be changed later, if needed.

- Click **Start** to begin.

Configure Guest Claim Attributes Page

Setting the attributes on the **Configure Guest Claim Attributes** page determines the minimum claims attributes (and, therefore, permissions) available to a guest, or not signed-in, user.

To change this later, see [Configuring Guest Claim Attributes](#).

Setup Types

DDF is pre-configured with several installation profiles.

- Standard Installation: **Recommended**. Includes these applications by default:
 - [Admin](#)
 - [Catalog](#)
 - [Platform](#)
 - [Security](#)
 - [Solr Catalog](#)
 - [Spatial](#)
 - [Search UI](#)
- Development: Includes all demo, beta, and experimental applications.
- Custom Installation: **Advanced**. Click **Customize** on either profile to add or remove applications to be installed.
 - If apps are preselected when the **Select Applications** page is reached, they will be uninstalled if unselected.

The Platform, Admin, and Security applications are required and CANNOT be selected or unselected.

WARNING

The Security Application appears to be unselected upon first view of the tree structure, but it is in fact automatically installed with a later part of the installation process.

System Configuration Settings

- System Settings: Set `hostname` and `ports` for this installation.
- Contact Info: Contact information for the point-of-contact or administrator for this installation.
- Certificates: Add PKI certificates for the Keystore and Truststore for this installation.
 - For a quick (test) installation, if the hostname/ports are not changed from the defaults, DDF includes self-signed certificates to use. Do not use in a working installation.
 - For more advanced testing, on initial startup of the **Admin Console** append the string `?dev=true` to the url (<https://localhost:8993/admin?dev=true>) to auto-generate self-signed certificates from a demo Certificate Authority(CA) which will enable change hostname and port settings. Do not use in a working installation.
 - NOTE: `?dev=true` generates certificates on initial installation only.
 - For more information about importing certificate from a Certificate Authority, see [Managing Keystores and Certificates](#).

Finished Page

Upon successful startup, the **Finish** page will redirect to the Admin Console to begin further configuration, ingest, or federation.

NOTE The redirect will only work if the certificates are configured in the browser.
Otherwise the redirect link must be used.

14. Configuring

Configuration Requirements

NOTE Because components can easily be installed and uninstalled, it's important to remember that for proper DDF functionality, at least the Catalog API, one Endpoint, and one Catalog Framework implementation must be active.

DDF can be configured in several ways, depending on need:

NOTE While there are multiple ways to configure DDF for use, the recommended method is to use the Admin Console.

- RECOMMENDED: Using the browser and the Admin Console. [Configuring From the Admin Console](#)
- Using a terminal and the Command Console. [Configuring From the Command Console](#)
- Editing configuration files. [Configuring From Configuration Files](#)

- Importing the configurations from an existing DDF instance. [Importing Configurations](#)

Security is an important consideration for DDF, so it is imperative to update configurations away from the defaults to unique, secure settings.

Securing DDF Components

DDF is enabled with an Insecure Defaults Service which will warn users/admins if the system is configured with insecure defaults.

IMPORTANT

A banner is displayed on the admin console notifying "The system is insecure because default configuration values are in use."

A detailed view is available of the properties to update.

Security concerns will be highlighted in the configuration sections to follow.

Security Hardening

To harden DDF, extra security precautions are required.

Where available, necessary mitigations to harden an installation of DDF are called out in the following configuration steps.

Refer to the [Hardening Checklist](#) for a compilation of these mitigations.

14.1. Hardening Checklist

The following list enumerates the required mitigations needed for hardening:

- [Environment Hardening](#)
- [Set Directory Permissions](#)
- [Configure Keystore and Certificates](#)
- [Configure Certificate Revocation List](#)
- [Disallow Login Without Certificates](#)
- [Restricting Access to Admin Console](#)
- [Restrict Feature, App, Service, and Configuration Access](#)
- [Limit Access to the STS](#)
- [Remove Default Users](#)
- [Enable Guest Authentication \(if allowing Guest users\)](#)
- [Configure Guest Claim Attributes \(if allowing Guest users\)](#)
- [Harden Solr Index](#)
- [Isolate Solr Cloud and Zookeeper. \(If using\)](#)
- [Configure Auditing](#)

14.2. Managing Keystores and Certificates

- Required Step for Security Hardening

DDF uses certificates in two ways:

1. Ensuring the privacy and integrity of messages sent or received over a network.
2. Authenticating an incoming user request.

Managing Keystores

Certificates, and sometimes their associated private keys, are stored in keystore files. DDF includes two default keystore files, the server key store and the server trust store. The server keystore holds the certificates and private keys that DDF uses to identify itself to other nodes on the network. The truststore holds the certificates of nodes or other entities that DDF needs to trust.

Certificates (and certificates with keys) can be managed in the Admin Console. Navigate to the Securityapplication and select it. After it opens, select the notebook tab labeled "Certificates". This view shows the alias (name) of every certificate in the trust store and the key store. It also displays if the entry includes a private key ("Is Key") and the encryption scheme (typically "RSA" or "EC").

This view allows administrators remove certificates from DDF's key and trust stores. It also allows administrators to import certificates and private keys into the keystores with the "+" button. The import function has two options: import from a file or import over HTTPS. The file option accepts a Java Keystore file or a PKCS12 keystore file. Because keystores can hold many keys, the import dialog asks the administrator to provide the alias of the key to import. Private keys are typically encrypted and the import dialog prompts the administrator to enter the password for the private key. Additionally, keystore files themselves are typically encrypted and the dialog asks for the keystore ("Store") password.

The name and location of the DDF trust and key stores can be changed by editing the system properties files, [etc/system.properties](#). Additionally, the password that DDF uses to decrypt (unlock) the key and trust stores can be changed here.

IMPORTANT

DDF assumes that password used to unlock the keystore is the same password that unlocks private keys in the keystore.

The location, file name, passwords and type of the server and trust key stores can be set in the [system.properties](#) file:

1. Setting the Keystore and Truststore Java Properties

```
javax.net.ssl.keyStore=etc/keystores/serverKeystore.jks
javax.net.ssl.keyStorePassword=changeit
javax.net.ssl.trustStore=etc/keystores/serverTruststore.jks
javax.net.ssl.trustStorePassword=changeit
javax.net.ssl.keyStoreType=jks
javax.net.ssl.trustStoreType=jks
```

Default Certificates

DDF comes with a default keystore that contains certificates. This allows the distribution to be unzipped and run immediately.

If the installer was used to install the DDF and a hostname other than "localhost" was given, the user will be prompted to upload new trust/key stores.

If the hostname is `localhost` or, if the hostname was changed *after* installation, the default certificates will not allow access to the DDF instance from another machine over HTTPS (now the default for many services). The Demo Certificate Authority will need to be replaced with certificates that use the fully-qualified hostname of the server running the DDF instance.

Demo Certificate Authority (CA)

DDF comes with a populated truststore containing entries for many public certificate authorities, such as Go Daddy and Verisign. It also includes an entry for the DDF Demo Root CA. This entry is a self-signed certificate used for testing. It enables DDF to run immediately after unzipping the distribution. The keys and certificates for the DDF Demo Root CA are included as part of the DDF distribution. This entry must be removed from the truststore before DDF can operate securely.

Creating New Server Keystore Entry with the CertNew Scripts

To create a private key and certificate signed by the Demo Certificate Authority, use the provided scripts. To use the scripts, run them out of the `<INSTALL_HOME>/etc/certs` directory.

*NIX Demo CA Script

For *NIX, use the `CertNew.sh` script.

```
sh CertNew.sh <FQDN>
```

Alternatively, a distinguished name can be provided to the script with a comma-delimited string.

```
sh CertNew.sh -dn "c=US, st=California, o=Yoyodyne, l=San Narciso, cn=<FQDN>"
```

Windows Demo CA Script

For Windows, use the `CertNew.cmd` script.

```
CertNew -cn <FQDN>
```

Alternatively, a distinguished name can be provided to the script with a comma-delimited string.

```
CertNew -dn "c=US, st=California, o=Yoyodyne, l=San Narciso, cn=<FQDN>"
```

The `CertNew` scripts:

- Create a new entry in the server keystore.
- Use the hostname as the fully qualified domain name (FQDN) when creating the certificate.
- Use the Demo Certificate Authority to sign the certificate so that it will be trusted by the default configuration.

To install a certificate signed by a different Certificate Authority, see [Managing Keystores](#).

Finally, restart the DDF instance. Browse the Admin Console at <https://<FQDN>:8993/admin> to test changes.

WARNING

If the server's fully qualified domain name is not recognized, the name may need to be added to the network's DNS server.

The DDF instance can be tested even if there is no entry for the FQDN in the DNS. First, test if the FQDN is already recognized. Execute this command:

`ping <FQDN>`

TIP

If the command responds with an error message such as unknown host, then modify the system's `hosts` file to point the server's FQDN to the loopback address. For example:

`127.0.0.1 <FQDN>`

NOTE

By default, the Catalog Backup Post-Ingest Plugin is **NOT** enabled. To enable, the Enable Backup Plugin configuration item must be checked in the Backup Post-Ingest Plugin configuration.

`Enable Backup Plugin: true`

Changing Default Passwords

This step is not required for a hardened system. If testing DDF with a

- The default password in `config.ldif` for `serverKeystore.jks` is `changeit`. This needs to be modified.
 - `ds-cfg-key-store-file: ../../keystores/serverKeystore.jks`
 - `ds-cfg-key-store-type: JKS`
 - `ds-cfg-key-store-pin: password`
 - `cn: JKS`
- The default password in `config.ldif` for `serverTruststore.jks` is `changeit`. This needs to be modified.
 - `ds-cfg-trust-store-file: ../../keystores/serverTruststore.jks`
 - `ds-cfg-trust-store-pin: password`
 - `cn: JKS`

Updating Key Store / Trust Store via the Admin Console

1. Open the Admin Console.
2. Select the **Security** application.
3. Select the **Certificates** tab.
4. Add and remove certificates and private keys as necessary.
5. Restart DDF.

IMPORTANT The default trust store and key store files for DDF included in `etc keystores` use self-signed certificates. Self-signed certificates should never be used outside of development/testing areas.

Managing Certificate Revocation List (CRL)

- **Required Step for Security Hardening**

For hardening purposes, it is recommended to implement a way to verify the CRL at least daily.

A Certificate Revocation List is a collection of formerly-valid certificates that should explicitly *not* be accepted.

Creating a Certificate Revocation List (CRL)

Create a CRL in which the token issuer's certificate is valid. The example uses OpenSSL.

```
$> openssl ca -gencrl -out crl-tokenissuer-valid.pem
```

Windows and OpenSSL

NOTE Windows does not include OpenSSL by default. For Windows platforms, a additional download of [OpenSSL](#) or an alternative is required.

Revoke a Certificate and Create a New CRL that Contains the Revoked Certificate

```
$> openssl ca -revoke tokenissuer.crt  
$> openssl ca -gencrl -out crl-tokenissuer-revoked.pem
```

Viewing a CRL

1. Use the following command to view the serial numbers of the revoked certificates: `$> openssl crl -inform PEM -text -noout -in crl-tokenissuer-revoked.pem`

Enabling Revocation

NOTE Enabling CRL revocation or modifying the CRL file will require a restart of DDF to apply updates.

1. Place the CRL in <DDF_HOME>/etc/keystores.

2. Add the line `org.apache.ws.security.crypto.merlin.x509crl.file=etc/keystores/<CRL_FILENAME>` to the following files (Replace `<CRL_FILENAME>` with the URL or file path of the CRL location):
 - a. `<DDF_HOME>/etc/ws-security/server/encryption.properties`
 - b. `<DDF_HOME>/etc/ws-security/issuer/encryption.properties`
 - c. `<DDF_HOME>/etc/ws-security/server/signature.properties`
 - d. `<DDF_HOME>/etc/ws-security/issuer/signature.properties`
3. (Replace `<CRL_FILENAME>` with the file path or URL of the CRL file used in previous step.)

Adding this property will also enable CRL revocation for any context policy implementing PKI authentication. For example, adding an authentication policy in the Web Context Policy Manager of `/search=SAML|PKI` will disable basic authentication, require a certificate for the search UI, and allow a SAML SSO session to be created. If a certificate is not in the CRL, it will be allowed through, otherwise it will get a 401 error. If no certificate is provided, the guest handler will grant guest access.

This also enables CRL revocation for the STS endpoint. The STS CRL Interceptor monitors the same `encryption.properties` file and operates in an identical manner to the PKI Authentication's CRL handler. Enabling the CRL via the `encryption.properties` file will also enable it for the STS, and also requires a restart.

Add Revocation to a Web Context

The PKIHandler implements CRL revocation, so any web context that is configured to use PKI authentication will also use CRL revocation if revocation is enabled.

1. After enabling revocation (see above), open the **Web Context Policy Manager**.
2. Add or modify a Web Context to use PKI in authentication. For example, enabling CRL for the search ui endpoint would require adding an authorization policy of `/search=SAML|PKI`
3. If guest access is required, add `GUEST` to the policy. Ex, `/search=SAML|PKI|GUEST`.

With guest access, a user with a revoked certificate will be given a 401 error, but users without a certificate will be able to access the web context as the guest user.

The STS CRL interceptor does not need a web context specified. The CRL interceptor for the STS will become active after specifying the CRL file path, or the URL for the CRL, in the `encryption.properties` file and restarting DDF.

NOTE Disabling or enabling CRL revocation or modifying the CRL file will require a restart of DDF to apply updates. If CRL checking is already enabled, adding a new context via the **Web Context Policy Manager** will not require a restart.

Adding Revocation to an Endpoint

NOTE This section explains how to add CXF's CRL revocation method to an endpoint and not the CRL revocation method in the `PKIHandler`.

This guide assumes that the endpoint being created uses CXF and is being started via Blueprint

from inside the OSGi container. If other tools are being used the configuration may differ.

Add the following property to the `jasws` endpoint in the endpoint's `blueprint.xml`:

```
<entry key="ws-security.enableRevocation" value="true"/>
```

Example xml snippet for the `jaxws:endpoint` with the property:

```
<jaxws:endpoint id="Test" implementor="#testImpl"
    wsdlLocation="classpath: META-INF/wsdl/TestService.wsdl"
    address="/TestService">

    <jaxws:properties>
        <entry key="ws-security.enableRevocation" value="true"/>
    </jaxws:properties>
</jaxws:endpoint>
```

Verifying Revocation

A **Warning** similar to the following will be displayed in the logs of the source and endpoint showing the exception encountered during certificate validation:

```

11:48:00,016 | WARN  | tp2085517656-302 | WSS4JInInterceptor           |
security.wss4j.WSS4JInInterceptor 330 | 164 - org.apache.cxf.cxf-rt-ws-security -
2.7.3 |
org.apache.ws.security.WSSecurityException: General security error (Error during
certificate path validation: Certificate has been revoked, reason: unspecified)
    at
org.apache.ws.security.components.crypto.Merlin.verifyTrust(Merlin.java:838)[161:org.a
pache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.validate.SignatureTrustValidator.verifyTrustInCert(SignatureTru
stValidator.java:213)[161:org.apache.ws.security.wss4j:1.6.9]

[ ... section removed for space]

Caused by: java.security.cert.CertPathValidatorException: Certificate has been
revoked, reason: unspecified
    at
sun.security.provider.certpath.PKIXMasterCertPathValidator.validate(PKIXMasterCertPath
Validator.java:139)[:1.6.0_33]
    at
sun.security.provider.certpath.PKIXCertPathValidator.doValidate(PKIXCertPathValidator.
java:330)[:1.6.0_33]
    at
sun.security.provider.certpath.PKIXCertPathValidator.engineValidate(PKIXCertPathValida
tor.java:178)[:1.6.0_33]
    at
java.security.cert.CertPathValidator.validate(CertPathValidator.java:250)[:1.6.0_33]
    at
org.apache.ws.security.components.crypto.Merlin.verifyTrust(Merlin.java:814)[161:org.a
pache.ws.security.wss4j:1.6.9]
    ... 45 more

```

Disallowing Login Without Certificates

DDF can be configured to prevent login without a valid PKI certificate.

- Navigate to **Admin Console**
- Under **Security**, select → **Web Context Policy Manager**
- Add a policy for each context requiring restriction
 - For example: `/search=SAML|PKI` will disallow login without certificates to the Search UI.
 - The format for the policy should be: `/<CONTEXT>=SAML|PKI`
- Click **Save**

NOTE Ensure certificates comply with organizational hardening policies.

14.3. Configuring from the Admin Console

The Admin Console is the centralized location for administering the system. The Admin Console allows an administrator to install and remove selected applications and their dependencies and access configuration pages to configure and tailor system services and properties. The default address for the Admin Console is <https://localhost:8993/admin>.

Use this brief tutorial or start at [Securing Admin Console](#).

Admin Console Tutorial

This overview covers general uses for the Admin Console.

Managing Applications from Admin Console

The **Manage** button enables activation/deactivation and adding/removing applications.

Activating / Deactivating Applications

The **Deactivate** button stops individual applications and any dependent apps. Certain applications are central to overall functionality and cannot be deactivated. These will have the **Deactivate** button disabled. Disabled apps will be moved to a list at the bottom of the page, with an enable button to reactivate, if desired.

The **Add Application** button is at the end of the list of currently active applications.

Removing Applications

To remove an application, it must first be deactivated. This enables the **Remove Application** button.

Upgrading Applications

Each application tile includes an upgrade button to select a new version to install.

System Settings Tab

The configuration and features installed can be viewed and edited from the System tab as well; however, it is recommended that configuration be managed from the applications tab.

IMPORTANT In general, applications should be managed via the applications tab. Configuration via this page could result in an unstable system. Proceed with caution!

Managing Federation in the Admin Console

It is recommended to use the **Catalog App** → **Sources** tab to configure and manage sites/sources.

Viewing Currently Active Applications from Admin Console

DDF displays all active applications in the Admin Console. This view can be configured according to preference. Either view has an > arrow icon to view more information about the application as currently configured.

Table 2. Admin Console Views

View	Description
Tile View	The first view presented is the Tile View, displaying all active applications as individual tiles.
List View	Optionally, active applications can be displayed in a list format by clicking the list view button.

Application Detailed View

Each individual application has a detailed view to view information specific to that application, adjust configurations or enable/disable features. All applications have a standard set of tabs, although some apps may have additional ones with further information.

Table 3. Individual Application Views

Tab	Explanation
Configuration	The Configuration tab lists all bundles associated with the application as links to configure any configurable properties of that bundle.
Details	The Details tab gives a description, version, status, and list of other applications that are required by, or rely on, the current application.
Features	The features tab breaks down the individual features of the application that can be installed or uninstalled as configurable features.

Managing Features Using the Admin Console

DDF includes many components, packaged as *features*, that can be installed and/or uninstalled without restarting the system. Features are collections of OSGi bundles, configuration data, and/or other features.

Transitive Dependencies

NOTE Features may have dependencies on other features and will auto-install them as needed.

In the Admin Console, Features are found on the **Features** tab of each application.

1. Select the appropriate application.
2. Select the **Features** tab.
3. Uninstalled features are shown with a **play** arrow under the **Actions** column.
 - a. Select the **play** arrow for the desired feature.
 - b. The **Status** will change from **Uninstalled** to **Installed**.
4. Installed features are shown with a **stop** icon under the **Actions** column.
 - a. Select the **stop** icon for the desired feature.
 - b. The **Status** will change from **Installed** to **Uninstalled**.

Adding Feature Repositories

If needed, custom feature repositories can be added to extend DDF functionality.

1. Select the **Manage** button in the upper right.
2. Select the **Add an Application** tile
3. Select **File Upload** to add a new **.kar** OR **.jar** file.
4. Select the **Maven URL** tab and enter the URL of the feature repository.
 - a. Select the **Add URL** button.
5. Select the **Save Changes** button.

14.3.1. Securing Admin Console

If you have integrated DDF with your existing security infrastructure, then you may want to limit access to parts of the DDF based on user roles/groups.

Restricting Access to Admin Console

- **Required Step for Security Hardening**

Limit access to the Admin Console to those users who need access. To set access restrictions on the Admin Console, consult the organization's security architecture to identify specific realms, authentication methods, and roles required.

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select the **Web Context Policy Manager**.
 - a. A dialogue will pop up that allows you to edit DDF access restrictions.
 - b. Once you have configured your **realms** in your security infrastructure, you can associate them with DDF contexts.
 - c. If your infrastructure supports multiple **authentication methods**, they may be specified on a per-context basis.
 - d. Role requirements may be enforced by configuring the **required attributes** for a given context.
 - e. The **white listed contexts** allows child contexts to be excluded from the authentication constraints of their parents.

Restricting Feature, App, Service, and Configuration Access

- **Required Step for Security Hardening**

Limit access to the individual applications, features, or services to those users who need access. Organizational requirements should dictate which applications are restricted and the extent to

which they are restricted.

1. Navigate to the **Admin Console**.
2. Select the **Admin** application.
3. Select the **Configuration** tab.
4. Select the **Admin Configuration Policy**.

5. To add a feature or app permission:

- a. Add a new field to "Feature and App Permissions" in the format of:

```
<feature name>/<app name> = "attribute name=attribute value","attribute name2=attribute value2", ...
```

- b. For example, to restrict access of any user without an admin role to the catalog-app:

```
catalog-app = "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=admin", ...
```

6. To add a configuration permission:

- a. Add a new field to "Configuration Permissions" in the format of:

```
configuration id = "attribute name=attribute value","attribute name2=attribute value2", ...
```

- b. For example, to restrict access of any user without an admin role to the Web Context Policy Manager:

```
org.codice.ddf.security.policy.context.impl.PolicyManager="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=admin"
```

If a permission is specified, any user without the required attributes will be unable to see or modify the feature, app, or configuration.

14.3.2. Configuring for an LDAP Server

WARNING

The configurations for Security STS LDAP and Roles Claims Handler and Security STS LDAP Login contain plain text default passwords for the embedded LDAP, which is insecure to use in production.

Use the encryption service, described in [Encryption Service](#), from the Command Console to set passwords for your LDAP server. Then change the LDAP Bind User Password in the configurations to use the encrypted password.

Table 4. STS Ldap Login Configuration

Name	Property	Type	Description	Default Value	Required
LDAP URL	<code>ldapUrl</code>	String	LDAP or LDAPS server and port	<code>ldaps://\${org.codice.ddf stem.hostname}:1636</code>	yes
StartTLS	<code>startTls</code>	Boolean	Determines whether or not to use StartTLS when connecting via the ldap protocol. This setting is ignored if the URL uses ldaps.	false	yes
LDAP Bind User DN	<code>ldapBindUserDn</code>	String	DN of the user to bind with LDAP. This user should have the ability to verify passwords and read attributes for any user.	<code>cn=admin</code>	yes
LDAP Bind User Password	<code>ldapBindUserPass</code>	Password	Password used to bind user with LDAP.	<code>secret</code>	yes
LDAP Username Attribute	<code>userNameAttribute</code>	String	Attribute used to designate the user's name in LDAP. Usually this is uid, cn, or something similar.	<code>uid</code>	yes
LDAP Base User DN	<code>userBaseDn</code>	String	Full LDAP path to where users can be found.	<code>ou=users,dc=example,dc=com</code>	yes
LDAP Base Group DN	<code>groupBaseDn</code>	String	<code>ou=groups,dc=example,dc=com</code>	Full LDAP path to where groups can be found.	yes

Configuring STS Claims Handlers

A claim is an additional piece of data about a principal that can be included in a token along with basic token data. A claims manager provides hooks for a developer to plug in claims handlers to ensure that the STS includes the specified claims in the issued token.

Claims handlers convert incoming user credentials into a set of attribute claims that will be populated in the SAML assertion. For example, the `LDAPClaimsHandler` takes in the user's credentials and retrieves the user's attributes from a backend LDAP server. These attributes are then mapped and added to the SAML assertion being created. Integrators and developers can add more claims handlers that can handle other types of external services that store user attributes.

Table 5. Security STS LDAP and Roles Claims Handler

Name	Property	Type	Description	Default Value	Required
LDAP URL	url	String	true	ldaps://\${org.codice.ddf stem.hostname}:1636	LDAP or LDAPS server and port
StartTLS	startTls	Boolean	Determines whether or not to use StartTLS when connecting via the ldap protocol. This setting is ignored if the URL uses ldaps.	false	true
LDAP Bind User DN	ldapBindUserDn	String	DN of the user to bind with LDAP. This user should have the ability to verify passwords and read attributes for any user.	cn=admin	true
LDAP Bind User Password	password	Password	Password used to bind user with LDAP.	secret	true
LDAP Group User Membership Attribute	membershipUserAttribute	String	Attribute used as the membership attribute for the user in the group. Usually this is uid, cn, or something similar.	uid	true
LDAP User Login Attribute	loginUserAttribute	String	Attribute used as the login username. Usually this is uid, cn, or something similar.	uid	true
LDAP Base User DN	userBaseDn	String	Full LDAP path to where users can be found.	ou=users,dc=example,dc=com	true
Override User Certificate DN	overrideCertDn	Boolean	When checked, this setting will ignore the DN of a user and instead use the LDAP Base User DN value.	false	true
LDAP Group ObjectClass	objectClass	String	ObjectClass that defines structure for group membership in LDAP. Usually this is groupOfNames or groupOfUniqueNames.	groupOfNames	true

Name	Property	Type	Description	Default Value	Required
LDAP Membership Attribute	memberNameAttribute	String	Attribute used to designate the user's name as a member of the group in LDAP. Usually this is member or uniqueMember.	member	true
LDAP Base Group DN	groupBaseDn	String	Full LDAP path to where groups can be found.	ou=groups\dc=example,dc=com	true
Attribute Map File	propertyFileLocation	String	Location of the file which contains user attribute maps to use.	<INSTALL_HOME>/etc/ws-security/attributeMap.properties	true

14.3.3. Removing Default Users

- Required Step for Security Hardening

Once DDF is configured to use an external user (such as LDAP), remove the `users.properties` file from the `<INSTALL_HOME>/etc` directory. Use of a `users.properties` file should be limited to emergency recovery operations and replaced as soon as effectively possible.

Emergency Use of `users.properties` file

Typically, the DDF does not manage passwords. Authenticators are stored in an external identity management solution. However, DDF may be configure the `users.properties` file to include an account with a username and password for emergency use.

NOTE If a system recovery account is configured in `users.properties`, ensure:

- The use of this account should be for as short a time as possible.
- The default username/password of “admin/admin” should not be used.
- All organizational standards for password complexity should still apply.
- The password should be encrypted.

NOTE It is recommended to perform yearly reviews of accounts for compliance with organizational account management requirements.

14.3.4. Hardening Guest User Access to the Search UI

This section explains how to protect the Search UI page from guest users. Depending on how the Search UI page is protected, users might be prompted with a login page to enter their credentials. Only authorized users are then allowed to continue to the Search UI page. By default, the Search UI

allows guest access as part of the karaf security realm. The security settings for the Search UI and all other web contexts can be changed via the [Web Context Policy Manager](#) configuration.

These instructions assume that all security components are running on the same physical or virtual machine. For installations where some or all of these components reside on different network locations, adjust accordingly.

- Make sure that all the default logical names for locations of the security services are defined.

Configuring Guest User for Unauthenticated Metadata Access

Unauthenticated access to a secured DDF system is provided by the **Guest** user. Guest authentication must be enabled to allow guest users. Once the guest user is configured, redaction and filtering of metadata is done for the guest user the same way it is done for normal users.

Enabling Guest Authentication

To enable guest authentication for a context, change the **Authentication Type** for that context to **Guest**.

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **Web Context Policy Manager**.
5. Select the desired context (`/`, `/search`, `/admin`, etc.).
6. Add **Guest** to the **Authentication Type** list.
 - a. Separate entries with a `|` symbol (eg. `/=SAML|Guest`).

Configuring Guest Interceptor

- **Required Step for Security Hardening**

If a legacy client requires the use of the secured SOAP endpoints, the **guest interceptor** should be configured. Otherwise, the guest interceptor and **public** endpoints should be uninstalled for a hardened system.

Configuring Guest Claim Attributes

A guest user's attributes define the most permissive set of claims for an unauthenticated user.

A guest user's claim attributes are stored in configuration, not in the LDAP as normal authenticated users' attributes are.

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select the **Security Guest Claims Handler**.

5. Add any additional attributes desired for the guest user.
6. Save changes.

14.3.5. Configuring HTTP Port from Admin Console

IMPORTANT

Do not use the Admin Console to change the HTTP port. While the Admin Console's Pax Web Runtime offers this configuration option, it has proven to be unreliable and may crash the system. Use the [Command Console](#) instead.

14.3.6. Configuring HTTP to HTTPS Proxy From the Admin Console

The [platform-http-proxy](#) feature proxies https to http for clients that cannot use HTTPS and should not have HTTP enabled for the entire container via the [etc/org.ops4j.pax.web.cfg](#) file.

1. Click the **Platform** application tile.
2. Choose the **Features** tab.
3. Select [platform-http-proxy](#).
4. Click on the **Play** button to the right of the word "Uninstalled"

Configuring the proxy:

NOTE

The hostname should be set by default. Only configure the proxy if this is not working.

1. Select **Configuration** tab.
2. Select **HTTP to HTTPS Proxy Settings**
 - a. Enter the Hostname to use for HTTPS connection in the proxy.
3. Click **Save changes**.

14.3.7. Configuring the Web Context Policy Manager

The Web Context Policy Manager defines all security policies for REST endpoints within DDF. It defines:

- the realms a context should authenticate against.
- the type of authentication that a context requires.
- any user attributes required for authorization.

See [Web Context Policy Manager Configurations](#) for detailed descriptions of all fields.

Context Realms

The karaf realm is the only realm available by default and it authenticates against the [users.properties](#) file. As JAAS authentication realms are added to the STS, more realms become available to authenticate against.

For example, installing the `security-sts-ldaplogin` feature adds an ldap realm. Contexts can then be pointed to the ldap realm for authentication and STS will be instructed to authenticate them against ldap.

Authentication Types

As you add REST endpoints, you may need to add different types of authentication through the Web Context Policy Manager.

Any web context that allows or requires specific authentication types should be added here with the following format:

```
/<CONTEXT>=<AUTH_TYPE>|<AUTH_TYPE>|...
```

Table 6. Default Types of Authentication

Authentication Type	Description
<code>saml</code>	Activates single-sign on (SSO) across all REST endpoints that use SAML.
<code>basic</code>	Activates basic authentication.
<code>PKI</code>	Activates public key infrastructure authentication.
<code>IdP</code>	Activates SAML Web SSO authentication support. Additional configuration is necessary.
<code>CAS</code>	Enables SSO through a Central Authentication Server
<code>guest</code>	provides guest access

Required Attributes

The fields for required attributes allows configuring certain contexts to only be accessible to users with pre-defined attributes. For example, the default required attribute for the `/admin` context is `role=system-admin`, limiting access to the Admin Console to system administrators

White Listed Contexts

White listed contexts are trusted contexts which will bypass security. Any sub-contexts of a white listed context will be white listed as well, unless they are specifically assigned a policy.

Limiting Access to the STS

- **Required Step for Security Hardening**

Be sure to limit the hosts that are allowed to connect to the STS:

- Open the `<DDF_HOME>/etc/system.properties` file.
- Edit the line `ws-security.subject.cert.constraints = .*`.
 - Remove the `.*` and replace with a comma-delimited list of desired hosts (`<MY_HOST>`):

- ws-security.subject.cert.constraints = <MY_HOST>,<OTHER_HOST>

14.3.8. Reconfiguring DDF with a Different Catalog Provider

This scenario describes how to reconfigure DDF to use a different catalog provider.

This scenario assumes DDF is already running.

Uninstall Catalog Provider (if installed).

1. Navigate to the **Admin Console**.
2. Select the **Solr Catalog** application.
3. Select the **Features** tab.
4. Find and Stop the installed Catalog Provider

Install the new Catalog Provider

1. Navigate to the **Admin Console**.
2. Select the **Solr Catalog** application.
3. Select the **Features** tab.
4. Find and Start the desired Catalog Provider.

14.3.9. Configuring DDF as a Fanout Proxy

This scenario describes how to configure DDF as a fanout proxy such that only queries and resource retrieval requests are processed and create/update/delete requests are rejected. All queries are enterprise queries and no catalog provider needs to be configured.

1. Reconfigure DDF in fanout proxy mode by going to the Features tab in the
2. Navigate to the **Admin Console**.
3. Select the **Catalog** application.
4. Select the **Configuration** tab.
5. Select **Catalog Standard Framework**.
6. Select **Enable Fanout Proxy**.
7. Save changes.

DDF is now operating as a fanout proxy. Only queries and resource retrieval requests will be allowed. All queries will be federated. Create, update, and delete requests will throw an **UnsupportedOperationException**, even if a Catalog Provider was configured prior to the reconfiguration to fanout.

14.3.10. Configuring the Product Cache from the Admin Console

All caching properties are part of the [Resource Download Settings](#).

Invalidating the Product Cache

1. The product cache directory can be administratively invalidated by turning off the product caching using the Enable Product Caching configuration.
2. Alternatively, an administrator may manually invalidate products by removing them from the file system. Products are cached at the directory specified in the Product Cache Directory configuration.

Format:

<INSTALL-DIR>/data/product-cache/<source-id>-<metocard-id>

Example:

<INSTALL-DIR>/data/product-cache/abc123

See [Metocard Ingest Network Plugin](#).

14.3.11. Configuring Solr from Admin Console

Solr can only be configured/installed from configuration files. See [Solr System Properties](#) and [Configuring Solr Catalog Provider Data Directory](#).

14.3.12. Securing Identity Provider/Service Provider

The Security Identity Provider (IdP) application provides service provider handling that satisfies the [SAML 2.0 Web SSO profile](#) in order to support external IdPs (Identity Providers).

IdP (Identity Provider) and SP (Service Provider)

IdP and SP are used for SSO authentication purposes.

Installing the IdP

The IdP bundles are not installed by default. They can be started by installing the `security-idp` feature.

1. Install the `security-idp` feature either by command line: `features:install security-idp`, or by the Admin Console: **Security** → **Features** → `security-idp`

Security IdP Service Provider (SP)

The IdP client that interacts with the specified Identity Provider.

IdP SSO Configuration

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

5. Populate IdP Metadata field through one of the following:

- a. An HTTPS URL (<https://>)
- b. A file URL (file:)
- c. An XML block to refer to desired metadata
 - i. (e.g., <https://localhost:8993/services/idp/login/metadata>)

IdP Client (SP) example.xml

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://localhost:8993/services/idp/login">
  <md:IDPSSODescriptor WantAuthnRequestsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>

MIIDEzCCAnygAwIBAgIJIAIzc4FYrIp9mMA0GCSqGSIb3DQEBBQUAMhcxCzAJBgNVBAYTA1VTMQswCQYDVQQIDA
JBWjEMMAoGA1UECgwDRERGMQwwCgYDVQQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBpkq
hkIG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMI
GDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWFrMQwwCgYDVQQKDANEREYxDDAK
BgNVBAsMA0RldjESMBAGA1UEAwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhXBsZS
5vcmcwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZhQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx
313A99s8WtA4mAeb6n0vTR9yNB0ekW4nY0iE0q//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7
+Pk1r0TDI4MRzPPkkTp4w1ORQ/a6CfvNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBvhvC
AQ0EGhYYRk9SIFRFU1RJTkcgvUFVSUE9TRSBPTkxZMB0GA1UDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrsDAfBg
NVHSMEGDAwBThVMex3wrCv61feF47CvkSBe9xjAgBgNVHREEGTAxRvzb2NhbGhvc3RAZXhhXBsZS5vcmcw
DQYJKoZIhvcNAQEFBQADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx0
7n1Sm0ESp5sE3jGTnh0GtYV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZ
CfT9m07XduxjsmDz0h1SGV=
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:KeyDescriptor use="encryption">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>

MIIDEzCCAnygAwIBAgIJIAIzc4FYrIp9mMA0GCSqGSIb3DQEBBQUAMhcxCzAJBgNVBAYTA1VTMQswCQYDVQQIDA
JBWjEMMAoGA1UECgwDRERGMQwwCgYDVQQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBpkq
hkIG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMI
GDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWFrMQwwCgYDVQQKDANEREYxDDAK
BgNVBAsMA0RldjESMBAGA1UEAwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhXBsZS
5vcmcwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZhQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx
313A99s8WtA4mAeb6n0vTR9yNB0ekW4nY0iE0q//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7
+Pk1r0TDI4MRzPPkkTp4w1ORQ/a6CfvNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBvhvC
AQ0EGhYYRk9SIFRFU1RJTkcgvUFVSUE9TRSBPTkxZMB0GA1UDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrsDAfBg

```

```

NVHSMEGDAWgBThVMeX3wrCv6lfef47CyvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbgHvc3RAZXhhbXBsZS5vcmcw
DQYJKoZIhvcNAQEFBQADgYEAtRUpl7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx0
7n1Sm0ESp5sE3jGTnh0GtYV0D219z/09n90cd/imAEhknJlavyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZ
CfT9m07XduxjsmDz0h1SGV0=
    </ds:X509Certificate>
    </ds:X509Data>
    </ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="https://localhost:8993/logout"/>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:8993/logout"/>
<md:NameIDFormat>
    urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
</md:NameIDFormat>
<md:NameIDFormat>
    urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
</md:NameIDFormat>
<md:NameIDFormat>
    urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
</md:NameIDFormat>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="https://localhost:8993/services/idp/login"/>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:8993/services/idp/login"/>
</md:IDPSSODescriptor>
</md:EntityDescriptor>

```

Security IdP Server

An internal Identity Provider solution.

Configuring IdP Server

1. Navigate to Admin Console.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **IdP Server**.
5. Click the + next to SP Metadata to add a new entry
6. Populate the new entry:
 - a. with an HTTPS URL (<https://>),
 - b. file URL (file:), or
 - c. XML block to refer to desired metadata, e.g. (<https://localhost:8993/services/saml/sso/metadata>)

IdP Server example.xml

```

<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID=
"https://localhost:8993/services/saml">
  <md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIIDEzCCAAnygAwIBAgIJIAIzc4FYrIp9mMA0GCSqGSIB3DQEBBQUAMHcxCzAJBgNVBAYTA1VTMQswCQYDVQQIDA
JBWjEMMAoGA1UECgwDRERGMQwwCgYDVQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBpkq
hkiG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMI
GDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWfYMQwwCgYDVQQKDANEREYxDDAK
BgNVBAsMA0R1djESMBAGA1UEAwwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBfhVsB2NhbGhvc3RAZXhhbXBsZS
5vcmcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTphQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx
313A99s8WtA4mAeb6n0vTR9yNB0ekW4nY0iE0q//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7
+Pk1r0TDI4MRzPPkkTp4w1ORQ/a6CfVsNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBhvC
AQ0EGhYYRk9S1FRFU1RJTkcgvUFV9UE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BrerFrsDAfB
NVHSMEGDAwgbThVMeX3wrCv6lfeF47CvkSBe9xjAgBgnVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcw
DQYJKoZIhvcNAQEFBQADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx0
7n1Sm0ESp5sE3jGTnh0GtYV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZ
CfT9m07XduxjsmDz0h1SGV0=
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:KeyDescriptor use="encryption">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIIDEzCCAAnygAwIBAgIJIAIzc4FYrIp9mMA0GCSqGSIB3DQEBBQUAMHcxCzAJBgNVBAYTA1VTMQswCQYDVQQIDA
JBWjEMMAoGA1UECgwDRERGMQwwCgYDVQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBpkq
hkiG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMI
GDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWfYMQwwCgYDVQQKDANEREYxDDAK
BgNVBAsMA0R1djESMBAGA1UEAwwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBfhVsB2NhbGhvc3RAZXhhbXBsZS
5vcmcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTphQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx
313A99s8WtA4mAeb6n0vTR9yNB0ekW4nY0iE0q//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7
+Pk1r0TDI4MRzPPkkTp4w1ORQ/a6CfVsNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBhvC
AQ0EGhYYRk9S1FRFU1RJTkcgvUFV9UE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BrerFrsDAfB
NVHSMEGDAwgbThVMeX3wrCv6lfeF47CvkSBe9xjAgBgnVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcw
DQYJKoZIhvcNAQEFBQADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx0
7n1Sm0ESp5sE3jGTnh0GtYV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZ
CfT9m07XduxjsmDz0h1SGV0=
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://localhost:8993/logout"/>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"

```

```
Location="https://localhost:8993/logout"/>
<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://localhost:8993/services/saml/sso"/>
<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://localhost:8993/services/saml/sso"/>
</md:SPSSODescriptor>
</md:EntityDescriptor>
```

Configuring IdP Authentication Types

Set the authentication types that will be accepted by the IdP.

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **Web Context Policy Manager**
5. Under **Authentication Types**, set the *IDP authentication type* as necessary. Note that it should *only* be used on context paths that will be accessed by users via web browsers. For example:
 - /search=IDP

Other authentication types can also be used in conjunction with the IDP type. For example, if you wanted to secure the entire system with the IDP, but still allow legacy clients that don't understand the SAML ECP specification to connect, you could set */=IDP|PKI*. With that configuration, any clients that failed to connect using either the SAML 2.0 Web SSO Profile or the SAML ECP spec would fall back to 2-way TLS for authentication.

NOTE

If you have configured /search to use IDP, ensure to select the "External Authentication" checkbox in **Search UI** standard settings.

Identity Provider 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.

14.3.13. Catalog Filtering

Filtering is performed by an Access plugin, after a query or delete has been performed or before ingest has been performed.

How Filtering Works

Each metocard result can contain security attributes that are pulled from the metadata record after being processed by a **PolicyPlugin** that populates this attribute. The security attribute is a Map containing a set of keys that map to lists of values. The metocard is then processed by a filter plugin that creates a **KeyValueCollectionPermission** from the metocard's security attribute. This permission is then checked against the user subject to determine if the subject has the correct claims to view that metocard. The decision to filter the metocard eventually relies on the installed **Policy Decision**

Point (PDP). The PDP that is being used returns a decision, and the metocard will either be filtered or allowed to pass through.

How a metocard gets filtered is left up to any number of FilterStrategy implementations that might be installed. Each FilterStrategy will return a result to the filter plugin that says whether or not it was able to process the metocard, along with the metocard or response itself. This allows a metocard or entire response to be partially filtered to allow some data to pass back to the requester. This could also include filtering any products sent back to a requester.

The security attributes populated on the metocard are completely dependent on the type of the metocard. Each type of metocard must have its own **PolicyPlugin** that reads the metadata being returned and then returns the appropriate attributes.

Example (represented as simple XML for ease of understanding):

```
<metocard>
  <security>
    <map>
      <entry assertedAttribute1="A,B" />
      <entry assertedAttribute2="X,Y" />
      <entry assertedAttribute3="USA,GBR" />
      <entry assertedAttribute4="USA,AUS" />
    </map>
  </security>
</metocard>
```

```
<user>
  <claim name="subjectAttribute1">
    <value>A</value>
    <value>B</value>
  </claim>
  <claim name="subjectAttribute2">
    <value>X</value>
    <value>Y</value>
  </claim>
  <claim name="subjectAttribute3">
    <value>USA</value>
  </claim>
  <claim name="subjectAttribute4">
    <value>USA</value>
  </claim>
</user>
```

In the above example, the user's claims are represented very simply and are similar to how they would actually appear in a SAML 2 assertion. Each of these user (or subject) claims will be converted to a **KeyValuePermission** object. These permission objects will be implied against the permission object generated from the metocard record. In this particular case, the metocard might be allowed if the policy is configured appropriately because all of the permissions line up correctly.

Configuring Filtering Policies

There are two options for processing filtering policies: internally, or through the use of a policy formatted in eXtensible Access Control Markup Language (XACML). The procedure for setting up a policy differs depending on whether that policy is to be used internally or by the external XACML processing engine. Setting up an internal policy is as follows:

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Click the **Configuration** tab.
4. Click on the **Security AuthZ Realm** configuration.
5. Add any attribute mappings necessary to map between subject attributes and the attributes to be asserted.
 - a. For example, the above example would require two Match All mappings of `subjectAttribute1=assertedAttribute1` and `subjectAttribute2=assertedAttribute2``
 - b. Match One mappings would contain `subjectAttribute3=assertedAttribute3`` and `subjectAttribute4=assertedAttribute4.`

With the `security-pdp-authz` feature configured in this way, the above Metocard would be displayed to the user. Note that this particular configuration would not require any XACML rules to be present. All of the attributes can be matched internally and there is no reason to call out to the external XACML processing engine. For more complex decisions, it might be necessary to write a XACML policy to handle certain attributes.

To set up a XACML policy, place the desired XACML policy in the `<distribution root>/etc/pdp/policies` directory and update the included `access-policy.xml` to include the new policy. This is the directory in which the PDP will look for XACML policies every 60 seconds. A sample XACML policy is located at the end of this page. Information on specific bundle configurations and names can be found on the Security PDP application page.

Catalog Filter Policy Plugins

Several Policy Plugins for catalog filtering exist currently: [Metocard Attribute Security Policy Plugin](#) and [XML Attribute Security Policy Plugin](#). These Policy Plugin implementations allow an administrator to easily add filtering capabilities to some standard Metocard types for all Catalog operations. These plugins will place policy information on the Metocard itself that allows the Filter Plugin to restrict unauthorized users from viewing content they are not allowed to view.

Creating a XACML Policy

This document assumes familiarity with the XACML schema and does not go into detail on the XACML language. When creating a policy, a target is used to indicate that a certain action should be run only for one type of request. Targets can be used on both the main policy element and any individual rules. Targets are geared toward the actions that are set in the request. These actions generally consist of the standard CRUD operations (create, read, update, delete) or a SOAPAction if the request is coming through a SOAP endpoint.

NOTE

These are only the action values that are currently created by the components that come with DDF. Additional components can be created and added to DDF to identify specific actions.

In the examples below, the policy has specified targets for the above type of calls. For the Filtering code, the target was set for "filter", and the Service validation code targets were geared toward two services: `query` and `LocalSiteName`. In a production environment, these actions for service authorization will generally be full URNs that are described within the SOAP WSDL.

XACML Policy Attributes

Attributes for the XACML request are populated with the information in the calling subject and the resource being checked.

XACML Policy Subject

The attributes for the subject are obtained from the SAML claims and populated within the XACML policy as individual attributes under the `urn:oasis:names:tc:xacml:1.0:subject-category:access-subject` category. The name of the claim is used for the `AttributeId` value. Examples of the items being populated are available at the end of this page.

XACML Policy Resource

The attributes for resources are obtained through the permissions process. When checking permissions, the XACML processing engine retrieves a list of permissions that should be checked against the subject. These permissions are populated outside of the engine and should be populated with the attributes that should be asserted against the subject. When the permissions are of a key-value type, the key being used is populated as the `AttributeId` value under the `urn:oasis:names:tc:xacml:3.0:attribute-category:resource` category.

Using a XACML Policy

To use a XACML policy, copy the XACML policy into the `<DDF_HOME>/etc/pdp/policies` directory.

14.3.14. Auditing

- **Required Step for Security Hardening**

Audit logging captures security-specific system events for monitoring and review. DDF provides a [Audit Plugin](#) that logs all catalog transactions to the `security.log`. Information captured includes user identity, query information, and resources retrieved.

Follow all operational requirements for the retention of the log files. This may include using cryptographic mechanisms, such as encrypted file volumes or databases, to protect the integrity of audit information.

NOTE

The Audit Log default location is `<DDF_HOME>/data/log/security.log`

Audit Logging Best Practices

For the most reliable audit trail, it is recommended to configure the operational environment of the DDF to generate alerts to notify administrators of:

NOTE

- auditing software/hardware errors
- failures in audit capturing mechanisms
- audit storage capacity (or desired percentage threshold) being reached or exceeded.

WARNING

The security audit logging function does not have any configuration for audit reduction or report generation. The logs themselves could be used to generate such reports outside the scope of DDF.

Enabling Fallback Audit Logging

• Required Step for Security Hardening

In the event the system is unable to write to the `security.log` file, DDF must be configured to fall back to report the error in the application log:

- edit `<INSTALL_HOME>/etc/org.ops4j.pax.logging.cfg`
 - uncomment the line (remove the `#` from the beginning of the line) for `log4j2` (`org.ops4j.pax.logging.log4j2.config.file = ${karaf.etc}/log4j2.config.xml`)
 - delete all subsequent lines
- edit `<INSTALL_HOME>/etc/startup.properties`
 - replace the artifact `pax-logging-service` with the artifact `pax-logging-log4j2` (keep same version and group)
- edit `<INSTALL_HOME>/etc/log4j2.config.xml`
 - find the entry for the `securityBackup` appender. (see example)
 - change value of `filename` and prefix of `filePattern` to the name/path of the desired failover security logs (`<NEW_FILE_NAME>`)

securityBackup Appender Before

```
<RollingFile name="securityBackup" append="true" ignoreExceptions="false"
              fileName="${sys:karaf.data}/log/securityBackup.log"
              filePattern="${sys:karaf.data}/log/securityBackup.log-%d{yyyy-MM-
dd-HH}-%i.log.gz">
```

securityBackup Appender After

```
<RollingFile name="securityBackup" append="true" ignoreExceptions="false"
    fileName="${sys:karaf.data}/log/<NEW_FILE_NAME>" 
    filePattern="${sys:karaf.data}/log/<NEW_FILE_NAME>-%d{yyyy-MM-dd}
-HH}-%i.log.gz">
```

14.3.15. Configuring the Landing Page

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

Installing the Landing Page

The Landing Page is installed by default with a standard installation.

Customizing the Landing Page

Configure the Landing Page from the Admin Console:

1. Navigate to the **Admin Console**.
2. Select **Platform** Application.
3. Select **Configuration** tab.
4. Select **Landing Page**.

Table 7. Landing Page

Name	ID	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.	true
Phone Number	phone	String	Specifies the phone number to display on the landing page.		true
Email Address	email	String	Specifies the email address to display on the landing page.		true
External Web Site	externalUrl	String	Specifies the external web site URL to display on the landing page.		true

Name	Id	Type	Description	Default Value	Required
Announcements	announcements	String	Announcements that will be displayed on the landing page.	null	true
Branding Background	background	String	Specifies the landing page background color. Use html css colors or #rrggbb.		true
Branding Foreground	foreground	String	Specifies the landing page foreground color. Use html css colors or #rrggbb.		true
Branding Logo	logo	String	Specifies the landing page logo. Use a base64 encoded image.		true
Additional Links	links	String	Additional links to be displayed on the landing page. Use the format <text>,<link> (e.g. example, http://www.example.com). Empty entries are ignored.		yes

14.4. Configuring from the Command Console

In environments where access to the Admin Console is not possible or not desired, configurations can also be performed through a command line interface, the Command Console.

14.4.1. Managing Applications From the Command Console

Applications can be installed from the Command Console using the following commands:

Table 8. App Commands

Command	Effect
app:add <appName>	Install an app.
app:list	List all installed apps and current status.
app:remove <appName>	Uninstall an app.
app:start	Start an inactive app.
app:status <appName>	Detailed view of application status
app:stop <appName>	Stop an active app.
app:tree	Dependency tree view of all installed apps.

14.4.2. Managing Features From the Command Console

Individual features can also be added via the Command Console.

1. Determine which feature to install by viewing the available features on DDF.

```
ddf@local>feature:list
```

2. The console outputs a list of all features available (installed and uninstalled). A snippet of the list output is shown below (the versions may differ):

State Description	Version	Name	Repository
[installed] [2.10.3] security-handler-api app-2.10.3 API for authentication handlers for web applications.			security-services-
[installed] [2.10.3] security-core app-2.10.3 DDF Security Core			security-services-
[uninstalled] [2.10.3] security-expansion app-2.10.3 DDF Security Expansion			security-services-
[uninstalled] [2.10.3] security-cas-client app-2.10.3 DDF Security CAS Client.			security-services-
[uninstalled] [2.10.3] security-cas-tokenvalidator app-2.10.3 DDF Security CAS Validator for the STS.			security-services-
[uninstalled] [2.10.3] security-cas-cxfservletfilter app-2.10.3 DDF Security CAS Servlet Filter for CXF.			security-services-
[installed] [2.10.3] security-pdp-authz app-2.10.3 DDF Security PDP.			security-services-
[uninstalled] [2.10.3] security-pep-serviceauthz app-2.10.3 DDF Security PEP Service AuthZ			security-services-
[uninstalled] [2.10.3] security-expansion-user-attributes app-2.10.3 DDF Security Expansion User Attributes Expansion			security-services-
[uninstalled] [2.10.3] security-expansion-metocard-attributes app-2.10.3 DDF Security Expansion Metocard Attributes Expansion			security-services-
[installed] [2.10.3] security-sts-server app-2.10.3 DDF Security STS.			security-services-
[installed] [2.10.3] security-sts-realm app-2.10.3 DDF Security STS Realm.			security-services-
[uninstalled] [2.10.3] security-sts-ldaplogin app-2.10.3 DDF Security STS JAAS LDAP Login.			security-services-
[uninstalled] [2.10.3] security-sts-ldapclaimshandler app-2.10.3 Retrieves claims attributes from an LDAP store.			security-services-

1. Check the bundle status to verify the service is started.

```
ddf@local>list
```

The console output should show an entry similar to the following:

```
[ 117] [Active ] [          ] [Started] [ 75] DDF :: Catalog :: Source ::  
Dummy (<version>)
```

Uninstalling Features from the Command Console

1. Check the feature list to verify the feature is installed properly.

```
ddf@local>feature:list
```

State	Version	Name	Repository
Description			
[installed]	[2.10.3]] ddf-core	ddf-2.10.3
[uninstalled]	[2.10.3]] ddf-sts	ddf-2.10.3
[installed]	[2.10.3]] ddf-security-common	ddf-2.10.3
[installed]	[2.10.3]] ddf-resource-impl	ddf-2.10.3
[installed]	[2.10.3]] ddf-source-dummy	ddf-2.10.3

1. Uninstall the feature.

```
ddf@local>feature:uninstall ddf-source-dummy
```

WARNING Dependencies that were auto-installed by the feature are not automatically uninstalled.

1. Verify that the feature has uninstalled properly.

```
ddf@local>feature:list
```

State	Version	Name	Repository	Description
[installed]	[2.10.3]] ddf-core	ddf-2.10.3	
[uninstalled]	[2.10.3]] ddf-sts	ddf-2.10.3	
[installed]	[2.10.3]] ddf-security-common	ddf-2.10.3	
[installed]	[2.10.3]] ddf-resource-impl	ddf-2.10.3	
[uninstalled]	[2.10.3]] ddf-source-dummy	ddf-2.10.3	

14.4.3. Configuring HTTP to HTTPS Proxy From the Command Console

DDF includes a proxy to transform HTTP to HTTPS for systems unable to use HTTPS.

- Type the command `feature:install platform-http-proxy`

14.4.4. Configuring Solr from Command Console

Solr can only be configured/installed from configuration files. See [Solr System Properties](#) and [Configuring Solr Catalog Provider Data Directory](#).

14.4.5. Standalone Security Token Service (STS) Installation

To run a STS-only DDF installation, uninstall the catalog components that are not being used. The following list displays the features that can be uninstalled to minimize the runtime size of DDF in an STS-only mode. This list is not a comprehensive list of every feature that can be uninstalled; it is a list of the larger components that can be uninstalled without impacting the STS functionality.

Unnecessary Features for Standalone STS

- `catalog-core-standardframework`
- `catalog-opensearch-endpoint`
- `catalog-opensearch-souce`

- catalog-rest-endpoint

14.4.6. Hardening Solr Index

• Required Step for Security Hardening

The DDF's design includes support for pluggable indexes. The default installation contains a Solr index to be used as the Metadata Catalog. If desired, this implementation can be replaced with an alternate 3rd party index implementation.

The following sections provide hardening guidance for Solr; however, they are to serve only as reference other an additional security requirements may be added.

Solr Admin User Interface Security

The Solr Admin user interface uses basic authentication as part of the server configuration.

Configuring Solr Node Security

The Solr server is protected by the built in REST security architecture. The configuration can be changed by editing the Web Context Policy Manager configuration for the `/solr` web context.

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **Web Context Policy Manager**.

By default, the configuration is set to `/solr=SAML|PKI|BASIC`. This allows a user or another system to connect to Solr using any of those authentication methods.

Configuring Solr Encryption

While it is possible to encrypt the Solr index, it decreases performance significantly. An encrypted Solr index also can only perform exact match queries, not relative or contextual queries. As this drastically reduces the usefulness of the index, this configuration is not recommended. The recommended approach is to encrypt the entire drive through the Operating System of the server on which the index is located.

14.5. Configuring from Configuration Files

As most configurations are stored in configuration files, in some instances it may make sense to edit those configuration files directly. Additionally, configuration files may be pre-created and copied into a DDF installation. Finally, in an environment hardened for security purposes, access to the Admin Console or the Command Console might be denied and using the latter in such an environment may cause configuration errors. It is necessary to configure DDF (e.g., providers, Schematron rulesets, etc.) using `.config` files.

14.5.1. Configuring Global Settings with system.properties

Global configuration settings are configured via the properties file `system.properties`. These properties can be manually set by editing this file or set via the initial configuration from the Admin Console.

NOTE Any changes made to this file require a restart of the system to take effect.

IMPORTANT The passwords configured in this section reflect the passwords used to decrypt JKS (Java KeyStore) files. Changing these values without also changing the passwords of the JKS causes undesirable behavior.

Table 9. Global Settings

Title	Property	Type	Description	Default Value	Required
Keystore and truststore java properties					
Keystore	<code>javax.net.ssl.keyStore</code>	String	Path to server keystore	<code>etc/keystores/serverKeystore.jks</code>	Yes
Keystore Password	<code>javax.net.ssl.keyStorePassword</code>	String	Password for accessing keystore	<code>changeit</code>	Yes
Truststore	<code>javax.net.ssl.trustStore</code>	String	The trust store used for SSL/TLS connections. Path is relative to <code><DDF_HOME></code> .	<code>etc/keystores/serverTruststore.jks</code>	Yes
Truststore Password	<code>javax.net.ssl.trustStorePassword</code>	String	Password for server Truststore	<code>changeit</code>	Yes
Keystore Type	<code>javax.net.ssl.keyStoreType</code>	String	File extension to use with server keystore	<code>jks</code>	Yes
Truststore Type	<code>javax.net.ssl.trustStoreType</code>	String	File extension to use with server truststore	<code>jks</code>	Yes
Headless mode					
Headless Mode	<code>java.awt.headless</code>	Boolean	Force java to run in headless mode for when the server doesn't have a display device	<code>true</code>	No
Global URL Properties					

Title	Property	Type	Description	Default Value	Required
Default Protocol	<code>org.codice.ddf.system.protocol</code>	String	Default protocol that should be used to connect to this machine.	<code>https://</code>	Yes
Host	<code>org.codice.ddf.system.hostname</code>	String	The hostname or IP address used to advertise the system. Do not enter <code>localhost</code> . Possibilities include the address of a single node or that of a load balancer in a multi-node deployment. If the hostname is changed during the install to something other than <code>localhost</code> a new keystore and truststore must be provided. See Managing Keystores and Certificates for details. NOTE: Does not change the address the system runs on.	<code>localhost</code>	Yes
HTTPS Port	<code>org.codice.ddf.system.httpsPort</code>	String	The https port used by the system. NOTE: This DOES change the port the system runs on.	<code>8993</code>	Yes
HTTP Port	<code>org.codice.ddf.system.httpPort</code>	String	The http port used by the system. NOTE: This DOES change the port the system runs on.	<code>8181</code>	Yes
Default Port	<code>org.codice.ddf.system.port</code>	String	The default port used to advertise the system. This should match either the http or https port. NOTE: Does not change the port the system runs on.	<code>8993</code>	Yes

Title	Property	Type	Description	Default Value	Required
Root Context	<code>org.codice.ddf.system.rootContext</code>	String	The the base or root context that services will be made available under.	<code>/services</code>	Yes

System Information Properties

Site Name	<code>org.codice.ddf.system.siteName</code>	String	The site name for DDF.	<code>ddf.distribution</code>	Yes
Site Contact	<code>org.codice.ddf.system.siteContact</code>	String	The email address of the site contact.		No
Version	<code>org.codice.ddf.system.version</code>	String	The version of DDF that is running. This value should not be changed from the factory default.	<code>2.10.3</code>	Yes
Organization	<code>org.codice.ddf.system.organization</code>	String	The organization responsible for this installation of DDF.	<code>Codice Foundation</code>	Yes

Thread Pool Settings

Thread Pool Size	<code>org.codice.ddf.system.threadPoolSize</code>	Integer	Size of thread pool used for handling UI queries, federating requests, and downloading resources. See Configuring Thread Pools	<code>128</code>	Yes
------------------	---	---------	--	------------------	-----

HTTPS Specific Settings

Cipher Suites	<code>https.cipherSuites</code>	String	Cipher suites to use with secure sockets. If using the JCE unlimited strength policy, use this list in place of the defaults: .	<code>TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</code>	No
Https Protocols	<code>https.protocols</code>	String	Protocols to allow for secure connections	<code>TLSv1.1,TLSv1.2</code>	No

Title	Property	Type	Description	Default Value	Required
Allow Basic Auth Over Http	<code>org.codice.allowBasicAuthOverHttp</code>	Boolean	Set to true to allow Basic Auth credentials to be sent over HTTP unsecurely. This should only be done in a test environment. These events will be audited.	<code>false</code>	Yes
Restrict the Security Token Service to allow connections only from DNs matching these patterns	<code>ws-security.subject.cert.constraints</code>	String	Set to a comma separated list of regex patterns to define which hosts are allowed to connect to the STS	<code>.*</code>	Yes

XML Settings

Parse XML documents into DOM object trees	<code>javax.xml.parsers.DocumentBuilderFactory</code>	String	Enables Xerces-J implementation of <code>DocumentBuilderFactory</code>	<code>org.apache.xerces.jaxp.DocumentBuilderFactoryImpl</code>	Yes
---	---	--------	--	--	-----

Catalog Source Retry Interval

Initial Endpoint Contact Interval	<code>org.codice.ddf.platform.util.http.initialRetryInterval</code>	Integer	If a Catalog Source is unavailable, try to connect to it after the initial interval has elapsed. After every retry, the interval doubles, up to a given maximum interval. The interval is measured in seconds.	<code>10</code>	Yes
-----------------------------------	---	---------	--	-----------------	-----

Title	Property	Type	Description	Default Value	Required
Maximum Endpoint Contact Interval	Maximum seconds between attempts to establish contact with unavailable Catalog Source.	Integer	Do not wait longer than the maximum interval to attempt to establish a connection with an unavailable Catalog Source. Smaller values result in more current information about the status of Catalog Sources, but cause more network traffic. The interval is measured in seconds.	300	Yes

File Upload Settings

File extensions flagged as potentially dangerous to the host system or external clients	<code>bad.file.extensions</code>	String	Files uploaded with these bad file extensions will have their file names sanitized before being saved	<code>.exe, .jsp, .html, .js, .php, .phtml, .php3, .php4, .php5, .phps, .shtml, .jhtml, .pl, .py, .cgi, .msi, .com, .scr, .gadget, .application, .pif, .hta, .cpl, .msc, .jar, .kar, .bat, .cmd, .vb, .vbs, .vbe, .jse, .ws, .wsf, .wsc, .wsh, .ps1, .ps1xml, .ps2, .ps2xml, .psc1, .psc2, .msh, .msh1, .msh2, .mshxml, .msh1xml, .msh2xml, .scf, .lnk, .inf, .reg, .dll, .vxd, .cpl, .cfg, .config, .crt, .cert, .pem, .jks, .p12, .p7b, .key, .der, .csr, .jsb, .mhtml, .mht, .xhtml, .xht</code>	Yes
---	----------------------------------	--------	---	---	-----

Title	Property	Type	Description	Default Value	Required
File names flagged as potentially dangerous to the host system or external clients	<code>bad.files</code>	String	Files uploaded with these bad file names will have their file names sanitized before being saved	<code>crossdomain.xml, clientaccesspolicy.xml, .htaccess, .htpasswd, hosts, passwd, group, resolv.conf, nfs.conf, ftpd.conf, ntp.conf, web.config, robots.txt</code>	Yes
Mime types flagged as potentially dangerous to external clients	<code>bad.mime.types</code>	String	Files uploaded with these mime types will be rejected from the upload	<code>text/html, text/javascript, text/x-javascript, application/x-shellscript, text/scriptlet, application/x-msdownload, application/x-msmetafile</code>	Yes

These properties are available to be used as variable parameters in input url fields within the Admin Console. For example, the url for the local csw service (<https://localhost:8993/services/csw>) could be defined as:

```
 ${org.codice.ddf.system.protocol}${org.codice.ddf.system.hostname}:${org.codice.ddf.system.port}${org.codice.ddf.system.rootContext}/csw
```

This variable version is more verbose, but will not need to be changed if the system `host`, `port` or `root` context changes.

WARNING Only root can access ports < 1024 on Unix systems.

14.5.2. Configuring with .config Files

The DDF is configured using `.config` files. Like the Karaf `.cfg` files, these configuration files must be located in the `<DDF_HOME>/etc/` directory, have a name that matches the *configuration persistence ID* (PID) they represent, and have a `service.pid` property set to the configuration PID.

As opposed to `.cfg` however, this type of configuration file supports lists within configuration values (metatype `cardinality` attribute greater than 1).

IMPORTANT

This new configuration file format **must** be used for any configuration that makes use of lists. Examples include Web Context Policy Manager (PID: `org.codice.ddf.security.policy.context.impl.PolicyManager`) and Security STS Guest Claims Handler (PID: `ddf.security.sts.guestclaims`).

WARNING

Only one configuration file should exist for any given PID. The result of having both a `.cfg` and a `.config` file for the same PID is undefined and could cause the application to fail.

The main purpose of the configuration files is to allow administrators to pre-configure DDF without having to use the Admin Console. In order to do so, the configuration files need to be copied to the `<DDF_HOME>/etc` directory after DDF zip has been extracted.

Upon start up, all the `.config` files located in `<DDF_HOME>/etc` are automatically read and processed. Files that have been processed successfully are moved to `<DDF_HOME>/etc/processed` so they will not be processed again when the system is restarted. Files that could not be processed are moved to the `<DDF_HOME>/etc/failed` directory.

DDF also monitors the `<DDF_HOME>/etc` directory for any new `.config` file that gets added. As soon as a new file is detected, it is read, processed and moved to the appropriate directory based on whether it was successfully processed or not.

14.5.3. Configuring Using a `.config` File Template

A template file is provided for some configurable DDF items so that they can be copied/renamed then modified with the appropriate settings.

The following steps define the procedure for configuring a new source or feature using a `config` file:

1. Copy/rename the provided template file in the `'etc/templates'` directory to the `etc` directory. (Refer to the table above to determine correct template.)
 - a. Not required, but a good practice is to change the instance name (e.g., `OpenSearchSource.1.config`) of the file to something identifiable (`OpenSearchSource.remote-site-1.config`).
2. Edit the copied file to etc with the settings for the configuration. (Refer to the table above to determine the configurable properties).
 - a. Consult the inline comments in the file for guidance on what to modify.

The new service can now be used as if it was created using the Admin Console.

Table 10. Templates included with DDF

DDF Service	Template File Name	Factory PID	Configurable Properties
DDF Catalog Framework	<code>ddf.catalog.impl.service.CatalogFrameWorkImpl.cfg</code>	<code>ddf.catalog.CatalogFrameworkImpl</code>	Standard Catalog Framework

14.5.4. Configuring WSS Using Standalone Servers

DDF can be configured to use SAML 2.0 Web SSO as a single sign-on service and LDAP and STS to keep track of users and user attributes. SAML, LDAP, and STS can be installed on a local DDF instance with only a few feature installs. Setting up these authentication components to run externally, however, is more nuanced, so this page will provide step-by-step instructions detailing the configuration process.

If using different keystore names, substitute the name provided in this document with the desired name for your setup. For this document, the following data is used:

Server	Keystore File	Comments
DDF	<code>serverKeystore.jks</code>	Keystore used for SSL/TLS connections.

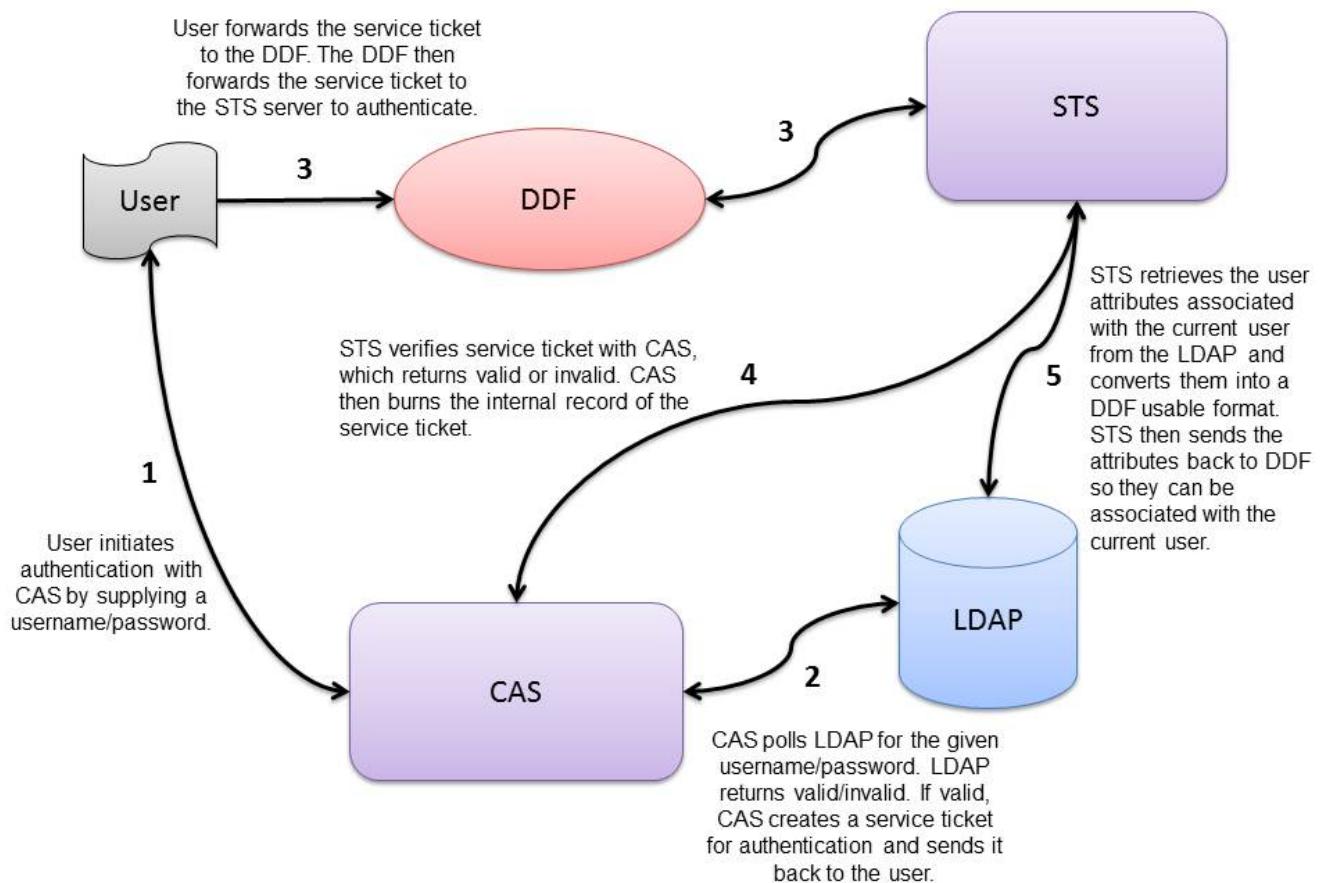


Figure 1. Login Authentication Scheme

14.5.5. Configuring Managed Service Factory Bundles

Services that are created using a Managed Service Factory can be configured using `.config` files as well. These configuration files, however, follow a different naming convention. The files must start with the Managed Service Factory PID, be followed by a unique identifier and have a `.config` extension. For instance, assuming that the Managed Service Factory PID is `org.codice.ddf.factory.pid` and two instances of the service need to be configured, files `org.codice.ddf.factory.pid.<UNIQUE_ID_1>.config` and `org.codice.ddf.factory.pid.<UNIQUE_ID_2>.config`.

2>.config should be created and added to <DDF_HOME>/etc.

The unique identifiers used in the file names have no impact on the order in which the configuration files are processed. No specific processing order should be assumed. Also, a new service will be created and configured every time a configuration file matching the Managed Service Factory PID is added to the directory, regardless of the *unique id* used.

These configuration files must also contain a `service.factoryPid` property set to the factory PID (without the sequential number). They should not however contain the `service.pid` property.

File Format

The basic syntax of the .config configuration files is similar to the older .cfg files but introduces support for lists and types other than simple strings. The type associated with a property must match the type attribute used in the corresponding metatype.xml file when applicable.

The following table shows the format to use for each property type supported.

Table 11. Property Formats

Type	Format	Example
Service PID	<code>service.pid = "servicePid"</code>	<code>service.pid = "org.codice.ddf.security.policy.context.impl.PolicyManager"</code>
Factory PID	<code>service.factoryPid = "serviceFactoryPid"</code>	<code>service.factoryPid = "Csw_Federated_Source"</code>
Strings	<code>name = "value"</code>	<code>name = "john"</code>
Booleans	<code>name = B"true false"</code>	<code>authorized = B"true"</code>
Integers	<code>name = I"value"</code>	<code>timeout=I"60"</code>
Longs	<code>name = L"value"</code>	<code>diameter = L"10000"</code>
FLOATS	<code>name = F"value"</code>	<code>cost = F"10.50"</code>
Doubles	<code>name = D"value"</code>	<code>latitude = D"45.0234"</code>
Lists of Strings	<code>name = ["value1", "value2", ...]</code>	<code>complexStringArray = [{"url": "\\\\http://test.sample.com\\\\layers\\\\[\"0\"]\\\\VERSION\\\\\"1.1 1.2\\\\\"\\\\image/png\\\\\"}\\\\beta\\\\\"1"}, {"url": "\\\\http://test.sample.com\\\\0.5"}, "/solr\\\\=SAML PKI basic", "/security-config\\\\=SAML basic"]</code>
Lists of Integers	<code>name = I["value1", "value1", ...]</code>	<code>sizes = I["10", "20", "30"]</code>

NOTE

- Lists of values can be prefixed with any of the supported types (B, I, L, F or D).
- To prevent any configuration issues, the = signs used in values should be escaped using a backslash (\).
- Boolean values will default to false if any value other than true is provided.
- Escape character in values must be used for double quotes ("") and spaces, but cannot be used with { } or [] pairings.

Sample configuration file

```
service.pid="org.codice.ddf.security.policy.context.impl.PolicyManager"

authenticationTypes=[ "/\=SAML|GUEST", "/admin\=SAML|basic", "/system\=basic", "/solr\=SAM
L|PKI|basic", "/sources\=SAML|basic", "/security-config\=SAML|basic", "/search\=basic" ]

realms=[ "/\=karaf" ]

requiredAttributes=[ "/\=", "/admin\={http://schemas.xmlsoap.org/ws/2005/05/identity/cla
ims/role\=admin}", "/solr\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\
=admin}", "/system\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\=admin}
", "/security-config\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role
\=admin}" ]

whiteListContexts=[ "/services/SecurityTokenService", "/services/internal/metrics", "/ser
vices/saml", "/proxy", "/services/csw" ]
```

14.5.6. Installing Multiple DDFs on the Same Host

To have multiple DDF instances on the same host, it is necessary to edit the port numbers in the files in the DDF install folder.

File to Edit	Property(ies)	Original Value	Example of New Value
bin/karaf.bat	address	5005	5006
etc/org.apache.karaf.m anagement.cfg	rmiRegistryPort	1099	1199
	rmiServerPort	44444	44445
etc/system.properties	httpsPort, port	8993	8994
	httpPort	8181	8281

14.5.7. Configuring Files in Home Directory Hierarchy

Many important configuration settings are stored in the <DDF_HOME> directory.

NOTE

Depending on the environment, it may be easier for integrators and administrators to configure DDF using the Admin Console prior to disabling it for hardening purposes. The Admin Console can be re-enabled for additional configuration changes.

In an environment hardened for security purposes, access to the Admin Console or the Command Console might be denied and using the latter in such an environment may cause configuration errors. It is necessary to configure DDF (e.g., providers, Schematron rulesets, etc.) using `.config` files.

A template file is provided for some configurable DDF items so that they can be copied/renamed then modified with the appropriate settings.

WARNING

If the Admin Console is enabled again, all of the configuration done via `.config` files will be loaded and displayed. However, note that the name of the `.config` file is not used in the Admin Console. Rather, a universally unique identifier (UUID) is added when the DDF item was created and displays this UUID in the console (e.g., `OpenSearchSource.112f298e-26a5-4094-befc-79728f216b9b`)

14.5.8. Configuring Solr Catalog Provider Data Directory

The HTTP Solr Catalog Provider and the Embedded Solr Catalog Provider writes index files to the file system. By default, these files are stored under `DDF_HOME/data/solr/catalog/data`. If there is inadequate space in `DDF_HOME`, or if it is desired to maintain backups of the indexes only, this directory can be changed.

In order to change the Data Directory, the `system.properties` file in `DDF_HOME/etc` must be edited prior to starting DDF.

Edit the `system.properties` file

```
# Uncomment the following line and set it to the desired path  
# solr.data.dir = ${karaf.home}/data/solr
```

Changing the Data Directory

It may become necessary to change the data directory after DDF has ingested data.

1. Shut down the DDF.
2. Create the new directory to hold the indexes.

Make new Data Directory

```
mkdir -p /path/to/new/data/dir
```

3. Copy the indexes to the new directory.

Copy the indexes to the new Directory.

```
cp /path/to/old/data/dir/* /path/to/new/data/dir/.
```

4. Set the `system.properties` file to use the new directory.

Update system.properties file

```
solr.data.dir = /path/to/new/data/dir
```

5. Restart the DDF.

Changes Require a Distribution Restart

WARNING

If the Data Directory File Path property is changed, no changes will occur to the Solr Catalog Provider until the distribution has been restarted.

NOTE

If data directory file path property is changed to a new directory, and the previous data is not moved into that directory, no data will exist in Solr. Instead, Solr will create an empty index. Therefore, it is possible to have multiple places where Solr files are stored, and a user can toggle between those locations for different sets of data.

14.5.9. Configuring Thread Pools

The `org.codice.ddf.system.threadPoolSize` property can be used to specify the size of thread pools used by:

- Federating requests between DDF systems
- Downloading resources
- Handling asynchronous queries, such as queries from the UI

By default, this value is set to 128. It is not recommended to set this value extremely high. If unsure, leave this setting at its default value of 128.

14.5.10. Configuring Web Service Providers

By default Solr, STS server, STS client and the rest of the services use the system property `org.codice.ddf.system.hostname` which is defaulted to 'localhost' and not to the fully qualified domain name of the DDF instance. Assuming the DDF instance is providing these services, the configuration must be updated to use the **fully qualified domain name** as the service provider.

This can be changed during [Initial Configuration](#) or later by editing the `<INSTALL_HOME>/etc/system.properties` file.

14.5.11. Isolating Solr Cloud and Zookeeper

- **Required Step for Security Hardening** (if using Solr Cloud/Zookeeper)

Zookeeper cannot use secure (SSL/TLS) connection. The configuration information that Zookeeper sends and receives is vulnerable to network sniffing. Also, the connections between the local Solr Catalog service and the Solr Cloud is not necessarily secure. The connections between Solr Cloud nodes are not necessarily secure. Any unencrypted network traffic is vulnerable to sniffing attacks. To use Solr Cloud and Zookeeper securely, these processes must be isolated on the network, or their

communications must be encrypted by other means. The DDF process must be visible on the network to allow authorized parties to interact with it.

Examples of Isolation:

- Create a private network for Solr Cloud and Zookeeper. Only DDF is allowed to contact devices inside the private network.
- Use IPsec to encrypt the connections between DDF, Solr Cloud nodes, and Zookeeper nodes.
- Put DDF, Solr Cloud and Zookeeper behind a firewall that only allows access to DDF.

14.6. Importing Configurations

The Configuration Export/Import capability allows administrators to export the current DDF configuration and use it as a starting point for a new installation. This is useful when upgrading or expanding use of DDF where an identical configuration of multiple instances is desired.

IMPORTANT

- Importing configuration files is only guaranteed to work when importing files from the same DDF version. Importing from a different version is not recommended as it may cause the new DDF instance to be incorrectly configured and become unusable.
- All configurations will be exported to `<DDF_HOME>/etc/exported` followed by their relative path from <DDF_HOME>. For instance, <DDF_HOME>/etc/keystores/keystore.jks will be exported to <DDF_HOME>/etc/exported/etc/keystores/keystore.jks, while <DDF_HOME>/etc/system.properties will be exported to <DDF_HOME>/etc/exported/etc/system.properties.
- To keep the export/import process simple and consistent, all system configuration files are required to be under the <DDF_HOME> directory.

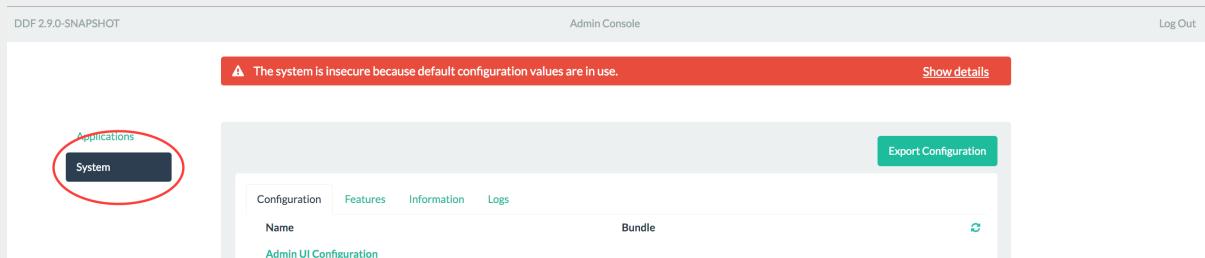
14.6.1. Exporting Existing Configurations

Exporting Existing Configurations from Admin Console

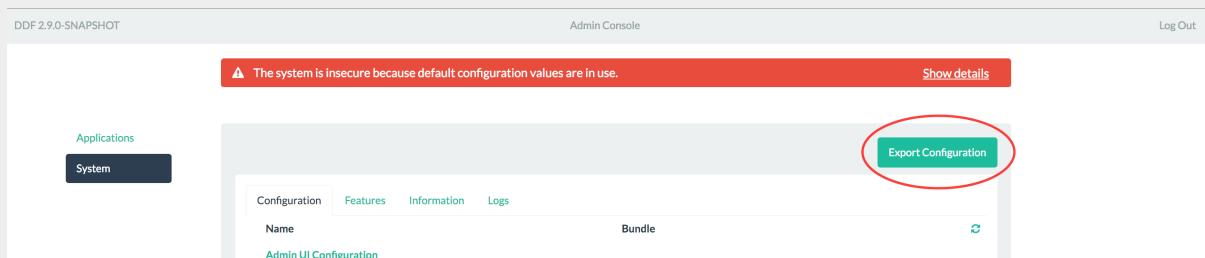
You can export the current system configurations using the Admin Console. This is useful for migrating from one running instance to another.

To export the current system configurations, follow these instructions:

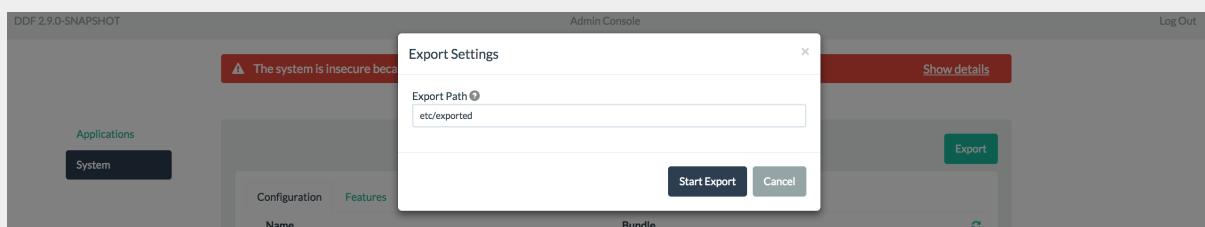
1. Select the **System** tab (next to the **Applications** tab)



2. Click the **Export Configuration** button



3. Fill out the form, specifying the *destination* for the export. A relative path will be relative to <DDF_HOME>.



4. Click the **Start Export** button.

5. If there are no warnings or errors, the form will automatically close upon finishing the export.

Export Existing Configuration Settings from Command Console

- Required Step for Security Hardening

To export the current DDF configuration from the Command Console:

1. Type in `migration:export <directory>`. This command creates the exported configuration files that are saved to the specified directory. If no directory is specified it will default to `<DDF_HOME>/etc/exported`
2. Zip up the exported files in the export directory.

```
cd <DDF_HOME>/etc/exported  
zip -r exportedFiles.zip *
```

Troubleshooting Common Warnings or Failures of Configuration Export

If export is unsuccessful, use this list to verify the correct configuration.

- Export Destination Directory Permissions Set to Read Only.

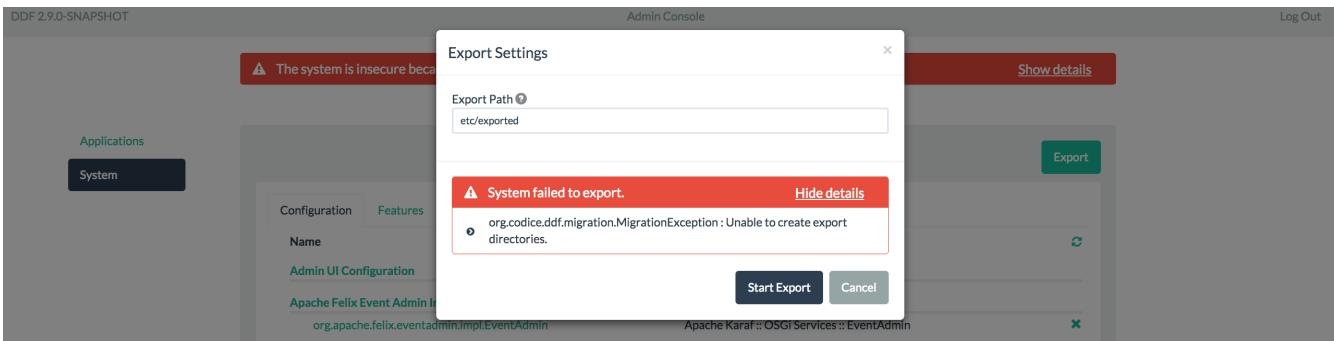


Figure 2. Insufficient Write Permissions

- Properties Set to Absolute File Paths
 - Setting properties to absolute paths is not allowed; so update the property to a value that is relative to `<DDF_HOME>`. However, notice that the export did not completely fail, but issued a warning that a specific file was excluded.

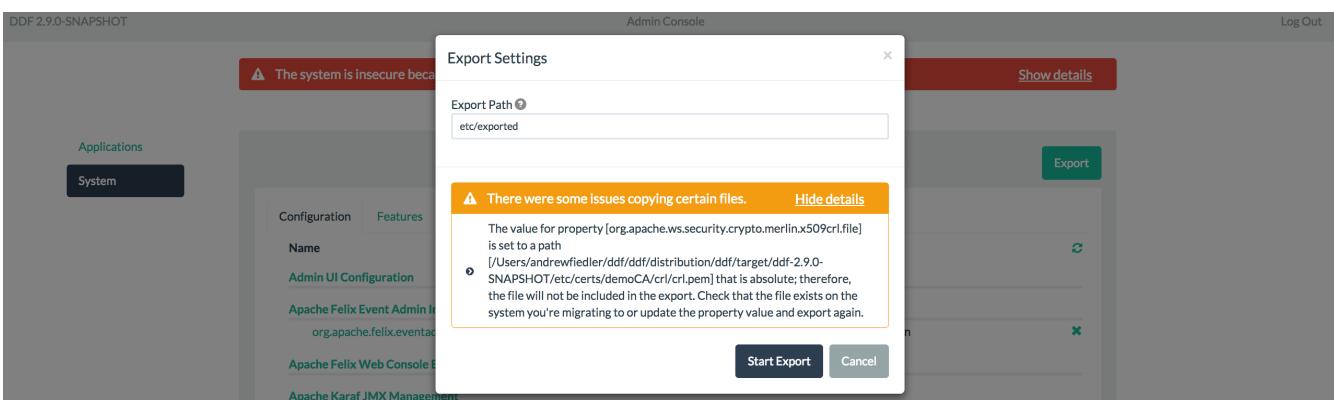


Figure 3. Absolute File Path in Configuration Export

IMPORTANT

Some system configuration files contain paths to other configuration files. For instance, the `system.properties` file contains the `javax.net.ssl.keyStore` property which provides the path to system key store. The files referred to in the system configuration files will be included in the export process only if the path is relative to `<DDF_HOME>`. Using absolute paths and/or symbolic links in those cases will cause the `migration:export` command to display warnings about those files and exclude them from the export process. The export process itself will not be aborted.

14.6.2. Importing the Exported Configuration Settings

WARNING

Importing of configurations must take place before starting the new installation for the first time.

To import a previously exported configuration:

1. Delete all existing `.config` files from `<DDF_HOME>/etc`:
 - a. `cd <{$branding}_HOME>`
 - b. `rm etc/*.config`
 - c. `unzip /path/to/exported/zip/exportFiles.zip`
2. Unzip the exported files from a previous installation to the new instance's `<DDF_HOME>/etc` directory: `unzip exportedFiles.zip <DDF_HOME>/etc`
3. If needed, manually update system configuration files such as `system.properties`, `users.properties`, keystores, etc.
4. Launch the newly installed DDF.
5. Step through the installation process. The newly installed DDF will have the previous DDF's settings imported.
6. To get a status of the import, run the `migration:status` from the Command Console.

14.7. Other Configurations

14.7.1. Environment Hardening

- Required Step for Security Hardening

IMPORTANT

It is recommended to apply the following security mitigations to the DDF.

Protocol/ Type	Risk	Mitigation

JMX	tampering, information disclosure, and unauthorized access	<ul style="list-style-type: none"> * Remove <code>-Dcom.sun.management.jmxremote</code> from <code><DDF_HOME>/bin/karaf</code>. * Disable DDF's JMX management <code>rmiRegistryPort</code> and <code>rmiServerPort (1099, 44444)</code> by removing these entries from <code>etc/org.apache.karaf.management.cfg</code> * Uninstall the management bundle using the command line console: <code>uninstall management</code>
File System Access	tampering, information disclosure, and denial of service	<p>Set OS File permissions under the <code><DDF_HOME></code> directory (e.g. <code>/deploy</code>, <code>/etc</code>) to ensure unauthorized viewing and writing is not allowed. If Caching is installed:</p> <ul style="list-style-type: none"> * Set permissions for the installation directory <code>/data/product-cache</code> such that only the DDF process and users with the appropriate SCI controls and classification levels can view any stored product. * Caching can be turned off as well to mitigate this risk. <p>To disable caching, navigate to Catalog Select Standard Catalog Framework Uncheck the <code>Enable Product Caching</code> box.</p> <p>* Install Security to ensure only the appropriate users are accessing the products.</p> <p>From the Admin Console, select Manage Applications.</p> <p>** Install Security, if applicable.</p> <p>* Cached files are written by the user running the DDF <code>process/application</code>.</p> <p>On system: ensure that not everyone can change ACLs on your object. On a network: ensure that you don't use up resources when contacted by an anonymous entity.</p>
SSH	tampering, information disclosure, and denial of service	<p>Create accounts for authentication (modify the default user's password) or remove ssh feature. By definition the connection will be secure when authenticated. Optionally, disable the ssh port (<code>default: 8101</code>) and set the <code>karaf.startRemoteShell</code> system property to false.</p> <p>[WARNING] ===== Disabling ssh will be more secure, but will disable any direct command line access to the DDF.</p> <p>===== set <code>karaf.shutdown.port=-1</code> in <code>etc/custom.properties</code> or <code>etc/config.properties</code></p>
SSL/TLS	man-in-the-middle, information disclosure	<p>Update the <code><DDF_HOME>/etc/org.ops4j.pax.web.cfg</code> file to add the entry <code>org.ops4j.pax.web.ssl.clientauthneeded=true</code>. [WARNING] ===== Setting this configuration may break compatibility to legacy systems that do not support two-way SSL. =====</p>
Session Inactivity Timeout	unauthorized access	<p>Update the Session configuration to have no greater than 10 minute Session Timeout.</p> <ul style="list-style-type: none"> * Navigate to the Admin Console. * Select the Security application. * Select the Configuration tab. * Select Session. * Set Session Timeout (in minutes) to 10 (or less).

15. Quick Start Tutorial

This quick tutorial will enable install, configuring and using a basic instance of DDF.

NOTE

This tutorial is intended for setting up a test, demonstration, or trial installation of DDF. For more complete installation and configuration steps, see [Installing](#).

These steps will demonstrate:

- [Prerequisites](#).
- [Quick Install of DDF](#).
- [Ingesting Data](#).

15.1. Quick Install Prerequisites

These are the basic requirements to set up the environment to run a DDF.

15.1.1. Hardware Requirements (Quick Install)

- At least 4096MB of memory for DDF.
 - This amount can be increased to support memory-intensive applications. See [Memory Considerations](#).

15.1.2. Java Requirements (Quick Install)

Set up Java to run DDF.

- Install/Upgrade to Java 8 [J2SE 8 SDK](#)
 - The recommended version is [8u60](#) or later.
 - Java Version and Build numbers must contain only number values.
- Supported platforms are *NIX - Unix/Linux/OSX, Solaris, and Windows.
- [JDK8](#) must be installed.
- The [JAVA_HOME](#) environment variable must be set to the location where the JDK is installed.

1. Setting JAVA_HOME variable (Replace <JAVA_VERSION> with the version and build number installed.)

1. Determine Java Installation Directory (This varies between operating system versions).

*Find Java Path in *NIX*

```
which java
```

Find Java Path in Windows

```
for %i in (java.exe) do @echo. %~$PATH:i
```

1. Copy path to Java installation. (example: /usr/java/<JAVA_VERSION>)

2. Set JAVA_HOME by replacing <PATH_TO_JAVA> with the copied path in this command:

*Setting JAVA_HOME on *NIX*

```
JAVA_HOME=<PATH_TO_JAVA><JAVA_VERSION>
export JAVA_HOME
```

Setting JAVA_HOME on Windows

```
set JAVA_HOME=<PATH_TO_JAVA><JAVA_VERSION> /m
```

Adding JAVA_HOME to PATH Environment Variable on Windows

```
setx PATH "%PATH%;%JAVA_HOME%\bin" /m
```

*NIX

WARNING Unlink JAVA_HOME if it is already linked to a previous version of the JRE: [unlink JAVA_HOME](#)

Verify that the JAVA_HOME was set correctly.

*NIX

```
echo $JAVA_HOME
```

TIP

Windows

```
echo %JAVA_HOME%
```

File Descriptor Limit on Linux

- For Linux systems, increase the file descriptor limit by editing `/etc/sysctl.conf` to include:

```
fs.file-max = 6815744
```

NOTE

- (This file may need permissions changed to allow write access).
- For the change to take effect, a restart is required.
 1. *Nix Restart Command

```
init 6
```

Check System Time

WARNING

Prior to installing DDF, ensure the system time is accurate to prevent federation issues.

15.2. Quick Install of DDF

Check System Time

WARNING

Prior to installing DDF, ensure the system time is accurate to prevent federation issues.

1. Download the DDF [zip file](#).
2. Install DDF by unzipping the zip file.

Windows Zip Utility Warning

The default Windows zip utility (such as double-clicking file) will not work to unzip the distribution, use Java or a third party utility instead.

WARNING

Use Java to Unzip in Windows(Replace <PATH_TO_JAVA> with correct path and <JAVA_VERSION> with current version.)

```
"<PATH_TO_JAVA>\jdk<JAVA_VERSION>\bin\jar.exe" xf ddf-2.10.3.zip
```

3. This will create an installation directory, which is typically created with the name and version of the application. This installation directory will be referred to as `<DDF_HOME>`. (Substitute the actual directory name.)
4. Start DDF by running the `<DDF_HOME>/bin/ddf` script (or `ddf.bat` on Windows).
5. The Command Console will display.

```
ddf@local>
```

15.3. Configuring (Quick Start)

Set the configurations needed to run DDF.

1. In a browser, navigate to the Admin Console at <https://localhost:8993/admin>.
 - a. The Admin Console may take a few minutes to start up.
2. Enter the default username of **admin** and the password of **admin**.
3. Follow the installer prompts for a standard installation.
 - a. Click start to begin the setup process.
 - b. Configure **guest claims attributes** or use defaults.
 - i. (these attributes represent the minimum authorization for all users)
 - c. Select **Standard Installation**.
 - d. On the System Configuration page, configure any port or protocol changes desired and add any keystores/truststores needed.
 - i. See [Configuring New Certificates](#) for more details.
 - e. Click **Next**
 - f. Click **Finish**

15.4. Ingesting (Quick Start)

Now that DDF has been configured, ingest some sample data to demonstrate search capabilities.

This is one way to ingest into the catalog, for a complete list of the different methods, see [Ingesting Data](#).

15.4.1. Ingesting Sample Data

1. Download a sample valid [GeoJson file here](#).
2. Navigate in the browser to the Search UI at <https://localhost:8993/search>.
3. Click the **upload** icon in the upper right corner.
4. Select the file to upload.

NOTE XML metadata for text searching is not automatically generated from GeoJson fields.

Querying from the Search UI (<https://localhost:8993/search>) will return the record for the file ingested:

- Enter **in** in the Text search box and click the *Search button.

NOTE

The sample data was selected as an example of well-formed metadata. Other data can and should be used to test other usage scenarios.

16. Running

Find directions here for running an installation of DDF.

Starting

Getting an instance of DDF up and running.

Maintaining

Keeping DDF running with useful tasks.

Monitoring

Tracking system health and usage.

Troubleshooting

Common tips for unexpected behavior.

16.1. Starting

Follow the below steps to start and stop DDF.

Memory Considerations

If the DDF will be running memory-intensive applications, consider increasing the java memory. The java memory can be changed by updating the setenv script:

*Setenv Script: *NIX*

<DDF_HOME>/bin/setenv
Update the JAVA_MAX_MEM property

NOTE

Setenv Script: Windows

<DDF_HOME>/bin/setenv.bat
Update the JAVA_OPTS -Xmx value

16.1.1. Starting from Startup Scripts

Run one of the start scripts from a command shell to start the distribution and open a local console:

*Start Script: *NIX*

```
<DDF_HOME>/bin/ddf
```

Start Script: Windows

```
<DDF_HOME>/bin/ddf.bat
```

16.1.2. Starting as a Service

Alternatively, to run DDF as a background service, run the **start** script:

**NIX*

```
<DDF_HOME>/bin/start
```

Windows

```
<DDF_HOME>/bin/start.bat
```

If console access is needed while running as a service, run the **client** script on the host where the DDF is running:

**NIX*

```
<DDF_HOME>/bin/client
```

NOTE

Windows

```
<DDF_HOME>/bin/client.bat -h localhost
```

If running the **client** script from a different host, use the **-h** option followed by the name or IP of the host where DDF is running.

16.1.3. Stopping DDF

There are two options to stop a running instance:

- Call shutdown from the console:

Shut down with a prompt

```
ddf@local>shutdown
```

Force Shutdown without prompt

```
ddf@local>shutdown -f
```

- Keyboard shortcut for shutdown
 - **Ctrl-D**
 - **Cmd-D**
- Or run the stop script:

*NIX

```
<DDF_HOME>/bin/stop
```

Windows

```
<DDF_HOME>/bin/stop.bat
```

Shut Down

IMPORTANT

Do not shut down by closing the window (Windows, Unix) or using the **kill -9 <pid>** command (Unix). This prevents a clean shutdown and can cause significant problems when DDF is restarted. Always use the shutdown command or the shortcut from the command line console.

16.1.4. Automatic Start on System Boot

Because DDF is built on top of Apache Karaf, DDF can use the Karaf Wrapper to enable automatic startup and shutdown.

1. Create the Karaf wrapper.

Within the DDF console

```
ddf@local> feature:install -r wrapper
ddf@local> wrapper:install -s AUTO_START -n ddf -d ddf -D "DDF Service"
```

2. (Windows users skip to next step) (All *NIX) If DDF was installed to run as a non-root user (recommended,) edit **<DDF_HOME>/bin/ddf-service**.

Change:

```
<DDF_HOME>/bin/ddf-service
```

```
#RUN_AS_USER=
```

to:

```
<DDF_HOME>/bin/ddf-service
```

```
RUN_AS_USER=<ddf-user>
```

3. Set the **DDF_HOME** property.

```
<DDF_HOME>/etc/ddf-wrapper.conf
```

```
set.default.DDF_HOME=%KARAF_HOME%
```

4. Set the memory in the wrapper config to match with DDF default memory setting.

```
<DDF_HOME>/etc/ddf-wrapper.conf
```

```
#Add the following:  
wrapper.java.additional.10=-Dddf.home=%KARAF_HOME%  
wrapper.java.additional.11=-Dderby.storage.fileSyncTransactionLog=true  
wrapper.java.additional.12=-server  
wrapper.java.additional.13=-Dcom.sun.management.jmxremote  
wrapper.java.additional.14=-Djava.security.egd=file:/dev/.urandom  
wrapper.java.additional.15=-Dfile.encoding=UTF8  
wrapper.java.additional.16=-Dkaraf.instances=%KARAF_HOME%/instances  
wrapper.java.additional.17=-Dkaraf.restart.jvm.supported=true  
wrapper.java.additional.18=-Djava.io.tmpdir=%KARAF_HOME%/data/tmp  
wrapper.java.additional.19=-Djava.util.logging.config.file=%KARAF_HOME%/etc/java  
.util.logging.properties  
wrapper.java.additional.20=-XX:+UnlockDiagnosticVMOptions  
wrapper.java.additional.21=-XX:+UnsyncloadClass  
wrapper.java.additional.22=-Dderby.system.home=%KARAF_HOME%/data/derby  
wrapper.java.additional.23=-Djava.awt.headless=true  
  
# Set the JVM max heap space as desired  
wrapper.java.additional.24=-Xmx4g
```

5. Install the wrapper startup/shutdown scripts.

Windows

Run the following command in a console window. The command must be run with elevated permissions.

```
<DDF_HOME>/bin/ddf-service.bat install
```

Startup and shutdown settings can then be managed through **Services** → **MMC Start** → **Control Panel** → **Administrative Tools** → **Services**.

Redhat

```
root@localhost# ln -s <DDF_HOME>/bin/ddf-service /etc/init.d/
root@localhost# chkconfig ddf-service --add
root@localhost# chkconfig ddf-service on
```

Ubuntu

```
root@localhost# ln -s <DDF_HOME>/bin/ddf-service /etc/init.d/
root@localhost# update-rc.d -f ddf-service defaults
```

Solaris

```
root@localhost# ln -s <DDF_HOME>/bin/ddf-service /etc/init.d/
root@localhost# ln -s /etc/init.d/ddf-service /etc/rc0.d/K20ddf-service
root@localhost# ln -s /etc/init.d/ddf-service /etc/rc1.d/K20ddf-service
root@localhost# ln -s /etc/init.d/ddf-service /etc/rc2.d/K20ddf-service
root@localhost# ln -s /etc/init.d/ddf-service /etc/rc3.d/S20ddf-service
```

WARNING

While it is not a necessary step, information on how to convert the System V init scripts to the Solaris System Management Facility can be found at <http://www.oracle.com/technetwork/articles/servers-storage-admin/scripts-to-smf-1641705.html>

WARNING

Solaris-Specific Modification

Due to a slight difference between the Linux and Solaris implementation of the `ps` command, the `ddf-service` script needs to be modified.

6. Locate the following line in <DDF_HOME>/bin/ddf-service

Solaris <DDF_HOME>/bin/ddf-service

```
pidtest='$PSEXEC -p $pid -o command | grep $WRAPPER_CMD | tail -1'
```

7. Change the word command to comm.

Solaris <DDF_HOME>/bin/ddf-service

```
pidtest='$PSEXEC -p $pid -o comm | grep $WRAPPER_CMD | tail -1'
```

Karaf Documentation

Because DDF is built on Apache Karaf, more information on operating DDF can be found in the [Karaf documentation](#).

16.2. Maintaining

16.2.1. Console Commands

Once the distribution has started, users will have access to a powerful command line console, the Command Console. This Command Console can be used to manage services, install new features and applications, and manage the state of the system.

Accessing the System Console

The Command Line Console is the console that is available to the user when the distribution is started manually. It may also be accessed by using the `bin/client.bat` or `bin/client.sh` scripts.

NOTE

The majority of functionality and information available on the Admin Console is also available on the Command Line Console.

Available System Console Commands

To get a list of commands, type in the namespace of the desired extension then press the **Tab** key.

For example, type `catalog`, then press **Tab**.

System Console Command Help

For details on any command, type `help` then the command. For example, `help search` (see results of this command in the example below).

Example Help

```
ddf@local>help search
DESCRIPTION
    catalog:search
        Searches records in the catalog provider.
SYNTAX
    catalog:search [options] SEARCH_PHRASE [NUMBER_OF_ITEMS]
ARGUMENTS
    SEARCH_PHRASE
        Phrase to query the catalog provider.
    NUMBER_OF_ITEMS
        Number of maximum records to display.
        (defaults to -1)
OPTIONS
    --help
        Display this help message
    case-sensitive, -c
        Makes the search case sensitive
    -p, -provider
        Interacts with the provider directly instead of the framework.
```

The `help` command provides a description of the provided command, along with the syntax in how to use it, arguments it accepts, and available options.

Table 12. Available Console Commands

Title	Namespace	Description
DDF::Catalog :: Core :: Commands	catalog	The Catalog Shell Commands are meant to be used with any CatalogProvider implementations. They provide general useful queries and functions against the Catalog API that can be used for debugging, printing, or scripting.
DDF :: Admin :: Application Service	app	The Admin Application Service contains operations to work with applications.
DDF :: Catalog :: Core :: PubSub Commands	subscription	The DDF PubSub shell commands provide functions to list the registered subscriptions in DDF and to delete subscriptions.
DDF Platform Commands	platform	The DDF Platform Shell Commands provide generic platform management functions
DDF::Persistence :: Core :: Commands	store	The Persistence Shell Commands are meant to be used with any PersistentStore implementations. They provide the ability to query and delete entries from the persistence store.

Catalog Commands

WARNING

Most commands can bypass the Catalog framework and interact directly with the Catalog provider if given the `--provider` option, if available. No pre/post plugins are executed and no message validation is performed if the `--provider` option is used.

Table 13. Catalog Command Descriptions

Command	Description
catalog:describe	Provides a basic description of the Catalog implementation.
catalog:dump	Exports metacards from the local Catalog. Does not remove them. See date filtering options below.

Command	Description
<code>catalog:envlist</code>	[IMPORTANT] === Deprecates as of ddf-catalog 2.5.0. Please use <code>platform:envlist</code> . === Provides a list of environment variables.
<code>catalog:ingest</code>	Ingests data files into the Catalog.
<code>catalog:inspect</code>	Provides the various fields of a metocard for inspection.
<code>catalog:latest</code>	Retrieves the latest records from the Catalog based on the Metocard.MODIFIED date.
<code>catalog:migrate</code>	Allows two <code>CatalogProvider</code> s to be configured and migrates the data from the primary to the secondary.
<code>catalog:range</code>	Searches by the given range arguments (exclusively).
<code>catalog:remove</code>	Deletes a record from the local Catalog.
<code>catalog:removeall</code>	Attempts to delete all records from the local Catalog.
<code>catalog:replicate</code>	Replicates data from a federated source into the local Catalog.
<code>catalog:search</code>	Searches records in the local Catalog.
<code>catalog:spatial</code>	Searches spatially the local Catalog.
<code>catalog:validate</code>	Validates an XML file against all installed validators and prints out human readable errors and warnings.

`catalog:dump` Options

The `catalog:dump` command was extended in DDF version 2.5.0 to provide selective export of metacards based on date ranges. The `--created-after` and `--created-before` options allow filtering on the date and time that the metocard was created, while `--modified-after` and `--modified-before` options allow filtering on the date and time that the metocard was last modified (which is the created date if no other modifications were made). These date ranges are exclusive (i.e., if the date and time match exactly, the metocard will not be included). The date filtering options (`--created-after`, `--created-before`, `--modified-after`, and `--modified-before`) can be used in any combination, with the export result including only metacards that match all of the provided conditions.

If no date filtering options are provided, created and modified dates are ignored, so that all metacards match.

Date Syntax

Supported dates are taken from the common subset of ISO8601, matching the datetime from the following syntax:

```

datetime      = time | date-opt-time
time         = 'T' time-element [offset]
date-opt-time = date-element ['T' [time-element] [offset]]
date-element  = std-date-element | ord-date-element | week-date-element
std-date-element = yyyy ['-' MM ['- dd]]
ord-date-element = yyyy ['- DDD]
week-date-element = xxxx '-W' ww ['- e]
time-element   = HH [minute-element] | [fraction]
minute-element = ':' mm [second-element] | [fraction]
second-element = ':' ss [fraction]
fraction       = ('.' | ',') digit+
offset         = 'Z' | (( '+' | '-' ) HH [':' mm [':' ss [('.') | ',' ) SSS]])
```

catalog:dump Examples

```

ddf@local> // Given we've ingested a few metacards
ddf@local> catalog:latest
#          ID                  Modified Date        Title
1  a6e9ae09c792438e92a3c9d7452a449f  2014-06-13T09:56:18+10:00
2  b4aced45103a400da42f3b319e58c3ed  2014-06-13T09:52:12+10:00
3  a63ab22361e14cee9970f5284e8eb4e0  2014-06-13T09:49:36+10:00  myTitle

ddf@local> // Filter out older files
ddf@local> catalog:dump --created-after 2014-06-13T09:55:00+10:00 /home/bradh/ddf-
catalog-dump
1 file(s) dumped in 0.015 seconds

ddf@local> // Filter out new file
ddf@local> catalog:dump --created-before 2014-06-13T09:55:00+10:00 /home/bradh/ddf-
catalog-dump
2 file(s) dumped in 0.023 seconds

ddf@local> // Choose middle file
ddf@local> catalog:dump --created-after 2014-06-13T09:50:00+10:00 --created-before
2014-06-13T09:55:00+10:00 /home/bradh/ddf-catalog-dump
1 file(s) dumped in 0.020 seconds

ddf@local> // Modified dates work the same way
ddf@local> catalog:dump --modified-after 2014-06-13T09:50:00+10:00 --modified-before
2014-06-13T09:55:00+10:00 /home/bradh/ddf-catalog-dump
1 file(s) dumped in 0.015 seconds

ddf@local> // Can mix and match, most restrictive limits apply
ddf@local> catalog:dump --modified-after 2014-06-13T09:45:00+10:00 --modified-before
2014-06-13T09:55:00+10:00 --created-before 2014-06-13T09:50:00+10:00 /home/bradh/ddf-
catalog-dump
1 file(s) dumped in 0.024 seconds

ddf@local> // Can use UTC instead of (or in combination with) explicit timezone offset
```

```

ddf@local>catalog:dump --modified-after 2014-06-13T09:50:00+10:00 --modified-before
2014-06-13T09:55:00Z /home/bradh/ddf-catalog-dump
2 file(s) dumped in 0.020 seconds
ddf@local>catalog:dump --modified-after 2014-06-13T09:50:00+10:00 --modified-before
2014-06-12T23:55:00Z /home/bradh/ddf-catalog-dump
1 file(s) dumped in 0.015 seconds

ddf@local>// Can leave off timezone, but default (local time on server) may not match
what you expect!
ddf@local>catalog:dump --modified-after 2014-06-13T09:50:00 --modified-before 2014-06-
13T09:55:00 /home/bradh/ddf-catalog-dump
1 file(s) dumped in 0.018 seconds

ddf@local>// Can leave off trailing minutes / seconds
ddf@local>catalog:dump --modified-after 2014-06-13T09 --modified-before 2014-06-
13T09:55 /home/bradh/ddf-catalog-dump
2 file(s) dumped in 0.024 seconds

ddf@local>// Can use year and day number
ddf@local>catalog:dump --modified-after 2014-164T09:50:00 /home/bradh/ddf-catalog-dump
2 file(s) dumped in 0.027 seconds

```

Application Commands

Application commands are used from the DDF Admin application to manage applications in the DDF.

NOTE The Application Commands are installed automatically with the Admin Application.

Table 14. App Command Descriptions

Command	Syntax	Description
add	app:add appUri	Adds an application with the given uri.
remove	app:remove appName	Removes an application with the given name.
start	app:start appName	Starts an application with the given name.
stop	app:stop appName	Stops an application with the given name.
list	app:list	Lists the applications that are in the system and gives their current state.
status	app:status appName	Shows status of an application. Gives information on the current state, features within the application, what required features are not started and what required bundles are not started.
tree	app:tree	Creates a hierarchy tree of all of the applications.

Application Command Usage

Listing All Applications

```
ddf@local>app:list
State      Name
[ACTIVE   ] catalog-app-<VERSION>
[ACTIVE   ] distribution-<VERSION>
[ACTIVE   ] platform-app-<VERSION>

[...]
```

This list shows all of the applications installed in DDF. From here, use the name of an application to get more information on its status.

Get Status for a Specific Application

```
ddf@local>app:status catalog-app-<VERSION>
catalog-app-<VERSION>
```

Current State is: ACTIVE

Features Located within this Application:

```
catalog-security-filter
catalog-transformer-resource
catalog-rest-endpoint
abdera
catalog-transformer-xml
catalog-transformer-thumbnail
catalog-transformer-metadata
catalog-transformer-xsltengine
catalog-core-fanoutframework
catalog-transformer-tika
catalog-core-api
catalog-opensearch-source
catalog-plugin-federationreplication
catalog-opensearch-endpoint
catalog-schematron-plugin
catalog-transformer-geoformatter
catalog-transformer-atom
catalog-core-sourcetricsplugin
catalog-core-metricsplugin
catalog-app
catalog-transformer-json
catalog-core-standardframework
catalog-core
```

Required Features Not Started

NONE

Required Bundles Not Started

NONE

Application in Failed State

If an application is in a 'FAILED' state, it means that there is a required feature or bundle that is not started.

App in Failed State

```
ddf@local>app:list
State      Name
[FAILED   ] catalog-app-<VERSION>
[ACTIVE   ] distribution-<VERSION>
[ACTIVE   ] platform-app-<VERSION>
```

In the above case, the catalog app is in a failed state. Checking the status of that application will show what did not start correctly.

Check Application Status

```
ddf@local>app:status catalog-app-<VERSION>
catalog-app-<VERSION>
```

Current State is: FAILED

Features Located within this Application:

```
catalog-security-filter
catalog-transformer-resource
catalog-rest-endpoint
abdera
catalog-transformer-xml
catalog-transformer-thumbnail
catalog-transformer-metadata
catalog-transformer-xsltengine
catalog-core-fanoutframework
catalog-transformer-tika
catalog-core-api
catalog-opensearch-source
catalog-plugin-federationreplication
catalog-opensearch-endpoint
catalog-schematron-plugin
catalog-transformer-geoformatter
catalog-transformer-atom
catalog-core-sourcetricsplugin
catalog-core-metricsplugin
catalog-app
catalog-transformer-json
catalog-core-standardframework
catalog-core
```

Required Features Not Started

NONE

Required Bundles Not Started

```
[261] catalog-opensearch-endpoint
```

This status shows that bundle &261, the catalog-opensearch-endpoint, did not start. Performing a 'list' on the console verifies this:

```
[ 261] [Resolved ] [ ] [ ] [ 80] DDF :: Catalog :: OpenSearch ::  
Endpoint (<VERSION>)
```

Once that bundle is started by fixing its error, the catalog application will show as being in an ACTIVE state.

Subscriptions Commands

NOTE The subscriptions commands are installed when the Catalog application is installed.

Table 15. Subscription Command Descriptions

Command	Description
<code>subscription:delete</code>	Deletes the subscription(s) specified by the search phrase or LDAP filter.
<code>subscription:list</code>	List the subscription(s) specified by the search phrase or LDAP filter.

subscriptions:list Command Usage Examples

Note that no arguments are required for the `subscriptions:list` command. If no argument is provided, all subscriptions will be listed. A count of the subscriptions found matching the list command's search phrase (or LDAP filter) is displayed first followed by each subscription's ID.

List All Subscriptions

```
ddf@local>subscriptions:list  
  
Total subscriptions found: 3  
  
Subscription ID  
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL  
my.contextual.id.v30|http://172.18.14.169:8088/mockEventConsumerBinding?WSDL  
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification
```

List a Specific Subscription by ID

```
ddf@local>subscriptions:list  
"my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL"  
  
Total subscriptions found: 1  
  
Subscription ID  
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL
```

WARNING

It is recommended to always quote the search phrase (or LDAP filter) argument to the command so that any special characters are properly processed.

List Subscriptions Using Wildcards

```
ddf@local>subscriptions:list "my*"
```

Total subscriptions found: 3

Subscription ID

```
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL  
my.contextual.id.v30|http://172.18.14.169:8088/mockEventConsumerBinding?WSDL  
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification
```

```
ddf@local>subscriptions:list "*json*"
```

Total subscriptions found: 1

Subscription ID

```
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification
```

```
ddf@local>subscriptions:list "*WSDL"
```

Total subscriptions found: 2

Subscription ID

```
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL  
my.contextual.id.v30|http://172.18.14.169:8088/mockEventConsumerBinding?WSDL
```

The example below illustrates searching for any subscription that has "json" or "v20" anywhere in its subscription ID.

List Subscriptions Using an LDAP Filter

```
ddf@local>subscriptions:list -f "((subscription-id=*json*) (subscription-id=*v20*))"
```

Total subscriptions found: 2

Subscription ID

```
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL  
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification
```

The example below illustrates searching for any subscription that has **json** and **172.18.14.169** in its subscription ID. This could be a handy way of finding all subscriptions for a specific site.

```
ddf@local>subscriptions:list -f "(&(subscription-id=*json*) (subscription-id=*172.18.14.169*))"

Total subscriptions found: 1

Subscription ID
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification
```

subscriptions:delete Command Usage

The arguments for the `subscriptions:delete` command are the same as for the `list` command, except that a search phrase or LDAP filter must be specified. If one of these is not specified an error will be displayed. When the `delete` command is executed it will display each subscription ID it is deleting. If a subscription matches the search phrase but cannot be deleted, a message in red will be displayed with the ID. After all matching subscriptions are processed, a summary line is displayed indicating how many subscriptions were deleted out of how many matching subscriptions were found.

Delete a Specific Subscription Using Its Exact ID

```
ddf@local>subscriptions:delete
"my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification"

Deleted subscription for ID =
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification

Deleted 1 subscriptions out of 1 subscriptions found.
```

Delete Subscriptions Using Wildcards

```
ddf@local>subscriptions:delete "my*"

Deleted subscription for ID =
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL
Deleted subscription for ID =
my.contextual.id.v30|http://172.18.14.169:8088/mockEventConsumerBinding?WSDL

Deleted 2 subscriptions out of 2 subscriptions found.

ddf@local>subscriptions:delete "*json*"

Deleted subscription for ID =
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification

Deleted 1 subscriptions out of 1 subscriptions found.
```

Delete All Subscriptions

```
ddf@local>subscriptions:delete *

Deleted subscription for ID =
my.contextual.id.v30|http://172.18.14.169:8088/mockEventConsumerBinding?WSDL
Deleted subscription for ID =
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL
Deleted subscription for ID =
my.contextual.id.json|http://172.18.14.169:8088/services/json/local/event/notification

Deleted 3 subscriptions out of 3 subscriptions found.
```

Delete Subscriptions Using an LDAP Filter

```
ddf@local>subscriptions:delete -f "(&(subscription-id=*WSDL) (subscription-
id=*172.18.14.169*))"

Deleted subscription for ID =
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL
Deleted subscription for ID =
my.contextual.id.v30|http://172.18.14.169:8088/mockEventConsumerBinding?WSDL

Deleted 2 subscriptions out of 2 subscriptions found.
```

Platform Commands

NOTE The Platform Commands are installed when the Platform application is installed.

Table 16. Platform Command Descriptions

Command	Description
config-export	Exports the current configurations.
config-status	Lists import status of configuration files.
describe	Shows the current platform configuration.
envlist	Provides a list of environment variables.

Persistence Store Commands

Table 17. Persistence Store Command Descriptions

Command	Description
store:delete	Delete entries from the persistence store that match a given CQL statement
store:list	Lists entries that are stored in the persistence store.

16.2.2. Command Scheduler

Command Scheduler is a capability exposed through the Admin Console (<https://localhost:8993/admin>) that allows administrators to schedule Command Line Commands to be run at specified intervals.

Using the Command Scheduler

The Command Scheduler allows administrators to schedule Command Line Shell Commands to be run in a platform-independent way. For instance, if an administrator wanted to use the Catalog commands to export all records of a Catalog to a directory, the administrator could write a cron job or a scheduled task to remote into the container and execute the command. Writing these types of scripts are specific to the administrator's operating system and also requires extra logic for error handling if the container is up. The administrator can also create a Command Schedule, which currently requires only two fields. The Command Scheduler only runs when the container is running, so there is no need to verify if the container is up. In addition, when the container is restarted, the commands are rescheduled and executed again.

Schedule a Command

Configure the Command Scheduler to execute a command at specific intervals.

1. Navigate to the Admin Console (<https://localhost:8993/admin>).
2. Select the **Platform** application.
3. Select **Platform Command Scheduler**.
4. Enter the command or commands to be executed in the **Command** text field. Commands can be separated by a semicolon and will execute in order from left to right.
5. Enter an interval in the **Interval** field. This can either be a Quartz Cron expression or a positive integer (seconds) (e.x. '0 0 0 1/1 * ? *' or '12').
6. Select the interval type in the **Interval Type** drop-down.
7. Select the **Save** button.

NOTE

Scheduled Commands can be updated and deleted. To delete, clear the fields and click **Save**. To update, modify the fields and click **Save**.

Updating a Scheduled Command

Change the timing, order, or execution of scheduled commands.

1. Navigate to the **Admin Console**.
2. Click on the **Platform** application.
3. Click on the **Configuration** tab.
4. Under the **Platform Command Scheduler** configuration are all the scheduled commands. Scheduled commands have the following syntax `ddf.platform.scheduler.Command.{GUID}` such as `ddf.platform.scheduler.Command.4d60c917-003a-42e8-9367-1da0f822ca6e`.

- Find the desired configuration to modify and update either the **Command** text field, the **Interval** field, or the **Interval Type**.
- Click **Save changes**. Once the Save button has been clicked, the command will be executed immediately. Its next scheduled execution happens after the time specified in Interval In Seconds and repeats indefinitely until the container is shutdown or the Scheduled Command is deleted.

Command Output

Commands that normally write out to the console will write out to the distribution's log. For example, if an `echo "Hello World"` command is set to run every five seconds, the log displays the following:

Sample Command Output in the Log

```
16:01:32,582 | INFO  | heduler_Worker-1 | ddf.platform.scheduler.CommandJob
68 | platform-scheduler  | Executing command [echo Hello World]
16:01:32,583 | INFO  | heduler_Worker-1 | ddf.platform.scheduler.CommandJob
70 | platform-scheduler  | Execution Output: Hello World
16:01:37,581 | INFO  | heduler_Worker-4 | ddf.platform.scheduler.CommandJob
68 | platform-scheduler  | Executing command [echo Hello World]
16:01:37,582 | INFO  | heduler_Worker-4 | ddf.platform.scheduler.CommandJob
70 | platform-scheduler  | Execution Output: Hello World
```

In short, administrators can view the status of a run within the log as long as INFO was set as the status level.

CQL Syntax

The CQL syntax used with console commands should follow the OGC CQL format. GeoServer provides a description of the grammar and examples at [CQL Tutorial](#).

CQL Syntax Examples

```
Finding all notifications that were sent due to a download:
ddf@local>store:list --cql "application='Downloads'" --type notification
```

```
Deleting a specific notification:
ddf@local>store:delete --cql "id='fdc150b157754138a997fe7143a98cfa'" --type notification
```

16.3. Monitoring

The DDF contains many tools to monitor system functionality, usage, and overall system health.

16.3.1. Metrics Reporting

Metrics are available in several formats and levels of detail.

Complete the following procedure now that several queries have been executed.

1. Select **Platform**
2. Select **Metrics** tab
3. For individual metrics, choose the format desired from the desired timeframe column:
 - a. PNG
 - b. CSV
 - c. XLS
4. For a detailed report of all metrics, at the bottom of the page are selectors to choose time frame and summary level. A report is generated in *xls* format.

16.3.2. Managing Logging

The DDF supports a dynamic and customizable logging system including log level, log format, log output destinations, roll over, etc.

Configuring Logging

Edit the configuration file `<DDF_HOME>/etc/org.ops4j.pax.logging.cfg]`

DDF log file

The name and location of the log file can be changed with the following setting:

`log4j.appender.out.file=<KARAF.DATA>/log/ddf.log`

Controlling log level

A useful way to debug and detect issues is to change the log level:

`log4j.rootLogger=DEBUG, out, osgi:VmLogAppender`

NOTE `osgi:VmLogAppender` is the Appender for the Karaf Console logs.

Controlling the size of the log file

Set the maximum size of the log file before it is rolled over by editing the value of this setting:

`log4j.appender.out.maxFileSize=20MB`

Number of backup log files to keep

Adjust the number of backup files to keep by editing the value of this setting:

`log4j.appender.out.maxBackupIndex=10`

Enabling logging of inbound and outbound SOAP messages for the DDF SOAP endpoints

By default, the DDF start scripts include a system property enabling logging of inbound and

outbound SOAP messages.

```
-Dcom.sun.xml.ws.transport.http.HttpAdapter.dump=true
```

In order to see the messages in the log, one must set the logging level for `org.apache.cxf.services` to `INFO`. By default, the logging level for `org.apache.cxf` is set to `WARN`.

```
ddf@local>log:set INFO org.apache.cxf.services
```

Logging External Resources

Other appenders can be selected and configured.

For more detail on configuring the log file and what is logged to the console see: [Karaf Documentation: Log](#).

Enabling HTTP Access Logging

To enable access logs for the current DDF, do the following:

- Update the `jetty.xml` file located in `etc/` adding the following xml:

```
<Get name="handler">
  <Call name="addHandler">
    <Arg>
      <New class="org.eclipse.jetty.server.handler.RequestLogHandler">
        <Set name="requestLog">
          <New id="RequestLogImpl" class="org.eclipse.jetty.server.NCSARequestLog">
            <Arg><SystemProperty name="jetty.logs" default="data/log/">
            </Arg>
            <Arg>/yyyy_mm_dd.request.log</Arg>
            <Set name="retainDays">90</Set>
            <Set name="append">true</Set>
            <Set name="extended">false</Set>
            <Set name="LogTimeZone">GMT</Set>
          </New>
        </Set>
      </New>
    </Arg>
  </Call>
</Get>
```

Change the location of the logs to the desired location. In the settings above, location will default to `data/log` (same place where the log is located).

The log is using *National Center for Supercomputing Association Applications (NCSA)* or Common format (hence the class 'NCSARequestLog'). This is the most popular format for access logs and can be parsed by many web server analytics tools. Here is a sample output:

```
127.0.0.1 - - [14/Jan/2013:16:21:24 +0000] "GET /favicon.ico HTTP/1.1" 200 0
127.0.0.1 - - [14/Jan/2013:16:21:33 +0000] "GET /services/ HTTP/1.1" 200 0
127.0.0.1 - - [14/Jan/2013:16:21:33 +0000] "GET /services//?stylesheet=1 HTTP/1.1"
200 0
127.0.0.1 - - [14/Jan/2013:16:21:33 +0000] "GET /favicon.ico HTTP/1.1" 200 0
```

Using the LogViewer

Monitor system logs with the **LogViewer**, a convenient "viewing portal" for incoming logs.

- Navigate to the LogViewer at <https://localhost:8993/admin/logviewer>

or

- Navigate to the Admin Console
- Navigate to the **System** tab
- Select **Logs**

The LogViewer displays the most recent 500 log messages by default, but will grow to a maximum of 5000 messages. To view incoming logs, select the **PAUSED** button to toggle it to **LIVE** mode. Switching this back to **PAUSED** will prevent any new logs from being displayed in the LogViewer. Note that this only affects the logs displayed by the LogViewer and does not affect the underlying log.

Log events can be filtered by:

- Log level (**ERROR**, **WARNING**, etc).
 - The LogViewer will display at the currently configured log level for the Karaf logs.
 - See [Controlling Log Level](#) to change log level.
- Log message text.
- Bundle generating the message.

WARNING

It is not recommended to use the LogViewer if the system logger is set to a low reporting level such as **TRACE**. The volume of messages logged will exceed the polling rate, and incoming logs may be missed.

The actual logs being polled by the LogViewer can still be accessed at `<INSTALL_HOME>/data/log`

NOTE

The LogViewer settings don't change any of the underlying logging settings, only which messages are displayed. It does not affect the logs generated or events captured by the system logger.

16.4. Troubleshooting

If, after configuration, a DDF is not performing as expected, consult this table of common fixes and workarounds.

Table 18. General Troubleshooting

Issue	Solution
Unable to unzip distribution on Windows platform	The default Windows zip utility is not compatible with the DDF distribution zip file. Use Java or a third-party zip utility.
Unable to federate on Windows Platform	Windows default firewall is not compatible with DDF.
Ingesting more than 200,000 data files stored NFS shares may cause Java Heap Space error (Linux-only issue).	This is an NFS bug where it creates duplicate entries for some files when doing a file list. Depending on the OS, some Linux machines can handle the bug better and able get a list of files but get an incorrect number of files. Others would have a Java Heap Space error because there are too many file to list. As a workaround, ingest files in batches smaller than 200,000.
Ingesting serialized data file with scientific notation in WKT string causes RuntimeException.	WKT string with scientific notation such as POINT (-34.8932113039107 -4.77974239601E-5) won't ingest. This occurs with serialized data format only.
Exception Starting DDF (Windows) An exception is sometimes thrown starting DDF on a Windows machine (x86). If using an unsupported terminal, <code>java.lang.NoClassDefFoundError: Could not initialize class org.fusesource.jansi.internal.Kernel32</code> is thrown.	Install missing Windows libraries. Some Windows platforms are missing libraries that are required by DDF. These libraries are provided by the Microsoft Visual C++ 2008 Redistributable Package x64 .
CXF BusException The following exception is thrown: <code>org.apache.cxf.BusException : No conduit initiator</code>	Restart DDF.. Shut down DDF: <code>ddf@local>shutdown</code> . Start up DDF: <code>./ddf</code>
Distribution Will Not Start DDF will not start when calling the start script defined during installation.	Complete the following procedure.. Verify that Java is correctly installed. + <code>java -version</code> . This should return something similar to: + <code>java version "1.8.0_45" Java™ SE Runtime Environment (build 1.8.0_45-b14) Java HotSpot™ Server VM (build 25.45-b02, mixed mode)</code> . If running *nix, verify that bash is installed. + <code>echo \$SHELL</code> . This should return: + <code>/bin/bash</code>

Issue	Solution
Multiple <code>java.exe</code> processes running, indicating more than one DDF instance is running. This can be caused when another DDF is not properly shut down.	Perform one or all of the following recommended solutions, as necessary. * Wait for proper shutdown of DDF prior to starting a new instance. * Verify running <code>java.exe</code> are not DDF (e.g., kill/close if necessary). * Utilize automated start/stop scripts to run DDF as a service.

17. Data Management

17.1. File Formats Supported

DDF supports a wide variety of file types and data types for ingest. The DDF's internal Input Transformers extract the necessary data into a generalized format. DDF supports ingest of many datatypes and commonly used file formats, such as Microsoft office products: Word documents, Excel spreadsheets, and PowerPoint presentations as well as .pdf files, GeoJson and others. See [complete list](#).

NOTE

These attributes will be available in all the specified file formats; however, values will only be present if present in the original document/resource.

These attributes are supported by any file type ingested into DDF:

Common Attributes in All File Types

- metadata
- id
- modified (date)
- title (filename)
- metadata content type (mime type)
- effective (date)
- created (date)

These 'media' file types have support for additional attributes to be available when ingested into DDF:

File Types Supporting Additional Attributes

- Video Types
 - WMV
 - AVI
 - MP4
 - MOV
 - h.264 MPEG2

- Image Types
 - JPEG-2000
- Document Types
 - .DOC, .DOCX, .DOTX, .DOCM
 - .PPT, .PPTX
 - .XLS, .XLSX
 - .PDF

These are the attributes common to any of the media file types which support additional attributes:

Additional Possible Attributes Common to 'Media' File Types

- `media.format-version`
- `media.format`
- `media.bit-rate`
- `media.bits-per-sample`
- `media.compression`
- `media.encoding`
- `media.frame-center`
- `media.frame-rate`
- `media.height-pixels`
- `media.number-of-bands`
- `media.scanning-mode`
- `media.type`
- `media.duration`
- `media.page-count`
- `datatype`
- `description`
- `contact.point-of-contact-name`
- `contact.contributor-name`
- `contact.creator-name`
- `contact.publisher-name`
- `contact.point-of-contact-phone`
- `topic.keyword`

Some file types support attributes unique to that individual type:

Additional Unique Attributes by File Type

Mp4

`ext.mp4.audio-sample-rate`

17.2. Ingesting Data

Ingesting is the process of getting metocard(s) into the Catalog Framework. Ingested files are "transformed" into a neutral format that can be searched against as well as migrated to other formats and systems. There are multiple methods available for ingesting files into the DDF.

17.2.1. Methods of Ingest

There are many methods of ingest supported by the DDF to support different needs.

Easy Methods (for fewer records or manual ingest)

These methods are the simplest for small data sets or for testing purposes.

Ingest Command (Command Console)

The DDF console application has a command-line option for ingesting files

The syntax for the ingest command is `ingest -t <transformer type> <file path>` relative to the installation path.

For XML data, run this command:

```
ingest -t xml examples/metacards/xml
```

Directory Monitor

The Catalog application contains a Directory Monitor feature that allows files placed in a single directory to be monitored and ingested automatically. For more information about configuring a directory to be monitored, see [Configuring the Content Directory Monitor](#).

Simply place the desired files in the monitored directory and it will be ingested automatically. If, for any reason, the files cannot be ingested, they will be moved to an automatically created sub-folder named `.errors`. Optionally, ingested files can be automatically moved to a sub-folder called `.ingested`.

17.2.2. Medium Complexity (for more than a few resources)

These methods are useful for slightly larger data sets or infrequent ingest.

External Methods

Several third-party tools, such as [cURL.exe](#) and the [Chrome Advanced Rest Client](#), can be used to send files and other types of data to DDF for ingest.

Windows Example

```
curl -H "Content-type: application/json;id=geojson" -i -X POST -d  
@"C:\path\to\geojson_valid.json" https://localhost:8993/services/catalog
```

**NIX Example*

```
curl -H "Content-type: application/json;id=geojson" -i -X POST -d @geojson_valid.json  
https://localhost:8993/services/catalog
```

Where:

-H adds an HTTP header. In this case, Content-type header **application/json;id=geojson** is added to match the data being sent in the request.

-i requests that HTTP headers are displayed in the response.

-X specifies the type of HTTP operation. For this example, it is necessary to POST (ingest) data to the server.

-d specifies the data sent in the POST request. The **@** character is necessary to specify that the data is a file.

The last parameter is the URL of the server that will receive the data.

This should return a response similar to the following (the actual catalog ID in the id and Location URL fields will be different):

Sample Response

```
HTTP/1.1 201 Created  
Content-Length: 0  
Date: Mon, 22 Apr 2015 22:02:22 GMT  
id: 44dc84da101c4f9d9f751e38d9c4d97b  
Location: https://localhost:8993/services/catalog/44dc84da101c4f9d9f751e38d9c4d97b  
Server: Jetty(7.5.4.v20111024)
```

1. Verify the entry was successfully ingested by entering in a browser the URL returned in the POST response's HTTP header. For instance in our example, it was [/services/catalog/44dc84da101c4f9d9f751e38d9c4d97b](https://localhost:8993/services/catalog/44dc84da101c4f9d9f751e38d9c4d97b). This should display the catalog entry in XML within the browser.
2. Verify the catalog entry exists by executing a query via the OpenSearch endpoint.
3. Enter the following URL in a browser </services/catalog/query?q=ddf>. A single result, in Atom format, should be returned.

17.2.3. Verifying Ingest by REST Endpoint

Verify GeoJson file was stored using the Content REST endpoint.

- Send a GET command to read the content from the content repository using the Content REST endpoint. This can be done using **cURL** command below. Note that the GUID will be different for

each ingest. The GUID can be determined by going to the `<DDF_HOME>/content/store` directory and copying the sub-directory in this folder (there should only be one).

Windows Example

```
curl -X GET https://localhost:8993/services/content/c90147bf86294d46a9d35ebbd44992c5
```

**NIX Example*

```
curl -X GET https://localhost:8993/services/content/c90147bf86294d46a9d35ebbd44992c5
```

The response to the GET command will be the contents of the `geojson_valid.json` file originally ingested.

17.2.4. Advanced (more records, automated ingest)

The DDF provides endpoints for both REST and SOAP services, allowing integration with other data systems and the ability to further automate ingesting data into the catalog.

17.3. Removing Expired Records from Catalog

DDF has many ways to remove expired records from the underlying Catalog data store. Nevertheless, the benefits of data standardization is that an attempt can be made to remove records without the need to know any vendor-specific information. Whether the data store is a search server, a No-SQL database, or a relational database, expired records can be removed universally using the Catalog API and the Catalog Commands.

17.3.1. Manual Removal of Expired Records

To manually remove expired records from the Catalog, execute in the Command Console:

```
catalog:removeall --expired
```

When prompted, type yes to remove all expired records.

TIP For help on the removeall command, execute

```
help catalog:removeall
```

17.3.2. Automated Removal of Expired Records

By default, the DDF runs a scheduled command every 24 hours to remove expired records. The command is executed and scheduled [Using the Command Scheduler](#). To change the configuration out of the box, follow the [Updating a Scheduled Command](#) instructions. If an administrator wants to create additional scheduled tasks to remove records from the local Catalog, the administrator can follow the steps provided in the Scheduling a Command section. In the Command text field, type the following:

```
catalog:removeall --force --expired
```

If it is intended to have this run daily, type 86400 for the amount of seconds. (60 seconds/min x 60 minutes/hr x 24 hours/day = 86400 seconds for one day)

Explanation of Command to Remove Expired Records

The `catalog:removeall` command states you want to remove records from the local Catalog.

The `--force` option is used to suppress the confirmation message which asks a user if the user intentionally wants to permanently remove records from the Catalog.

The `--expired` option is to remove only expired records.

IMPORTANT

If the `--expired` option is omitted, then all records will be removed from the Catalog.

17.3.3. Non-Universal or Catalog Specific Removal

Using the Catalog Commands is convenient for achieving many goals such as removing expired records, but is not always the most efficient since not all Catalog implementation details are known. The Catalog API does not allow for every special nuance of a specific data store. Therefore, whether an administrator's data store is from Oracle, Solr, or any other vendor, the administrator should consult the specific Catalog implementation's documentation on the best method to remove metadata.

17.3.4. Automatic Catalog Backup

To backup local catalog records, a [Catalog Backup Plugin](#) is available. It is disabled by default for performance reasons.

It can be enabled and configured in the *Admin Console:

1. Navigate to the *Admin Console.
2. Select the *Catalog application.
3. Select the **Configuration** tab.
4. Select the [Backup Post-Ingest Plugin](#).

17.3.5. 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 process for changing catalog providers involves first exporting the metadata from the original catalog provider and ingesting it into another.

Exporting Metadata Out of a 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"
```

Ingesting Exported Metadata into a 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"
```

Translating 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.

Using

Get information here on the available user interfaces in DDF.

- [Using the Landing Page](#)
- [Using the Catalog Search](#)
- [Using the Standard Search](#)
- [Using Simple Search](#)

18. Using the Landing Page

The DDF Landing Page is the starting point for using DDF. It is accessible at <https://localhost:8993>.

18.1. Search DDF Button

The search button navigates to the Search UI, enabling catalog queries.

18.2. Data Source Availability

The data source availability pane provides a quick glance at the status of configured data sources.

18.3. Announcements

The announcements pane contains messages from system administrators.

19. Using the Catalog Search

Introduction: The Catalog UI represents the most advanced search interface available with DDF. It provides a metadata search and discovery, resource retrieval, and workspace management with a 3d or optional 2d map visualization.

NOTE

For more detail on any feature or button within the Catalog UI, click the **?** button in the upper right of the screen; then, hover over any item on the screen and a contextual tooltip will appear to define its purpose. To exit this mode, click the **?** again or press **escape**.

19.1. Accessing Catalog UI

The default location for the Catalog UI is <https://localhost:8993/search/catalog>

NOTE

Catalog UI Guest Users

If Guest access has been enabled, users not signed in to DDF (guest users) will have access to search functions, but all workspace configuration and settings will only exist locally and not be available for sharing.

The default view for the Catalog UI is the workspaces view. For other views or to return to workspaces, select the **navigation** icon in the upper-left corner of the Catalog UI.



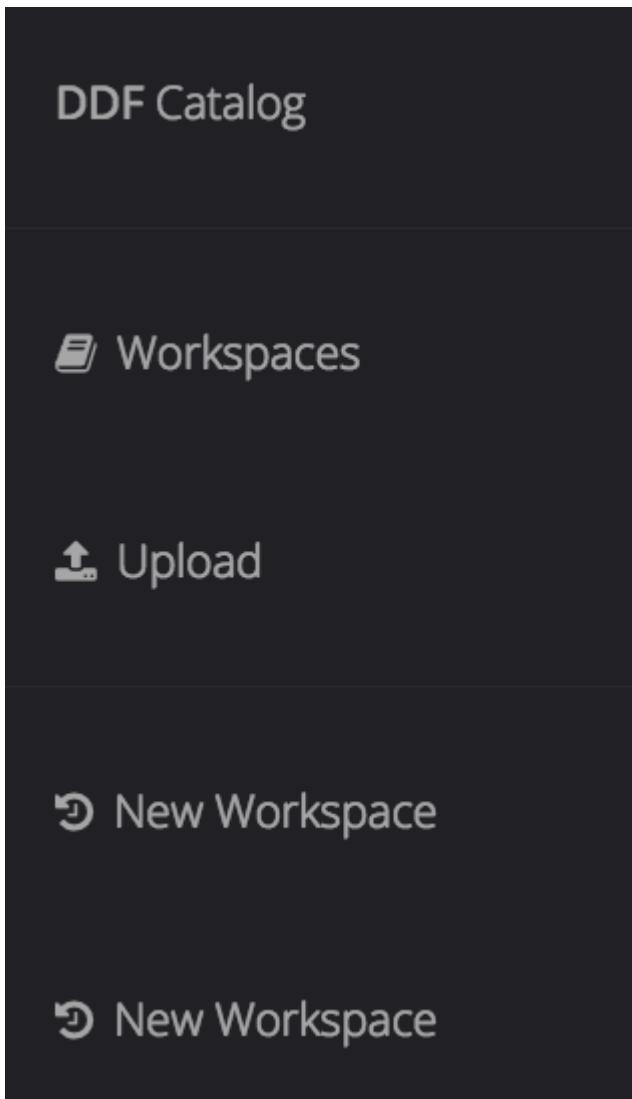


Figure 4. Select the desired view from the **navigator** menu.

19.2. Workspaces in Catalog UI

Within the Catalog UI, workspaces are collections of settings, searches, and bookmarks that can be shared between users and stored to be accessed repeatedly.

19.2.1. Creating a Workspace in Catalog UI

Before searching in DDF, at least one workspace must be created. Use an existing template or create a blank workspace.

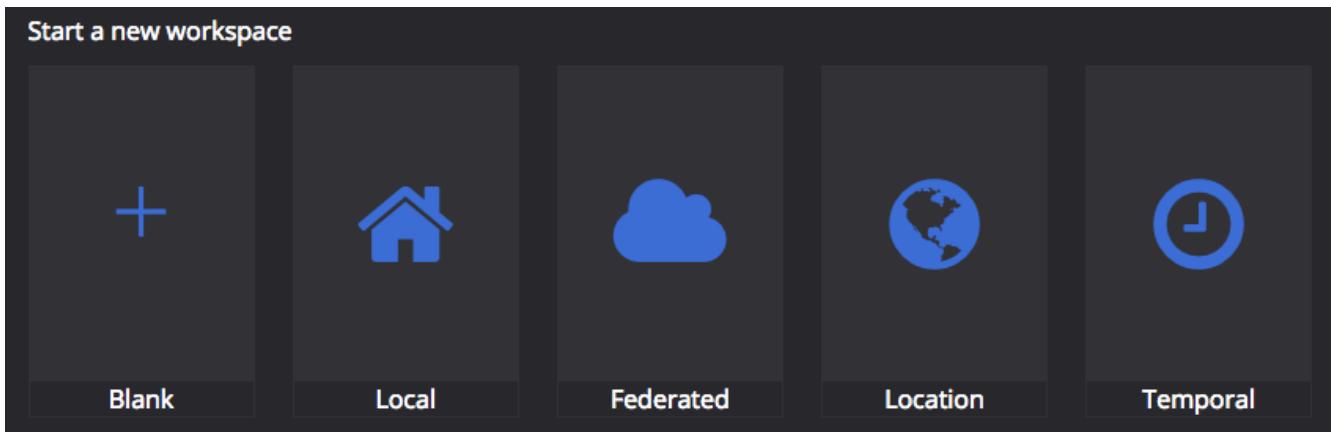


Figure 5. Default Workspace Templates

Local

an example of a local search.

Federated

an example of a example of a search across all connected sources.

Location

an example of a geographically constrained search.

Temporal

an example of a time-range search.

19.2.2. Configuring a Workspace in Catalog UI

Configure each workspace with queries and share options.

Adding searches

1. Select the default name of the workspace (**New Workspace** or template name) to change to a unique name.
2. If **Blank** workspace was selected, select **Add Search** to add the first search. Each workspace can have up to ten searches.
 - a. Change the name of the search.
 - b. Select **Basic** Search to select simple search criteria, such as **text**, **time**, and **location**.
 - c. Select **Advanced** Search to access a query-builder for more complex queries.

Navigator Menu Options

- **Workspaces:** View all available workspaces.
- **Upload:** Add new resources to catalog.
- **Open Workspaces:** [Workspace Name] (if applicable) Navigate to the [Workspace Name] workspace. Each open workspace is listed in this section of the menu.

Workspace Menu Options

-



Select the **Options** button (⋮) next to the Workspace name.

- **Save:** Save changes to the workspace.
- **Run All Searches:** Start all saved searches within this workspace.
- **Cancel All Searches:** Cancel all running searches.
- **Open in New Tab:** Opens this workspace in a separate tab.
- **View Sharing:** View and edit settings for sharing this workspace. (User must be signed in to share or view shared workspaces.)
- **View Details:** View the current details for a cloud-based workspace (User must be signed in).
- **Duplicate:** Create a copy of this workspace.
- **Subscribe:** Receive email notifications for search results on this workspace (If email notifications have been enabled).
- **Move to Trash:** Delete (archive) this workspace.

19.2.3. Sharing Workspaces

In the Catalog UI, workspaces can be shared between users at different levels of access as needed.

Share a Workspace

1. Select the Workspace.
2. Select the **Options** menu.
3. Select **View Sharing**.
 - a. To share by user role, set the drop-down menu to **Can Access** for each desired role. All users with that role will be able to view.
 - b. To share with an individual user, add their email to the email list.
4. Click **Apply**.

Remove Sharing on a Workspace

1. Select the Workspace.
2. Select the Options Menu (⋮).
 - a. To remove the workspace from users with specific roles, set the drop-down menu to **No Access** for that role.
 - b. To remove individual users, remove their email from the email list.
3. Select **View Sharing**.

19.3. Searching with Catalog UI

The Search pane has three tabs: **Searches**, **Results**, and **Bookmarks**.

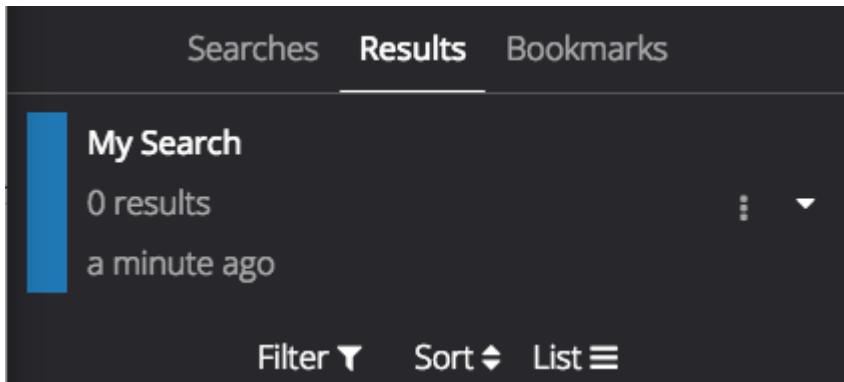


Figure 6. Search Pane Tabs

19.3.1. Searches Tab

View, edit, and add searches from the **Searches** tab.

Search Menu Options

- **Run:** Trigger this search to begin immediately.
- **Stop:** Stop this search.
- **Delete:** Remove this search
- **Duplicate:** Create a copy of this search as a starting point
- **Search Archived:** Execute the search, but specifically for archived results.
- **Search Historical:** Execute the search, but specifically for historical results.

Editing a Search

Existing searches and their settings can be updated by clicking on the search in the **Searches** tab of the workspace.

The editable components of a search are defined on these tabs:

- **Basic:** Define a [Text](#), [Spatial](#), [Temporal](#), or [Datatype](#) Search.
- **Advanced:** Advanced query builder to create more specific searches.
- **Settings:** Set preferences for **Sorting** and **Federation**.
- **Notifications:** Define the frequency at which to run this search and receive notification of new results
- **Status:** View details of recent instances of this search.

Editing Search Settings

Under the settings tab for each search, edit the default **Sorting** and **Federation** (which sources are searched).

Editing Search Notifications

Under the **Notifications** tab for a search, edit the **Frequency** settings to set the interval at which to run the search.

Viewing Search Status

The **Status** tab on the search preview displays the creation date and the sources successfully searched.

Intersecting Polygon Searches

If a self intersecting polygon is used to perform a geographic search, the polygon will be converted into a non-intersection one via a convex hull conversion. In the example below the blue line shows the original self intersecting search polygon and the red line shows the converted polygon that will be used for the search. The blue dot shows a search result that was not within the original polygon but was returned because it was within the converted polygon.

NOTE



Figure 7. Self Intersecting Polygon Conversion Example

19.3.2. Results Tab

Returned search results can be refined further, bookmarked, and/or downloaded from the **Results** tab. Result sets are color-coded by source as a visual aid. There is no semantic meaning to the colors assigned.



Figure 8. Results Tab Options

1. On the **Results** tab, select a search from the drop-down list.
2. Perform any of these actions upon the results list of the selected query:
 - a. Filter the result set locally (does not re-execute the search),
 - b. Customize results sorting (Default: Title in Ascending Order).
 - c. Toggle results view between **List** and **Gallery**.

19.3.3. Bookmarks Tab

Search Results that have been bookmarked can be filtered and sorted. They are denoted with a **star** icon. ()

Bookmarked Result Options

- **Add to Bookmarks:** Adds the result to bookmarks.
 - Changes to **Remove from Bookmarks** for bookmarked items: Removes the result from bookmarks.
- **Hide from Future Searches:** Adds to a list of results that will be hidden from future searches.
- **Expand Metocard View:** Navigates to a view that only focuses on this particular result.
- **Download:** Downloads the result's associated product directly to local machine.

19.4. Viewing Search Results

On the **Result Preview** pane, view a summary of information about the selected resource. The **Details** tab provides a list of specific metadata.

History

View revision history of this resource.

Associations

View or edit the relationship(s) between this resource and others in the catalog.

Quality

View the completeness and accuracy of the metadata for this resource.

Actions

Export the metadata/resource to a specific format.

Archive

Remove the selected record from standard search results.

Overwrite

Overwrite a record.

19.4.1. Editing Records

Records can be edited from the **Summary** or **Details** tabs.

19.4.2. Editing Associations on a Record

Update relationships between records through **Associations**.

1. Select the desired record from the **Results** tab.
2. Select the **Associations** tab.
3. Select **Edit**.
4. For a new association, select **Add Association**. (Only items in the current result set can be added as associations).
 - a. Select the related record from either the **Parent** or **Child** menu.
 - b. Select the type of relationship from the **Relationship** menu.
 - c. Select **Save**.
5. To edit an existing association, update the selections from the appropriate menu and select **Save**.

View a graphical representation of the associations by clicking the **graph** icon().

19.4.3. Viewing Revision History

View the complete revision history of a record.

1. Select the desired record from the **Results** tab.
2. Select the **History** tab.
 - a. Select a previous version from the list to preview.
 - b. If desired, select **Revert to Selected Version** to undo changes made after that revision.

19.4.4. Viewing Metadata Quality

View and fix issues with metadata quality in a record.

NOTE Correcting metadata issues may require heightened permissions.

1. Select the desired record from the **Results** tab.
2. Select **More**.
3. Select **Quality**.

4. A report is displayed showing any issues:

- a. Metacard Validation Issues.
- b. Attribute Validation Issues.

19.4.5. Exporting a Record

Export and view a resource and/or its metadata into another format.

1. Select the desired record from the **Results** tab.
2. Select **More**.
3. Select **Actions**.
4. Available export formats are listed. Select desired format.
5. Export opens in a new browser tab. Save, if desired.

19.4.6. Archiving a Record

To remove a record from active search results, archive it.

1. Select the desired record from the **Results** tab.
2. Select **More**.
3. Select **Archive**.
4. Confirm archiving the record.

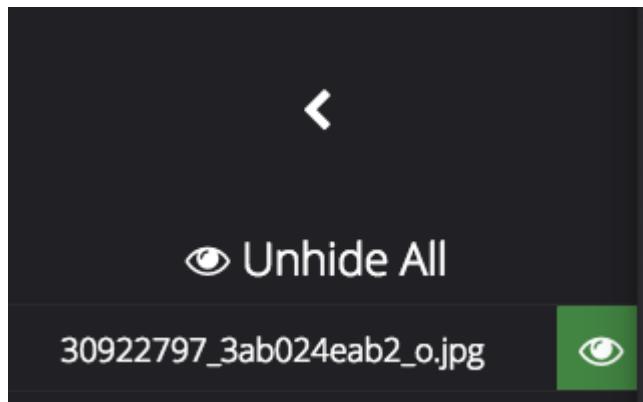
19.4.7. Restoring Archived Records

Restore an archived record to return it to active search results.

1. Select the **Search Archived** option from the query options menu.
2. Select the desired record from the **Results** tab.
3. Select **Archive**.
4. Confirm restoring the record.

Restore all archived records to active search results.

1. Select **Settings** icon () on navigation bar.
2. Select **Hidden**.
3. Remove selected from hidden list.
 - a. Or select **Remove All** to clear list.



19.4.8. Overwriting a Record

Replace a resource.

1. Select the desired record from the **Results** tab.
2. Select **More**.
3. Select **Overwrite**.

19.4.9. Catalog UI Settings

Customize the look and feel of the Catalog UI using the **Settings** () menu on the Navigation Bar.

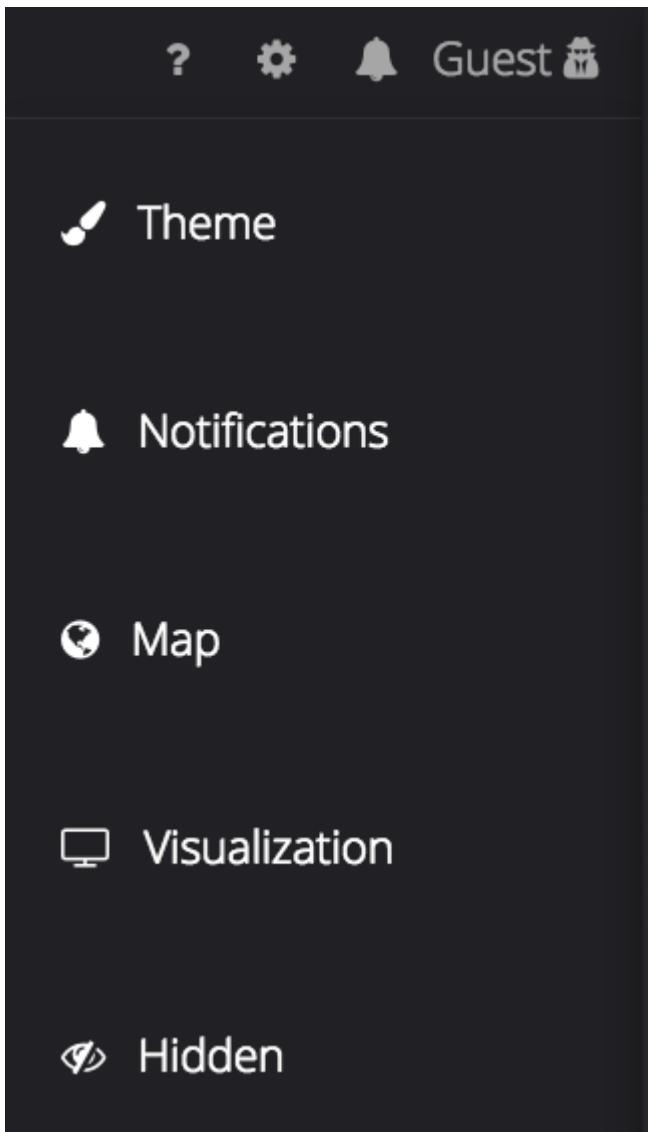


Figure 9. Settings Menu Options

- Theme: Visual options for page layout.
- Notifications: Select if notifications persist across sessions.
- Map: Select options for map layers.
- Visualization: Select options for viewing data collections
- Hidden: View or edit a list of records that have been hidden from search results.

19.4.10. Catalog UI Notifications

Notifications can be checked by clicking the **Notifications** icon () on the navigator bar.

20. Using the Standard Search

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

Located at the bottom of the left pane of the Search UI are two tabs: [Search](#) and [Workspaces](#). The Search tab contains basic fields to query the Catalog and other sources. The workspaces feature uses the same search criteria that are provided in the Search tab, and it also allows the user to create custom searches that can be saved for later execution. The right-side pane displays a map that, when records are found, shows the location of where the record was retrieved.

20.1. Search

The Standard Search UI allows users to search for records in the local Catalog and federated sources based on the criteria entered in the Search tab. After a search is executed, the UI provides results based on the defined criteria and detailed information about each result. Additionally, a user can save individual records that were returned by the query to a workspace, so they can be referenced in the future. The user can also save the search criteria to a workspace so the query can be run again.

20.1.1. Search Criteria

The Search UI queries a Catalog using the following criteria.

Criteria	Description
Text	Search by free text using the grammar of the underlying endpoint/Catalog.
Time	Search based on relative or absolute time of the created, modified, or effective date.
Location	Search by latitude/longitude or the USNG using a point-radius or bounding box.
Type	Search for specific content types.
Sorting	Sort results by relevance, distance, created time, modified time or effective time.
Additional Sources	Select All Sources or Specific Sources .
All Sources	Create an enterprise search. All federations are queried.
Specific Sources	Search a specific source(s). If a source is unavailable, it is displayed in red text.

20.1.2. Results

After a query is executed, the records matching the search criteria are automatically displayed in the Results pane.

Item	Description
Search criteria	Enhanced search toggle. Enables the enhanced search menu (see below). Allows users to filter and refine search and build more sophisticated queries.
Results	The number of records that were returned as a result of the query. Only the top 250 results are displayed, with the most relevant records displayed at the top of the list. If more than 250 results are returned, try narrowing the search criteria.

Item	Description
Search button	Navigates back to the Search pane.
Save button	Allows the user to select individual records to save.
Records list	Shows the results of the search. The following information is displayed for each record: Title – The title of the record is displayed in blue text. Select the title of the record to view more details. Source – The gray text displayed below the record title is the data source (e.g., ddf.distribution) and the amount of time that has passed since the record was modified (e.g., an hour ago).

20.1.3. Enhanced Search

The enhanced search menu allows more granular filtering of results and the ability to construct sophisticated queries.

Item	Description
Source	List of all sources searched with check boxes to allow users to refine searches to the most relevant sources.
Metadata Content Type	List of metadata content types found in the search, with the ability to select and deselect content type.
Query	The Query builder enables users to construct a very granular search query. The first drop-down menu contains the metadata elements in the search results, and the second contains operators based on the field selected (greater than, equals, contains, matchcase, before, after, etc.) Click the + to add further constraints to the query, or x to remove. Click Search to use the new query.
Search button	Executes an enhanced search on any new parameters that were specified or the query built above.

20.1.4. Record Summary

When an individual record is selected in the results list, the Record pane opens. When the Summary button is selected in the Record pane, the following information is displayed.

Item	Description
Results button	Navigates back to the original query results list.
Up and down arrows	Navigate through the records that were returned in the search. When the end or the beginning of the search results list is reached, the respective up or down arrow is disabled.
Details button	Opens the Details tab, which displays more information about the record.
Title	The title of the record is displayed in white font.

Item	Description
Source	The location that the metadata came from, which could be the local provider or a federated source.
Created time	When the record was created.
Modified time	Time since the record was last modified.
Locate button	Centers the map on the record's originating location.
Thumbnail	Depicts a reduced-size image of the original artifact for the current record, if available.
Download	A link to download the record. The size, if known, is indicated.

20.1.5. Record Details

When an individual record is selected in the results list, the Record pane opens. When the Details button is selected in the Record pane, the following information is displayed.

Item	Description
Results button	Navigates back to the original query results list.
Up and down arrows	Navigate through the records that were returned in the search. When the end or the beginning of the search results list is reached, the respective up or down arrow is disabled.
Summary button	Opens the Summary tab, which provides a high level overview of the result set.
Id	The record's unique identifier.
Source Id	Where the metadata was retrieved from, which could be the local provider or a federated source.
Title	The title of the record is displayed in white font.
Thumbnail	Depicts a reduced size image of the original artifact for the current record, if available.
Resource URI	Identifies the stored resource within the server.
Created time	When the record was created.
Metocard Content Type version	The version of the metadata associated with the record.
Metocard Type	The type of metocard associated with the record.

Item	Description
Metocard Content Type	The type of the metadata associated with the record.
Resource size	The size of the resource, if available.
Modified	Time since the record was last modified.
Download	When applicable, a download link for the product associated with the record is displayed. The size of the product is also displayed, if available. If the size is not available, N/A is displayed.
Metadata	Shows a representation of the metadata XML, if available.

20.2. Actions

Depending on the contents of the metocard, various actions will be available to perform on the metadata.

Troubleshooting: if no actions are available, ensure IP address is configured correctly under global configuration in Admin Console.

20.2.1. Save a Search

Saved searches are search criteria that are created and saved by a user. Each saved search has a name that was defined by the user, and the search can be executed at a later time or be scheduled for execution. Bookmarked records that the user elected to save for future use are returned as part of a search. These queries can be saved to a [workspace](#), which is a collection of searches and records created by a user. Complete the following procedure to create a saved search.

1. Select the Search tab at the bottom of the left pane.
2. Use the fields provided to define the [Search Criteria](#) for the query to be saved.
3. Select the **Save** button. The Select Workspace pane opens.
4. Type a name for the query in the **ENTER NAME FOR SEARCH** field.
5. Select a workspace in which to save the query, or create a workspace by typing a title for the new workspace in the **New Workspace** field.
6. Select the **Save** button.

The size of the product is based on the value in the associated metocard's resource-size attribute. This is defined when the metocard was originally created and may or may not be accurate. Often it will be set to N/A, indicating that the size is unknown or not applicable.

NOTE

However, if the administrator has enabled caching on DDF, and has installed the [catalog-core-resourcesizeplugin](#) PostQuery Plugin, and if the product has been retrieved, it has been cached and the size of the product can be determined based on the cached file's size. Therefore, subsequent query results that include that product will display an accurate size under the download link.

20.3. Workspaces

Each user can create multiple workspaces and assign each of them a descriptive name. Each workspace can contain multiple [saved searches](#) and contain multiple saved records. Workspaces are saved for each user and are loaded when the user logs in. Workspaces and their contents are persisted, so they survive if DDF is restarted. Within the Standard Search UI, workspaces are private and cannot be viewed by other users.

20.3.1. Create a Workspace

1. Select the Workspaces tab at the bottom of the Search UI's left pane. The Workspaces pane opens, which displays the existing workspaces that were created by the user. At the top of the pane, an option to **Add** and an option to **Edit** are displayed.
2. Select the **Add** button at the top of the left pane. A new workspace is created.
3. In the **Workspace Name** field, enter a descriptive name for the workspace.
4. Select the **Add** button. The Workspaces pane opens, which now displays the new workspace and any existing workspaces.
5. Select the name of the new workspace. The data (i.e., saved searches and records) for the selected workspace is displayed in the Workspace pane.
6. Select the + icon near the top of the Workspace pane to begin adding queries to the workspace. The Add/Edit Search pane opens.
7. Enter a name for the new query to be saved in the **QUERY NAME** field.
8. Complete the rest of the [Search Criteria](#).
9. Select the **Save & Search** button. The Search UI begins searching for records matching the criteria, and the new query is saved to the workspace. When the search is complete, the Workspace pane opens.
10. Select the name of the search to view the query results.
11. If necessary, in the Workspace pane, select the **Edit** button then select the pencil icon next to the name of a query to change the search criteria.
12. If necessary, in the Workspace pane, select the delete icon next to the name of a query to delete the query from the workspace.

20.4. Notifications

The Standard Search UI receives all notifications from DDF. These notifications appear as pop-up windows inside the Search UI to alert the user of an event of interest. To view all notifications, select the notification icon.

Currently, the notifications provide information about product retrieval only. After a user initiates a resource download, they receive periodic notifications that provide the progress of the download (e.g., the download completed, failed, or is being retried).

NOTE

A notification pop-up remains visible until it is dismissed or the browser is refreshed. Once a notification is dismissed, it cannot be retrieved again.

20.5. Activities

Similar to notifications, activities appear as pop-up windows inside the Search UI. Activity events include the status and progress of actions that are being performed by the user, such as searches and downloads. To view all activities, select the activity icon in the top-right corner of the window. 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.

20.6. Downloads

Downloads from the UI are currently managed by the user-specific browser's download manager. The UI itself does not have a built-in download manager utility.

20.7. Maps

The right side of the Search UI contains a map to locate search results on. There are three views for this map, 3D, 2D, and Columbus View. To choose a different view, select the map icon in the upper right corner. (The icon will change depending on current view selected.)

21. Using the Simple Search

The DDF Simple Search UI application provides a low-bandwidth option for searching records in the local Catalog (provider) and federated sources. Results are returned in HTML format.

21.1. Search

The **Input** form allows the user to specify keyword, geospatial, temporal, and type query parameters. It also allows the user to select the sources to search and the number of results to return.

21.1.1. Search Criteria

Enter one or more of the available search criteria to execute a query:

Keyword Search

A text box allowing the user to enter a textual query. This supports the use of (*) wildcards. If blank, the query will contain a contextual component.

Temporal Query

Select from **any**, **relative**, or **absolute**. Selecting **Any** results in no temporal restrictions on the query, selecting **relative** allows the user to query a period from some length of time in the past until now, and selecting **absolute** allows the user to specify a **start** and **stop** date range.

Spatial Search

Select from **any**, **point-radius**, and **bounding box**. Selecting **Any** results in no spatial restrictions on the query, selecting **point-radius** allows the user to specify a **lat/lon** and **radius** to search, and selecting a **bounding box** allows the user to specify an **eastern**, **western**, **southern** and **northern** boundary to search within.

Type Search

Select from **any**, or a specific type. Selecting **Any** results in no type restrictions on the query, and Selecting **Specific Types** shows a list of known content types on the federation, and allows the user to select a specific type to search for.

Sources

Select from **none**, **all sources**, or **specific sources**. Selecting **None** results in querying only the local provider, Selecting **All Sources** results in an enterprise search where all federations are queried, and selecting **Specific Sources** allows the user to select which sources are queried.

Results per Page

Select the number of results to be returned by a single query.

21.1.2. Results

The table of results shows the details of the results found, as well as a link to download the product if applicable.

Results Summary

Total Results

Total Number of Results available for this query. If there are more results than the number displayed per page then a page navigation links will appear to the right.

Pages

Provides page navigation, which generate queries for requesting additional pages of results.

Results Table

The Results table provides a preview of and links to the results. The table consists of these columns:

Title

Displays title of the metocard. This will be a link which can clicked to view the metocard in the Metocard View.

Source

Displays where the metadata came from, which could be the local provider or a federated source.

Location

Displays the WKT Location of the metocard, if available.

Time

Shows the Received (Created) and Effective times of the metocard, if available.

Thumbnail

Shows the thumbnail of the metocard, if available.

Download

A download link to retrieve the product associated with the metocard, when applicable, if available.

21.1.3. Result View

This view shows more detailed look at a result.

Back to Results Button

Returns the view back to the Results Table.

Previous & Next

Navigation to page through the results one by one.

Result Table

Provides the list of properties and associated values of a single search result.

Metadata

The metadata, when expanded, displays a tree structure representing the result's custom metadata.

Integrating

Integrators will use the existing applications to support their external frameworks. This section will provide details for finding, accessing and using the components of DDF.

22. Data and Metadata

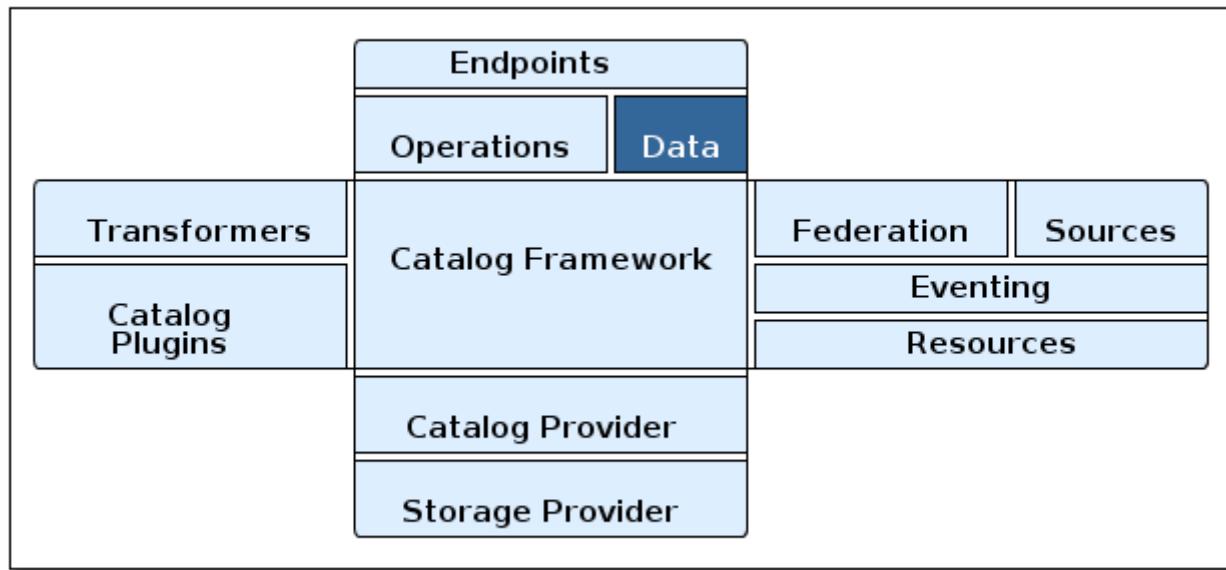


Figure 10. Catalog Architecture Diagram: Data

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

22.1. Metacards

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

22.1.1. Metocard Type

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

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

Default Metocard Type and Attributes

Most metacards within the system are created using the default metocard type or a metocard type based on the default type. The default metocard type of the system can be programmatically retrieved by calling `ddf.catalog.data.BasicTypes.BASIC_METACARD`. The name of the default **MetocardType** can be retrieved from `ddf.catalog.data.MetocardType.DEFAULT_METACARD_TYPE_NAME`.

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

Core Attributes

NOTE	It is highly recommended when referencing a default attribute name to use the <code>ddf.catalog.data.types.*</code> interface constants whenever possible. Mapping to a normalized taxonomy allows for higher quality transformations between different formats and for improved federation. This neutral profile facilitates improved search and discovery across disparate data types.
WARNING	Every <code>Source</code> should at the very least return an ID attribute according to Catalog API. Other fields may or may not be applicable, but a unique ID must be returned by a source.

Extensible Metacards

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

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

WARNING	Not all <code>Catalog Providers</code> support extensible metacards. Nevertheless, each Catalog Provider should at least have support for the default <code>MetocardType</code> ; i.e., it should be able to store and query on the attributes and attribute formats specified by the default metocard type. Catalog providers are neither expected nor required to store attributes that are not in a given metocard's type. Consult the documentation of the Catalog Provider in use for more information on its support of extensible metacards.
----------------	--

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

`Metocard` objects are extensible because they allow clients to store and retrieve standard and custom key/value Attributes from the `Metocard`. All `Metacards` must return a `MetocardType` object that includes an `AttributeDescriptor` for each `Attribute`, indicating it's key and value type. `AttributeType` support is limited to those types defined by the Catalog.

New `MetocardType` implementations can be made by implementing the `MetocardType` interface.

22.1.2. Metocard Type Registry

WARNING

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

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

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

`MetacardTypes` should be registered with the `MetacardTypeRegistry`. The `MetacardTypeRegistry` makes those `MetacardTypes` available to other DDF `CatalogFramework` components. Other components that need to know how to interpret metadata or metacards should look up the appropriate `MetacardType` from the registry. By having these `MetacardTypes` available to the `CatalogFramework`, these components can be aware of the custom attributes.

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

MetacardTypeRegistry Service Injection

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

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

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

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

22.1.3. Attributes

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

Attribute Types

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

Attribute Format

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

Table 19. Attribute Formats

AttributeFormat	Description
BINARY	Attributes of this attribute format must have a value that is a Java <code>byte[]</code> and <code>AttributeType.getBinding()</code> should return <code>Class<Array></code> of byte.
BOOLEAN	Attributes of this attribute format must have a value that is a Java boolean.
DATE	Attributes of this attribute format must have a value that is a Java date.
DOUBLE	Attributes of this attribute format must have a value that is a Java double.
FLOAT	Attributes of this attribute format must have a value that is a Java float.
GEOMETRY	Attributes of this attribute format must have a value that is a WKT-formatted Java string.
INTEGER	Attributes of this attribute format must have a value that is a Java integer.
LONG	Attributes of this attribute format must have a value that is a Java long.
OBJECT	Attributes of this attribute format must have a value that implements the serializable interface.
SHORT	Attributes of this attribute format must have a value that is a Java short.
STRING	Attributes of this attribute format must have a value that is a Java string and treated as plain text.
XML	Attributes of this attribute format must have a value that is a XML-formatted Java string.

Result

A single "hit" included in a query response.

A result object consists of the following:

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

22.1.4. Creating Metacards

The quickest way to create a `Metocard` is to extend or construct the `MetocardImpl` object. `MetocardImpl` is the most commonly used and extended `Metocard` implementation in the system because it provides a convenient way for developers to retrieve and set `Attributes` without having to create a new `MetocardType` (see below). `MetocardImpl` uses `BASIC_METACARD` as its `MetocardType`.

Limitations

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

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

Processing Metacards

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

Basic Types

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

Table 20. Basic Types

Name	Type	Description
<code>BASIC_METACARD</code>	<code>MetocardType</code>	Represents all required Metocard Attributes.
<code>BINARY_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.BINARY</code> .
<code>BOOLEAN_TYPE</code>	<code>AttributeType</code>	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.BOOLEAN</code> .

Name	Type	Description
DATE_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.DATE</code> .
DOUBLE_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.DOUBLE</code> .
FLOAT_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.FLOAT</code> .
GEO_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.GEOMETRY</code> .
INTEGER_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.INTEGER</code> .
LONG_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.LONG</code> .
OBJECT_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.OBJECT</code> .
SHORT_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.SHORT</code> .
STRING_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.STRING</code> .
XML_TYPE	AttributeType	A Constant for an <code>AttributeType</code> with <code>AttributeType.AttributeFormat.XML</code> .

22.2. JSON "Definition" Files.

WARNING

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

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

The following may be defined in a JSON definition file:

- Attribute Types

- Metocard Types
- Global Attribute Validators
- Default Attribute Values
- Attribute Injections

22.2.1. Definition File Format

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

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

22.2.2. Deploying Definition Files

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

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

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

22.3. Data Validation Services

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

22.3.1. Validation with Schematron

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

Custom schematron rulesets can be used to validate metocard 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.

Configuring Schematron Services

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:

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

22.4. Catalog Taxonomy

To facilitate data sharing while maximizing the usefulness of metadata, the attributes on resources are normalized into a common taxonomy that maps to attributes in the desired output format.

22.4.1. Attribute Categories

Core Attributes

Core attributes expected to be relevant to all metocard types.

Metocard Attributes

Attributes of the metocard itself.

History Attributes

History/versioning of the metocard.

Associations Attributes

Associations between products.

Date/Time Attributes

Temporal aspects about the resource.

Contact Attributes

Metadata about different kinds of people/groups/units/organizations that can be associated with a metocard.

Location Attributes

Location aspects about the resource.

Media Attributes

Metadata about media in general.

Security Attributes

Security of the resource and metadata.

Topic Attributes

The topic of the resource.

Validation Attributes

Validation issues with the metocard and/or resource.

Table 21. Core Attributes

Term	Definition	Datatype	Constraints	Example Value
title	A name for the resource. Dublin Core elements-title .	String	< 1024 characters	
source-id	ID of the source where the Metocard is cataloged. While this cannot be moved or renamed for legacy reasons, it should be treated as non-mappable, since this field is overwritten by the system when federated results are retrieved.	String	< 1024 characters	
metadata-content-type [deprecated] <i>see Media Attributes</i>	Content type of the resource.	String	< 1024 characters	
metadata-content-type-version [deprecated] <i>see Media Attributes</i>	Version of the metadata content type of the resource.	String	< 1024 characters	
metadata-target-namespace [deprecated] <i>see Media Attributes</i>	Target namespace of the metadata.	String	< 1024 characters	
metadata	Additional XML metadata describing the resource.	XML	A valid XML string per RFC 4825 (must be well-formed but not necessarily schema-compliant).	

Term	Definition	Datatype	Constraints	Example Value
location	The primary geospatial location of the resource.	Geometry	Valid Well Known Text (WKT) per http://www.opengeospatial.org/standards/wkt-crs <i>Coordinates must be in lon-lat coordinate order</i>	POINT(150 30)
expiration	The expiration date of the resource.	Date		
effective [deprecated]	The effective time of the event or resource represented by the metocard. Deprecated in favor of [created] and [modified] .	Date		
point-of-contact [deprecated]	The name of the point of contact for the resource. This is set internally to the user's subject and should be considered read-only to other DDF components.	String	< 1024 characters	
resource-uri	Location of the resource for the metocard.	String	Valid URI per RFC 2396	
resource-download-url	URL location of the resource for the metocard. This attribute provides a resolvable URL to the download location of the resource.	String	Valid URL per RFC 2396	
resource-size	Size in bytes of resource.	String	Although this type cannot be changed for legacy reasons, its value should always be a parseable whole number.	
thumbnail	The thumbnail for the resource in JPEG format.	Base 64 encoded binary string per RFC 4648	≤ 128 KB	

Term	Definition	Datatype	Constraints	Example Value
description	An account of the resource. Dublin Core elements-description.	String		
checksum	Checksum value for the primary resource for the metocard.	String	< 1024 characters	
checksum-algorithm	Algorithm used to calculate the checksum on the primary resource of the metocard.	String	< 1024 characters	
created	The creation date of the resource Dublin Core terms-created.	Date		
modified	The modification date of the resource Dublin Core terms-modified.	Date		
language	The language(s) of the resource. Dublin Core language.	List of Strings	Alpha-3 language code(s) per ISO_639-2.	
resource.derived-download-url	Download URL(s) for accessing the derived formats for the metocard resource.	List of Strings	Valid URL(s) per RFC 2396.	
resource.derived-uri	Location(s) for accessing the derived formats for the metocard resource.	List of Strings	Valid URI per RFC 2396	
datatype	The generic type(s) of the resource. Dublin Core terms-type and extended to include other DDF supported types.	List of Strings	Collection, Dataset, Event, Image, Interactive Resource, Service, Software, Sound, Text, Video, Document.	

Table 22. Metocard: Attributes in this group describe the metocard itself.

Term	Definition	Datatype	Constraints	Example Value
metocard.created	The creation date of the metocard.	Date		
metocard.modified	The modified date of the metocard.	Date		

Term	Definition	Datatype	Constraints	Example Value
metocard.owner	The email address of the metocard owner.	String	A valid email address per RFC 5322	
metocard-tags	Collections of data that go together, used for filtering query results. NOTE: these are system tags. For descriptive tags, Topic Attributes .	List of Strings	< 1024 characters per entry	

Table 23. History: Attributes in this group describe the history/versioning of the metocard.

Term	Definition	Datatype	Constraints	Example Value
metocard.version.id	Internal attribute identifier for which metocard this version is representing	String	A valid metocard ID (conventionally, a type 4 random UUID with hyphens removed).	70809f17782c42b8ba15747b86b50ebf
metocard.version.edited-by	Internal attribute identifying the editor of a history metocard.	String	A valid email address per RFC 5322	
metocard.version.versioned-on	Internal attribute for the versioned date of a metocard version.	Date		
metocard.version.action	Internal attribute for the action associated with a history metocard.	String	One of Deleted, Deleted-Content, Versioned, Versioned-Content	
metocard.version.tags	Internal attribute for the tags that were on the original metocard.	String		
metocard.version.type	Internal attribute for the metocard type of the original metocard.	String		
metocard.version.type-binary	Internal attribute for the serialized metocard type of the original metocard.	Binary		

Term	Definition	Datatype	Constraints	Example Value
metocard.version.resource-uri	Internal attribute for the original resource uri.	URI		

Table 24. Associations: Attributes in this group represent associations between products.

Term	Definition	Datatype	Constraints	Example Value
metocard.associations.derived	ID of one or more metacards derived from this metacard.	List of Strings	A valid metocard ID (conventionally, a type 4 random UUID with hyphens removed).	70809f17782c42b8ba15747b86b50ebf
metocard.associations.related	ID of one or more metacards related to this metacard.	List of Strings	A valid metocard ID (conventionally, a type 4 random UUID with hyphens removed).	70809f17782c42b8ba15747b86b50ebf
associations.external	One or more URI's identifying external associated resources.	List of Strings	A valid URI.	https://infocorp.org/wikia/reference

Table 25. DateTime: Attributes in this group reflect temporal aspects about the resource.

Term	Definition	Datatype	Constraints	Example Value
datetime.start	Start time(s) for the resource.	List of Dates		
datetime.end	End time(s) for the resource.	List of Dates		
datetime.name	A descriptive name for the corresponding temporal attributes. See datetime.start and datetime.end.	List of Strings	< 1024 characters per entry	

Table 26. Contact: Attributes in this group reflect metadata about different kinds of people/groups/units/organizations that can be associated with a metacard.

Term	Definition	Datatype	Constraints	Example Value
contact.creator-name	The name(s) of this metocard's creator(s).	List of Strings	< 1024 characters per entry	
contact.creator-address	The physical address(es) of this metocard's creator(s).	List of Strings	< 1024 characters per entry	
contact.creator-email	The email address(es) of this metocard's creator(s).	List of Strings	A valid email address per RFC 5322.	
contact.creator-phone	The phone number(s) of this metocard's creator(s).	List of Strings	< 1024 characters per entry	
contact.publisher-name	The name(s) of this metocard's publisher(s).	List of Strings	< 1024 characters per entry	
contact.publisher-address	The physical address(es) of this metocard's publisher(s).	List of Strings	< 1024 characters per entry	
contact.publisher-email	The email address(es) of this metocard's publisher(s).	List of Strings	A valid email address per RFC 5322.	
contact.publisher-phone	The phone number(s) of this metocard's publisher(s).	List of Strings	< 1024 characters per entry	
contact.contributor-name	The name of the contributor(s) to this metocard.	List of Strings	< 1024 characters per entry	
contact.contributor-address	The physical address(es) of the contributor(s) to this metocard.	List of Strings	< 1024 characters per entry	
contact.contributor-email	The email address(es) of the contributor(s) to this metocard.	List of Strings	A valid email address per RFC 5322.	
contact.contributor-phone	The phone number(s) of the contributor(s) to this metocard.	List of Strings	< 1024 characters per entry	
contact.point-of-contact-name	The name(s) of the point(s) of contact for this metocard.	List of Strings	< 1024 characters per entry	

Term	Definition	Datatype	Constraints	Example Value
contact.point-of-contact-address	The physical address(es) of a point(s) of contact for this metocard.	List of Strings	< 1024 characters per entry	
contact.point-of-contact-email	The email address(es) of the point(s) of contact for this metocard.	List of Strings	A valid email address per RFC 5322.	
contact.point-of-contact-phone	The phone number(s) of the point(s) of contact for this metocard.	List of Strings	< 1024 characters per entry	

Table 27. Location: Attributes in this group reflect location aspects about the resource.

Term	Definition	Datatype	Constraints	Example Value
location.altitude-meters	Altitude of the resource in meters.	List of Doubles	> 0	
location.country-code	One or more country codes associated with the resource.	List of Strings	ISO_3166-1 alpha-3 codes	
location.crs-code	Coordinate reference system code of the resource.	List of Strings	< 1024 characters per entry	EPSG:4326
location.crs-name	Coordinate reference system name of the resource.	List of Strings	< 1024 characters per entry	WGS 84

Table 28. Media: Attributes in this group reflect metadata about media in general.

Term	Definition	Datatype	Constraints	Example Value
media.format	The file format, physical medium, or dimensions of the resource. Dublin Core elements-format	String	< 1024 characters	txt, docx, xml - typically the extension or a more complete name for such, note that this is not the mime type
media.format-version	The file format version of the resource. Note that the syntax can vary widely from format to format.	String	< 1024 characters	POSIX, 2016, 1.0
media.bit-rate	The bit rate of the media, in bits per second.	Double		

Term	Definition	Datatype	Constraints	Example Value
media.frame-rate	The frame rate of the video, in frames per second.	Double		
media.frame-center	The center of the video frame.	Geometry	Valid Well Known Text (WKT)	
media.height-pixels	The height of the media resource in pixels.	Integer		
media.width-pixels	The width of the media resource in pixels.	Integer		
media.compression	The type of compression this media uses. EXIF STANAG 4559 NC, NM, C1, M1, I1, C3, M3, C4, M4, C5, M5, C8, M8	String	One of the values defined for EXIF Compression tag.	
media.bits-per-sample	The number of bits per image component.	Integer		
media.type (RFC 2046)	A two-part identifier for file formats and format content.	String	A valid mime-type per https://www.ietf.org/rfc/rfc2046.txt	application/json
media.encoding	The encoding format of the media.	List of Strings	< 1024 characters per entry	MPEG-2, RGB
media.number-of-bands	The number of spectral bands in the media.	Integer	The significance of this number is instrumentation-specific, but there are eight commonly recognized bands. https://en.wikipedia.org/wiki/Multispectral_image	
media.scanning-mode (MPEG2)	Indicate if progressive or interlaced scans are being applied.	String	PROGRESSIVE, INTERLACED	

Table 29. Security: Attributes in this group relate to security of the resource and metadata.

Term	Definition	Datatype	Constraints	Example Value
security.access-groups	Attribute name for storing groups to enforce access controls upon.	List of Strings	< 1024 characters per entry	
security.access-individuals	Attribute name for storing the email addresses of users to enforce access controls upon.	List of Strings	A valid email address per RFC 5322.	

Table 30. Topic: Attributes in this group describe the topic of the resource.

Term	Definition	Datatype	Constraints	Example Value
topic.category	A category code from a given vocabulary.	List of Strings	A valid entry from the corresponding controlled vocabulary.	
topic.keyword	One or more keywords describing the subject matter of the metocard or resource.	List of Strings	< 1024 characters per entry	
topic.vocabulary	An identifier of a controlled vocabulary from which the topic category is derived.	List of Strings	Valid URI per RFC 2396.	

Table 31. Validation: Attributes in this group identify validation issues with the metocard and/or resource.

Term	Definition	Datatype	Constraints	Example Value
validation-warnings	Textual description of validation warnings on the resource.	List of Strings	< 1024 characters per entry	
validation-errors	Textual description of validation errors on the resource.	List of Strings	< 1024 characters per entry	

23. Endpoints

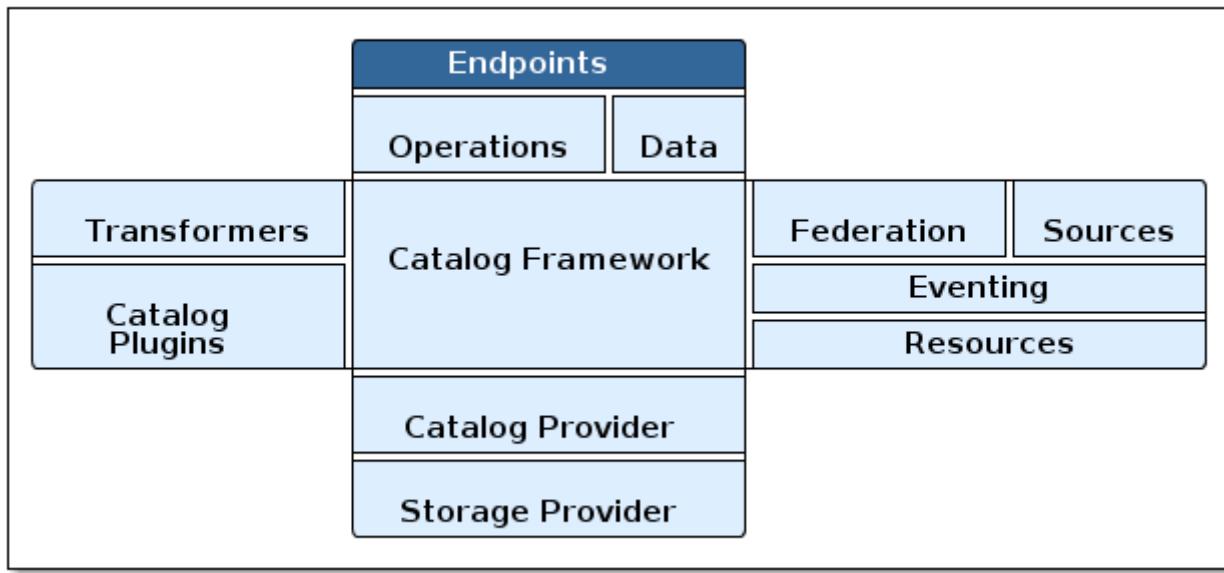


Figure 11. Endpoints

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

23.1. Available Endpoints

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

[Application Upload Endpoint](#)

Uploads new/upgraded applications to the system.

[Catalog REST Endpoint](#)

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

[CometD Endpoint](#)

Enables asynchronous search capabilities. *Implements CometD*.

[CSW Endpoint](#)

Enables a client to search collections of descriptive information (metadata) about geospatial data and services. *Implements Catalogue Services for Web (CSW) standard, XML-RPC, ISO 19115/ISO191119*.

[FTP Endpoint](#)

Provides a method for ingesting files directly into the DDF Catalog using the FTP protocol. *Implements FTP*.

[IdP Endpoint](#)

Enables configuration of DDF as an IdP server. *Implements SAML 2.0*.

[KML 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. [Implements Keyhole Markup Language](#).

[Metrics Endpoint](#)

Used by the [Metrics Collection Application](#) to report on system metrics.

[OpenSearch Endpoint](#)

Provides an endpoint that a client accesses to send query parameters and receive search results. [Implements JAX-RS](#), [CDR IPT BrokeredSearch], [CDR IPT OpenSearch](#), [CDR REST Brokered Search 1.1](#), [CDR REST Search v3.0](#), and [OpenSearch](#).

23.1.1. Application Upload Endpoint

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

Installing Application Upload Endpoint

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

Application Upload Endpoint

The Application endpoint is available at <<https://localhost:8993/services/application>>.

Configuring Application Upload Endpoint

No configuration necessary.

23.1.2. CometD Endpoint

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

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

Installing CometD Endpoint

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

Configuring CometD Endpoint

Configure the CometD endpoint from the Admin Console:

1. Navigate to the [Admin Console](#).
2. Select [Search UI](#) application.
3. Select [Configuration](#) tab.

Table 32. Search UI Endpoint

Name	Id	Type	Description	Default Value	Required
Disable Cache	<code>cacheDisabled</code>	Boolean	Disables use of cache.	false	false
Disable Normalization	<code>normalizationDisabled</code>	Boolean	Disables relevance and distance normalization.	false	false

Using CometD Endpoint

CometD Endpoint URL

```
https://localhost:8993/search/cometd
```

CometD Queries

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

Table 33. Query Parameters

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

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

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

Then the following example query could be executed:

```
/service/query
```

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

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

Query Request Examples

Enterprise Contextual Query

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

Multiple Site Temporal Absolute Query

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

Enterprise Spatial Bounding Box Query

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

Query Response Channel

The query responses are returned on the `/<id>` channel, which should be subscribed to in order to

retrieve the results. Replace `<id>` with the id that was used in the request. The [Subscribing to Notifications](#) section details how to subscribe to a CometD channel.

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

Table 34. Query Response Message Format

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

Query Response Examples

Example Query Response

```
{
  "data": {
    "hits": 1,
    "metocard-types": {
      "ddf.metocard": {...}
    }
  }
}
```

```

},
"id": "6f0e04e9-acd1-4935-b9dd-c83e770a36d5",
"results": [
{
  "metacard": {
    "is-resource-local": false,
    "cached": "2016-07-13T19:22:18.220+0000",
    "geometry": {
      "coordinates": [
        -84.415337,
        42.729925
      ],
      "type": "Point"
    },
    "type": "Feature",
    "actions": [...],
    "properties": {
      "thumbnail": "...",
      "metadata": "<?xml version=\"1.0\" encoding=\"UTF-8\n\"?><metadata>...</metadata>",
      "resource-size": "362417",
      "created": "2010-06-10T12:07:26.000+0000",
      "resource-uri": "content/faade630a2a247468ca9a9b57303b437",
      "metacard-tags": [
        "resource"
      ],
      "checksum-algorithm": "Adler32",
      "metadata-content-type": "image/jpeg",
      "metacard-type": "ddf.metacard",
      "resource-download-url":
      "https://localhost:8993services/catalog/sources/ddf.distribution/faade630a2a247468ca9a9b57303b437?transform=resource",
        "title": "example.jpg",
        "source-id": "ddf.distribution",
        "effective": "2016-07-13T19:22:06.966+0000",
        "point-of-contact": "",
        "checksum": "dc7337c5",
        "modified": "2010-06-10T12:07:26.000+0000",
        "id": "faade630a2a247468ca9a9b57303b437"
      }
    }
  }
],
"status": [
{
  "hits": 1,
  "elapsed": 453,
  "reasons": [],
  "id": "ddf.distribution",
  "state": "SUCCEEDED",
  "results": 1
}
]
}

```

```

        },
    ],
    "successful": true
},
{
    "successful": true
},
{
    "channel": "/service/query",
    "id": "142",
    "successful": true
}

```

CometD Notifications

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

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

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

Using CometD Notifications

NOTE

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

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

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

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

For example,
`http://DDF_HOST:8181/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1w1ql079j6tscii19jszwp9s2i55` notifications contain the following information:

Table 35. Notification Contents

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

Example: Notification Message

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

Receive Notifications

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

Notification Events

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

Persistence of Notifications

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

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

Notification Operations Channel

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

Table 36. Operation Format

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

Example: Notification Operation Request

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

Activity Events Channel

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

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

Table 37. CometD Activity Format

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

Property	Description	Value Type
<code>operations</code>	Map of operations that can be performed on this activity. If the value is a URL, the client should invoke the URL as a result of the user invoking the activity operation. 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: * <code>cancel</code> * <code>download</code> * <code>remove</code>	JSON Map
<code>progress</code>	Percentage value of activity completion	String (Integer between 0 - 100 followed by a %)
<code>status</code>	Enumerated value that displays the current state of the activity	String + * <code>STARTED</code> * <code>RUNNING</code> * <code>COMPLETED</code> * <code>STOPPED</code> * <code>PAUSED</code> * <code>FAILED</code>
<code>timestamp</code>	Time that the activity update was sent	Date-Time
<code>title</code>	User-readable title for the activity update	String
<code>subject</code>	User who started the activity	String
<code>bytes</code>	Number of bytes the activity consumed (upload or download)	Positive Integer
<code>session</code>	The session ID of the user/subject	String
<code>Custom Value</code>	Additional keys can be inserted by the component sending the activity notification	Any JSON Type

Example: Activity update with custom 'bytes' field

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

Activity Operations Channel

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

Table 38. CometD Activity Format

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

Example: Activity Operation Request Message

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

23.1.3. Catalog REST Endpoint

The Catalog REST Endpoint allows clients to perform CRUD operations on the Catalog using REST, a simple architectural style that performs communication using HTTP. The URL exposing the REST functionality is located at [http://\\${public_endpoint}/services/catalog](http://${public_endpoint}/services/catalog).

Installing the Catalog REST Endpoint

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

Configuring the Catalog REST Endpoint

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

Using the Catalog REST Endpoint

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

Operation	HTTP Request	Details	Example URL
create	HTTP POST	HTTP request body contains the input to be ingested. <code><input transformer></code> is the name of the transformer to use when parsing metadata (optional).	<code>http://localhost:8181/services/catalog?transformer=<input transformer></code>
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. <code><metacardId></code> is the <code>Metacard.ID</code> of the metacard to be updated and <code><input transformer></code> is the name of the transformer to use when parsing an override metadata attribute (optional).	<code>http://localhost:8181/services/catalog/<metacardId>?transformer=<input transformer></code>
delete	HTTP DELETE	The ID of the Metacard to be deleted is appended to the end of the URL. <code><metacardId></code> is the <code>Metacard.ID</code> of the metacard to be deleted.	<code>http://localhost:8181/services/catalog/<metacardId></code>

Operation	HTTP Request	Details	Example URL
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. <code><metacardId></code> is the Metacard.ID of the metacard to be retrieved.	http://localhost:8181/services/catalog/<metacardId>
federated read	HTTP GET	The SOURCE ID of a federated source is appended to the URL before the ID of the Metacard to be retrieved is appended to the end. <code><sourceId></code> is the FEDERATED SOURCE ID and <code><metacardId></code> is the Metacard.ID of the Metacard to be retrieved.	http://localhost:8181/services/catalog/sources/<sourceId>/<metacardId>
sources	HTTP GET	Retrieves information about federated sources, including <code>sourceId</code> , <code>availability</code> , <code>contentTypes</code> , and <code>version</code> .	http://localhost:8181/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 metocard 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.)

The REST endpoint can be used to upload resources as attachments. The `create` and `update` methods both support the multipart mime format. If only a single attachment exists, it will be interpreted as a resource to be parsed, which will result in a metocard and resource being stored in the system.

If multiple attachments exist, then the REST endpoint will assume that 1 attachment is the actual resource (attachment should be named `parse.resource`) and the other attachments are overrides of metocard attributes (attachment names should follow metocard attribute names). In the case of the metadata attribute, it is possible to also have the system transform that metadata and use the results of that to override the metocard that would be generated from the resource (attachment should be named `parse.metadata`).

For example:

```

POST /services/catalog?transform=xml HTTP/1.1
Host: localhost:8993
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
Cache-Control: no-cache

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

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

-----WebKitFormBoundary7MA4YWxkTrZu0gW--

```

Metocard 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 Metocard Transform

```
http://localhost:8181/services/catalog/<metacardId>?transform=<TRANSFORMER_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>

See [Metocard Transformers](#) for details on metocard transformers.

23.1.4. CSW Endpoint

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

Installing CSW Endpoint

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

Configuring CSW Endpoint

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

CSW Endpoint URL

The CSW endpoint is accessible from <https://localhost:8993/services/csw>.

CSW Endpoint Operations

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 (Key-Value Pairs) Encoding

```
https://localhost:8993/services/csw?service=CSW&version=2.0.2&request=GetCapabilities
```

GetCapabilities HTTP POST

The **HTTP POST** form of **GetCapabilities** operates on the root CSW endpoint URL (<https://localhost:8993/services/csw>) with an XML message body that is defined by the **GetCapabilities** element of the [CSW-Discovery XML Schema](#).

GetCapabilities XML Request

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

GetCapabilities Sample Response (application/xml)

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

```

<ows:ServiceIdentification>
    <ows:Title>Catalog Service for the Web</ows:Title>
    <ows:Abstract>DDF CSW Endpoint</ows:Abstract>
    <ows:ServiceType>CSW</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
</ows:ServiceIdentification>
<ows:ServiceProvider>
    <ows:ProviderName>DDF</ows:ProviderName>
    <ows:ProviderSite/>
    <ows:ServiceContact/>
</ows:ServiceProvider>
<ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get ns2:href="https://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:Operation>
</ows:OperationsMetadata>

```

```

<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>Greater Than</ogc:ComparisonOperator>
            <ogc:ComparisonOperator>Greater Than Equal To</ogc:ComparisonOperator>
            <ogc:ComparisonOperator>Less Than</ogc:ComparisonOperator>
            <ogc:ComparisonOperator>Less Than Equal To</ogc:ComparisonOperator>
            <ogc:ComparisonOperator>Equal To</ogc:ComparisonOperator>
            <ogc:ComparisonOperator>Not Equal To</ogc:ComparisonOperator>
        </ogc:ComparisonOperators>
    </ogc:Scalar_Capabilities>
    <ogc:Id_Capabilities>
        <ogc:EID/>
    </ogc:Id_Capabilities>
</ogc:Filter_Capabilities>
</csw:Capabilities>

```

DescribeRecord Operation

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

Namespace

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

Service

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

Version

The version of the service being used (2.0.2).

OutputFormat

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

SchemaLanguage

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

DescribeRecord HTTP GET

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

DescribeRecord KVP (Key-Value Pairs) Encoding

```
https://localhost:8993/services/csw?service=CSW&version=2.0.2&request=DescribeRecord&NAMESPACE=xmlns(http://www.opengis.net/cat/csw/2.0.2)&outputFormat=application/xml&schemaLanguage=http://www.w3.org/XML/Schema
```

DescribeRecord HTTP POST

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

DescribeRecord XML Request

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

DescribeRecord Sample Response (application/xml)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:DescribeRecordResponse xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml=
"http://www.opengis.net/gml" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:ns6="http://www.w3.org/2001/SMIL20/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/" xmlns:ns9=
"http://www.w3.org/2001/SMIL20/Language" xmlns:ns10="http://www.w3.org/2001/XMLSchema-
instance" ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-
publication.xsd">
    <csw:SchemaComponent targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
schemaLanguage="http://www.w3.org/XMLSchema">
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault=
"qualified" id="csw-record" targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
version="2.0.2">
            <xsd:annotation>
                <xsd:appinfo>
                    <dc:identifier>
http://schemas.opengis.net/csw/2.0.2/record.xsd</dc:identifier>

                    </xsd:appinfo>
                    <xsd:documentation xml:lang="en">
                        This schema defines the basic record types that must be supported
                        by all CSW implementations. These correspond to full, summary, and
                        brief views based on DCMI metadata terms.
                    </xsd:documentation>

                    </xsd:annotation>
                    <xsd:import namespace="http://purl.org/dc/terms/" schemaLocation="rec-
dcterms.xsd"/>
                    <xsd:import namespace="http://purl.org/dc/elements/1.1/" schemaLocation=
"rec-dcmes.xsd"/>
                    <xsd:import namespace="http://www.opengis.net/ows" schemaLocation=
"../../ows/1.0.0/owsAll.xsd"/>
                    <xsd:element abstract="true" id="AbstractRecord" name="AbstractRecord"
type="csw:AbstractRecordType"/>
                    <xsd:complexType abstract="true" id="AbstractRecordType" name=
"AbstractRecordType"/>
                </xsd:documentation>
            </xsd:annotation>
        </xsd:schema>
    </csw:SchemaComponent>
</csw:DescribeRecordResponse>
```

```

<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>
    </xsd:complexType>

```

```

</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/XMLSchema">
    <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/XMLSchema"
  xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
  <TypeName>csw:Record</TypeName>
</DescribeRecord>

```

DescribeRecord Sample Response (application/xml)

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:DescribeRecordResponse xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml=
"http://www.opengis.net/gml" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:ns6="http://www.w3.org/2001/SMIL20/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/" xmlns:ns9=
"http://www.w3.org/2001/SMIL20/Language" xmlns:ns10="http://www.w3.org/2001/XMLSchema-
instance" ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-
publication.xsd">
  <csw:SchemaComponent targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
  schemaLanguage="http://www.w3.org/XMLSchema">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault=
    "qualified" id="csw-record" targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
    version="2.0.2">
      <xsd:annotation>
        <xsd:appinfo>
          <dc:identifier>
            http://schemas.opengis.net/csw/2.0.2/record.xsd</dc:identifier>
          </xsd:appinfo>
        <xsd:documentation xml:lang="en">
          This schema defines the basic record types that must be supported
          by all CSW implementations. These correspond to full, summary, and
          brief views based on DCMI metadata terms.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:schema>
  </csw:SchemaComponent>
</csw:DescribeRecordResponse>

```

```

</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:complexType>
                        <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:complexType>
                        <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:complexContent>

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

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

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

```

GetRecords Operation

The **GetRecords** operation is the principal means of searching the catalog. The matching entries may be included with the response. The client may assign a **requestId** (absolute URI). A distributed search is performed if the **DistributedSearch** element is present and the catalog is a member of a federation. Profiles may allow alternative query expressions. There are two types of request types: one for **GET** and one for **POST**. Each request has the following common data parameters:

Namespace

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

Service

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

Version

The version of the service being used (2.0.2).

OutputFormat

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

OutputSchema

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 (Key-Value Pairs) Encoding

```
https://localhost:8993/services/csw?service=CSW&version=2.0.2&request=GetRecords&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPACE=xm&lns(csw=http://www.opengis.net/cat/csw/2.0.2)&resultType=results&typeNames=csw:Record&ElementSetName=brief&ConstraintLanguage=CQL_TEXT&constraint=AnyText Like '%25'
```

GetRecords HTTP POST

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

GetRecords XML Request

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

GetRecords Specific Source

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

NOTE

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

GetRecords XML Request

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

GetRecords Sample Response (application/xml)

```
<csw:GetRecordsResponse version="2.0.2" xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows" xmlns:xs
    ="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <csw:SearchStatus timestamp="2014-02-19T15:33:44.602-05:00"/>
        <csw:SearchResults numberOfRecordsMatched="41" numberOfRecordsReturned="4"
        nextRecord="5" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">
            <csw:SummaryRecord>
                <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>
                <dc:title>Product10</dc:title>
```

```

<dc:type>pdf</dc:type>
<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>034cc757abd645f0abeb6acaccfe194de</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 Sample Response (application/xml)

```
<?xml version='1.0' encoding='UTF-8'?>
<csw:GetRecordsResponse xmlns:dct="http://purl.org/dc/terms/" xmlns:xml=
"http://www.w3.org/XML/1998/namespace" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:ows="http://www.opengis.net/ows" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0.2">
    <csw:SearchStatus timestamp="2016-03-23T11:31:34.531-06:00"/>
    <csw:SearchResults numberofrecordsMatched="7" numberofrecordsReturned="1"
nextRecord="2" recordSchema="http://www.isotc211.org/2005/gmd" elementSet="summary">
        <MD_Metadata xmlns="http://www.isotc211.org/2005/gmd" xmlns:gco=
"http://www.isotc211.org/2005/gco">
            <fileIdentifier>
                <gco:CharacterString>
d5f6acd5ccf34d18af5192c38a276b12</gco:CharacterString>
            </fileIdentifier>
            <hierarchyLevel>
                <MD_ScopeCode codeListValue="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>
                    </EX_GeographicBoundingBox>
                </geographicElement>
            </EX_Extent>
        </extent>
    </MD_DataIdentification>
</identificationInfo>

```

```

        </southBoundLatitude>
        <northBoundLatitude>
            <gco:Decimal>32.32333</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 **GetRecordById** operation request retrieves the default representation of catalog records using their identifier. This operation presumes that a previous query has been performed in order to obtain the identifiers that may be used with this operation. For example, records returned by a **GetRecords** operation may contain references to other records in the catalog that may be retrieved using the **GetRecordById** operation. This operation is also a subset of the **GetRecords** operation and is included as a convenient short form for retrieving and linking to records in a catalog.

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

partial downloads. This would typically occur at the request of a browser when a download was prematurely terminated.

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

Namespace

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

Service

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

Version

The version of the service being used (2.0.2).

OutputFormat

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

OutputSchema

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 KVP (Key-Value Pairs) Encoding*

```
https://localhost:8993/services/csw?service=CSW&version=2.0.2&request=GetRecordById&NAMESPACE=xmlns="http://www.opengis.net/cat/csw/2.0.2"&ElementSetName=full&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&id=fd7ff1535dfe47db8793b550d4170424,ba908634c0eb439b84b5d9c42af1f871
```

GetRecordById *HTTP POST*

```
<GetRecordById xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  service="CSW"
  version="2.0.2"
  outputFormat="application/xml"
  outputSchema="http://www.opengis.net/cat/csw/2.0.2"
  xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
  ../../../../../../csw/2.0.2/CSW-discovery.xsd">
  <ElementSetName>full</ElementSetName>
  <Id>182fb33103414e5cbb06f8693b526239</Id>
  <Id>c607440db9b0407e92000d9260d35444</Id>
</GetRecordById>
```

GetRecordByIdResponse Sample Response (application/xml)

```
<csw:GetRecordByIdResponse xmlns:dc="http://purl.org/dc/elements/1.1/"  
    xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows"  
    xmlns:xs="http://www.w3.org/2001/XMLSchema"  
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
    <csw:Record>  
        <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>  
        <dct:bibliographicCitation>182fb33103414e5cbb06f8693b526239</dct:bibliographicCitation>  
        <dc:title>Product10</dc:title>  
        <dct:alternative>Product10</dct:alternative>  
        <dc:type>pdf</dc:type>  
        <dc:date>2014-02-19T15:22:51.563-05:00</dc:date>  
        <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
        <dct:created>2014-02-19T15:22:51.563-05:00</dct:created>  
        <dct:dateAccepted>2014-02-19T15:22:51.563-05:00</dct:dateAccepted>  
        <dct:dateCopyrighted>2014-02-19T15:22:51.563-05:00</dct:dateCopyrighted>  
        <dct:dateSubmitted>2014-02-19T15:22:51.563-05:00</dct:dateSubmitted>  
        <dct:issued>2014-02-19T15:22:51.563-05:00</dct:issued>  
        <dc:source>ddf.distribution</dc:source>  
        <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
            <ows:LowerCorner>20.0 10.0</ows:LowerCorner>  
            <ows:UpperCorner>20.0 10.0</ows:UpperCorner>  
        </ows:BoundingBox>  
    </csw:Record>  
    <csw:Record>  
        <dc:identifier>c607440db9b0407e92000d9260d35444</dc:identifier>  
        <dct:bibliographicCitation>c607440db9b0407e92000d9260d35444</dct:bibliographicCitation>  
        <dc:title>Product03</dc:title>  
        <dct:alternative>Product03</dct:alternative>  
        <dc:type>pdf</dc:type>  
        <dc:date>2014-02-19T15:22:51.563-05:00</dc:date>  
        <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
        <dct:created>2014-02-19T15:22:51.563-05:00</dct:created>  
        <dct:dateAccepted>2014-02-19T15:22:51.563-05:00</dct:dateAccepted>  
        <dct:dateCopyrighted>2014-02-19T15:22:51.563-05:00</dct:dateCopyrighted>  
        <dct:dateSubmitted>2014-02-19T15:22:51.563-05:00</dct:dateSubmitted>  
        <dct:issued>2014-02-19T15:22:51.563-05:00</dct:issued>  
        <dc:source>ddf.distribution</dc:source>  
        <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
            <ows:LowerCorner>6.0 3.0</ows:LowerCorner>  
            <ows:UpperCorner>6.0 3.0</ows:UpperCorner>  
        </ows:BoundingBox>  
    </csw:Record>  
</csw:GetRecordByIdResponse>
```

Table 39. CSW Record to Metocard 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
dct:educationLevel				0-n
dct:extent				0-n
dct:hasFormat				0-n

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

Transaction Operations

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

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

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>2dbcfa3f3e24e3e8f68c50f5a98a4d1</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>
```

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>
```

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 (Key-Value Pairs) Encoding

```
http://<DDF_HOST>:<DDF_PORT>/services/csw/subscription?service=CSW&version=2.0.2&request=GetRecords&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPACE=xmlns(csw=http://www.opengis.net/cat/csw/2.0.2)&resultType=results&typeNames=csw:Record&elementSetName=brief&responseHandler=https%3A%2F%2Fsome.ddf%2Fservices%2Fcsw%2Fsubscription%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 used to update an existing subscription. It is the same as the **POST**, except the **requestid** URN is appended to the url.

Subscription GetRecords XML Request

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

Subscription GetRecords XML Response

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2008-09-28T18:49:45" xmlns=
"http://www.opengis.net/cat/csw/2.0.2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../../../../csw/2.0.2/CSW-
discovery.xsd">
<EchoedRequest>
  <GetRecords
    requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
    service="CSW"
    version="2.0.2"
    maxRecords="4"
    startPosition="1"
    resultType="results"
    outputFormat="application/xml"
    outputSchema="urn:catalog:metacard">
    <ResponseHandler>
      https://some.ddf/services/csw/subscription/event</ResponseHandler>
      <Query 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/XMLSchema-instance"
                               xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 ../../../../../../csw/2.0.2/CSW-discovery.xsd">
<EchoedRequest>
  <GetRecords
    requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
    service="CSW"
    version="2.0.2"
    maxRecords="4"
    startPosition="1"
    resultType="results"
    outputFormat="application/xml"
    outputSchema="urn:catalog:metocard">
    <ResponseHandler>
https://some.ddf/services/csw/subscription/event</ResponseHandler>
    <Query typeNames="Record">
      <ElementSetName>summary</ElementSetName>
      <Constraint version="1.1.0">
        <ogc:Filter>
          <ogc:PropertyIsLike wildCard "%" singleChar "_" escapeChar="\\">
            <ogc:PropertyName>xml</ogc:PropertyName>
            <ogc:Literal>%</ogc:Literal>
          </ogc:PropertyIsLike>
        </ogc:Filter>
      </Constraint>
    </Query>
  </GetRecords>
</EchoedRequest>
<RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

23.1.5. FTP Endpoint

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

Installing the FTP Endpoint

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

To install:

- Navigate to Admin Console.
- Navigate to **Catalog** application.
- Navigate to **Features** tab.
- Install the **catalog-core-ftp** feature.

Configuring the FTP Endpoint

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

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

Table 40. FTP Endpoint

Name	Id	Type	Description	Default Value	Required
FTP Port Number	<code>port</code>	Integer	The port number for the FTP server to listen on.	8021	true
Client Authentication	<code>clientAuth</code>	String	Whether or not client authentication is required or wanted. A value of "Need" requires client auth, a value of "Want" leaves it up to the client.	want	true

Using FTP Endpoint

FTP Endpoint URL

```
ftp://localhost:8021/
```

The FTP endpoint supports the **PUT**, **MPUT**, **DELE**, **RETR**, **RMD**, **APPE**, **RNTO**, **STOU**, and **SITE** operations.

The FTP endpoint supports files being uploaded as a dot-file (e.g., `.foo`) and then being renamed to the final filename (e.g., `some-file.pdf`). The endpoint will complete the ingest process when the rename command is sent.

From Code:

Custom Ftplets can be implemented by extending the `DefaultFtpplet` class provided by Apache FTP Server. Doing this will allow custom handling of various FTP commands by overriding the methods of the `DefaultFtpplet`. Refer to <https://mina.apache.org/ftpserver-project/ftpplet.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**. If FTPS is enabled with 2-way TLS, a client that supports client authentication is required.

23.1.6. KML Endpoint

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

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.)

Installing the KML Endpoint

The KML Network Link Endpoint is installed by default with a standard installation as part of the Spatial application.

Configuring the KML Endpoint

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

Table 41. Spatial KML Endpoint

Name	Id	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)		false
Icons Location	iconLoc	String	Location of icons for the KML endpoint		false

Name	Id	Type	Description	Default Value	Required
Description	<code>description</code>	String	Description of this NetworkLink. Enter a short description of what this NetworkLink provides.		false
Web Site	<code>webSite</code>	String	URL of the web site to be displayed in the description.		false
Logo	<code>logo</code>	String	URL to the logo to be displayed in the description.		false
Visible By Default	<code>visibleByDefault</code>	Boolean	Check if the source NetworkLinks should be visible by default.	false	false
Max Number of Results	<code>maxResults</code>	Integer	The maximum number of results that should be returned from each layer.	100	false

Using the KML Endpoint

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

```
https://localhost:8993/services/catalog/kml
```

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

Example Output from KML Endpoint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:ns2="http://www.google.com/kml/ext/2.2"
      xmlns:ns3="http://www.w3.org/2005/Atom" xmlns:ns4="urn:oasis:names:tc:cii: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>
```

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

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

Return KML Icon

```
https://localhost:8993/services/catalog/kml/icons?<icon-name>
```

23.1.7. Metrics Endpoint

NOTE EXPERIMENTAL

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

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

Installing the Metrics Endpoint

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

Configuring the Metrics Endpoint

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

Using the Metrics Endpoint

Metrics Endpoint URL

```
https://localhost:8993/services/internal/metrics/catalogQueries.png?startDate=2013-03-31T06:00:00-07:00&endDate=2013-04-01T11:00:00-07:00
```

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

Table 42. Metrics Endpoint URL Options

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

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

Metrics Data Supported Formats

The metric's historical data can be displayed in several formats, including PNG , a CSV file, an Excel .xls file, a PowerPoint .ppt file, an XML file, and a JSON file. The PNG, CSV, and XLS formats are accessed via hyperlinks provided in the Metrics tab web page. The PPT, XML, and JSON formats are accessed by specifying the format in the custom URL, e.g., <http://localhost:8181/services/internal/metrics/catalogQueries.json?dateOffset=1800>.

The table below describes each of the supported formats, how to access them, and an example where applicable. (NOTE: all example URLs begin with \https://localhost:8993 which is omitted in the table for brevity.)

Table 43. Metrics Formats

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

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

Sample XML-Formatted Output

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

Sample JSON-formatted Output

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

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.

Usage Limitations of the Metrics Endpoint

The Metrics Collecting Application uses a “round robin” database. It uses one that does not store individual values but, instead, stores the rate of change between values at different times. Due to the nature of this method of storage, along with the fact that some processes can cross time frames,

small discrepancies (differences in values of one or two have been experienced) may appear in values for different time frames. These will be especially apparent for reports covering shorter time frames such as 15 minutes or one hour. These are due to the averaging of data over time periods and should not impact the values over longer periods of time.

23.1.8. OpenSearch Endpoint

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

Installing the OpenSearch Endpoint

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

Configuring the OpenSearch Endpoint

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

OpenSearch URL

```
https://localhost:8993/services/catalog/query
```

From Code:

The OpenSearch specification defines a file format to describe an OpenSearch endpoint. This file is XML-based and is used to programmatically 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://localhost: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

Table 44. 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

Table 45. Temporal Extension

OS Element	HTTP Parameter	Possible Values	Comments
start	dtstart	RFC-3399-defined value	yyyy-MM-dd'T' 'HH:mm:ss.SSSZ
end	dtend	RFC-3399-defined value	yyyy-MM-dd'T' 'HH:mm:ss.SSSZ

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

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

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

Geospatial Extension

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

OS Element	HTTP Parameter	Possible Values	Comments
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
geometry	geometry	WKT Geometries: POINT, POLYGON, MULTIPOLYPOINT, MULTIPOLYGON	Examples: POINT(10 20) where 10 is the longitude and 20 is the latitude. POLYGON ((30 10, 10 20, 20 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 46. 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 47. 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 <code>src=local</code> to query the local site. If src is not provided, the default behavior is to execute an enterprise search to the entire federation.

Table 48. 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	Any valid datatype	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.

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

23.2. Endpoint Utility Services

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

23.2.1. Compression Services

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

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

The formats supported in DDF are:

gzip

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

exi

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

Installing a Compression Service

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

To Install:

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

WARNING

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

Configuring Compression Services

None.

Compression Imported Services

None.

Table 49. Compression Exported Services

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

23.3. Developing Endpoints

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 [Working with OSGi - Service Registry](#) for examples.)

Deploy the packaged service to DDF. (Refer to [Working with OSGi - Bundles](#).)

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.

Table 50. 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.

24. Eventing

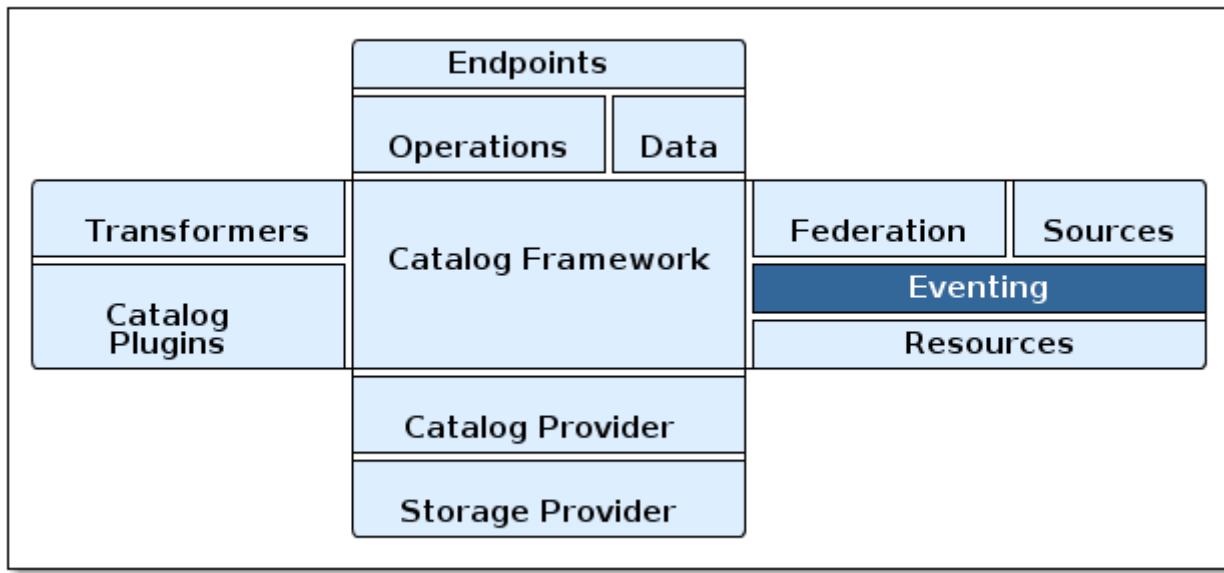


Figure 12. Eventing Architecture

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

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

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

24.1. Eventing Components

The key components of DDF Eventing include:

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

After reading this section, you will be able to:

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

24.1.1. Subscriptions

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

the OGC Filter specification.

Subscription Lifecycle

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

Creation

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

Evaluation

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

Update Evaluation

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

Durability

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

Subscriptions are not persisted by the Catalog itself.

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

IMPORTANT

24.1.2. Creating a Subscription

Currently, the Catalog reference implementation does not contain a subscription endpoint. Nevertheless, an endpoint that exposes a web service interface to create, update, and delete subscriptions would provide a client's subscription's filtering criteria to be used by Catalog's Event Processor to determine which create, update, and delete events are of interest to the client. The endpoint client also provides the callback URL of the event consumer to be called when an event matching the subscription's criteria is found. This callback to the event consumer is made by a Delivery Method implementation that the client provides when the subscription is created.

Whenever an event occurs in the Catalog matching the subscription, the Delivery Method implementation will be called by the Event Processor. The Delivery Method will, in turn, send the event notification out to the event consumer. As part of the subscription creation process, the Catalog verifies that the event consumer at the specified callback URL is available to receive callbacks. Therefore, the client must ensure the event consumer is running prior to creating the subscription. The Catalog completes the subscription creation by executing any pre-subscription Catalog Plugins, and then registering the subscription with the OSGi Service Registry. The Catalog does not persist subscriptions by default.

Delivery Method

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

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

25. Sources

A **source** is a system consisting of a catalog containing Metacards.

Catalog sources are used to connect Catalog components to data sources, local and remote. Sources act as proxies to the actual external data sources, e.g., a RDBMS database or a NoSQL database.

25.1. Types of Sources

Remote Source

Read-only data sources that support query operations but cannot be used to create, update, or delete metacards.

NOTE Remote sources currently extend the `ResourceReader` interface. However, a `RemoteSource` is not treated as a `ResourceReader`. The `getSupportedSchemes()` method should never be called on a `RemoteSource`, thus the suggested implementation for a `RemoteSource` is to return an empty set. The `retrieveResource(...)` and `getOptions(...)` methods will be called and MUST be properly implemented by a `RemoteSource`.

Federated Sources

A federated source is a remote source that can be included in federated queries by request or as part of an enterprise query. Federated sources support query and site information operations only. Catalog modification operations, such as create, update, and delete, are not allowed. Federated sources also expose an event service, which allows the Catalog Framework to subscribe to even notifications when metacards are created, updated, and deleted.

Catalog instances can also be federated to each other. Therefore, a Catalog can also act as a

federated source to another Catalog.

Catalog Providers

A Connected Source is a local or remote source that is always included in every local and enterprise query, but is hidden from being queried individually. A connected source's identifier is removed in all query results by replacing it with DDF's source identifier. The Catalog Framework does not reveal a connected source as a separate source when returning source information responses.

Catalog Stores

A Catalog Provider is used to interact with data providers, such as files systems or databases, to query, create, update, or delete data. The provider also translates between DDF objects and native data formats.

All sources, including federated source and connected source, support queries, but a Catalog provider also allows metacards to be created, updated, and deleted. A Catalog provider typically connects to an external application or a storage system (e.g., a database), acting as a proxy for all catalog operations.

Catalog Stores

A Catalog Store is an editable store that is either local or remote.

25.2. Federation Strategy

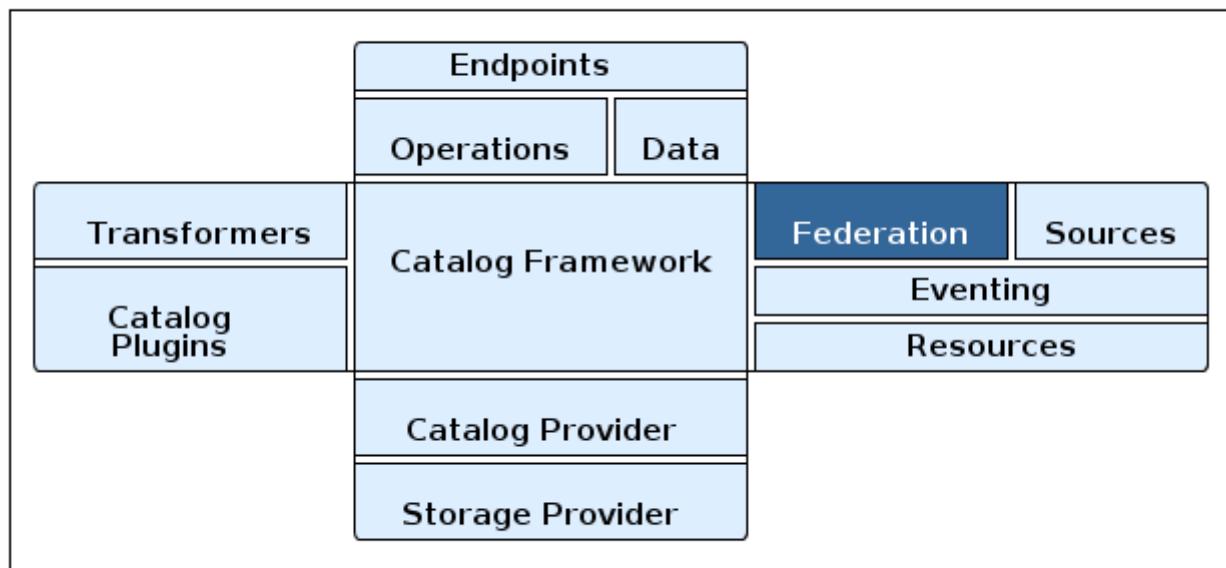


Figure 13. Federation

The Catalog normalizes incoming queries into an OGC Filter format. When the query is disseminated by the Catalog Framework to the sources, each source is responsible for denormalizing the OGC Filter formatted query into the format understood by the external store that the source is acting as a proxy. This normalization/denormalization is what allows any endpoint to interface with any type of source. For example, a query received by the OpenSearch Endpoint can

be executed against an OpenSearch Source.

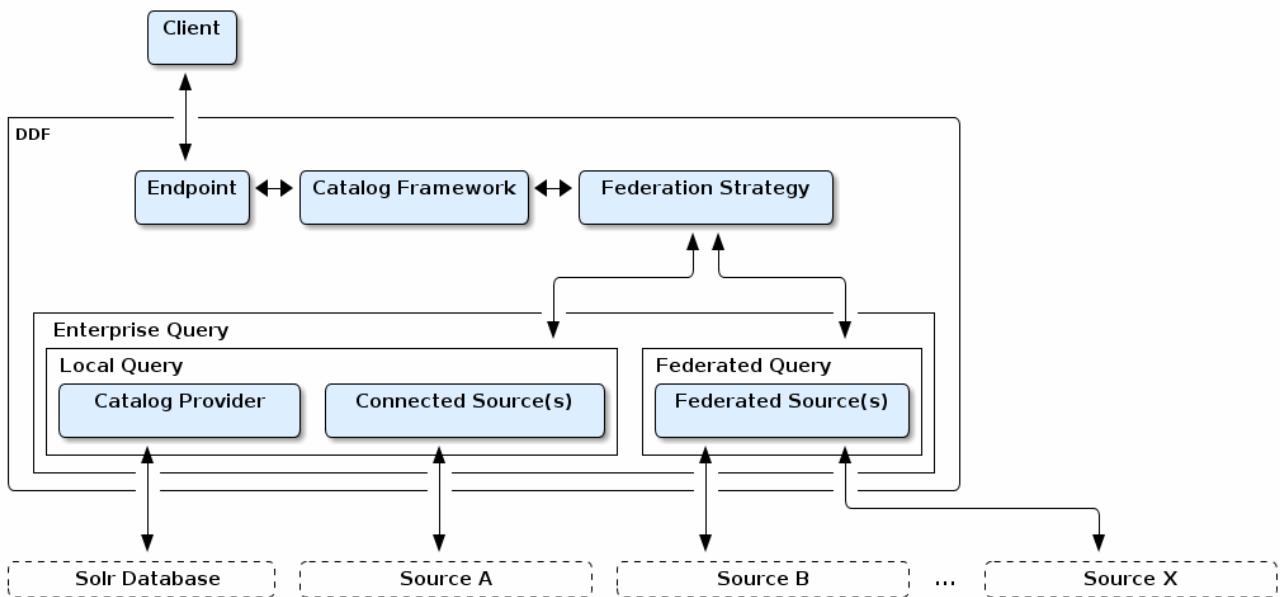


Figure 14. Federation Data Flow

A federation strategy federates a query to all of the Remote Sources in the query's list, processes the results in a unique way, and then returns the results to the client. For example, implementations can choose to halt processing until all results return and then perform a mass sort or return the results back to the client as soon as they are received back from a Federated Source.

An endpoint can optionally specify the federation strategy to use when it invokes the query operation. Otherwise, the Catalog provides a default federation strategy that will be used: the Catalog Federation Strategy.

25.2.1. Catalog Federation Strategy

The Catalog Federation Strategy is the default federation strategy and is based on sorting metacards by the sorting parameter specified in the federated query.

The possible sorting values are:

- metacard's effective date/time
- temporal data in the query result
- distance data in the query result
- relevance of the query result

The supported sorting orders are ascending and descending.

The default sorting value/order automatically used is relevance descending.

WARNING

The Catalog Federation Strategy expects the results returned from the Source to be sorted based on whatever sorting criteria were specified. If a metadata record in the query results contains null values for the sorting criteria elements, the Catalog Federation Strategy expects that result to come at the end of the result list.

Configuring Federation Strategy

The Catalog Federation Strategy configuration can be found in the Admin Console.

1. Navigate to Admin Console.
2. Select **Catalog**
3. Select **Configuration**
4. Select **Catalog Federation Strategy**.

Table 51. Catalog Federation Strategy

Name	Id	Type	Description	Default Value	Required
Maximum start index	maxStartIndex	Integer	Sets a limit on the number of results this sorted federation strategy can handle from each federated source. A large start index in conjunction with several federated sources could yield a large result set, which the sorted federation strategy has a limited ability to do. The admin can make a rough calculation to decide what maximum start index to use based on the amount of memory in the system, the amount of federated sources, the number of threads, and the expected amount of query results requested ((average # of threads) * (maximum # of federated sources) * (maxstartIndex + maximumQueryResults)) must fit into the allocated memory of the running distribution. This field will be removed when sorted federation strategy has the ability to sort a larger amount of results.	50000	true
Expiration Interval	expirationIntervalInMinutes	Long	Interval that Solr Cache checks for expired documents to remove.	10	true
Expiration Age	expirationAgeInMinutes	Long	The number of minutes a document will remain in the cache before it will expire. Default is 7 days.	10080	true

Name	Id	Type	Description	Default Value	Required
Cache Everything	cachingEverything	Boolean	Cache all results unless configured as native	false	true
Cache Remote Ingests	cacheRemoteIngests	Boolean	Cache remote ingest results	false	true
Show Validation Errors	showErrors	Boolean	Show metacards with validation errors in search results	false	true
Show Validation Warnings	showWarnings	Boolean	Show metacards with validation warnings in search results	true	true

Table 52. Catalog Federation Strategy Exported Services

Managed Service PID	ddf.catalog.federation.impl.CachingFederationStrategy
Managed Service Factory PID	N/A

25.3. Federated Sources

The following federated sources are included with a standard installation of DDF:

Atlassian Confluence® Federated Source

Retrieves pages, comments, and attachments from an Atlassian Confluence® REST API.

CSW Federated Source

Supports searching collections of descriptive information (metadata) for data, services, and related information objects.

CSW Federation Profile Source

Supports searching collections of descriptive information (metadata) for data, services, and related information objects.

GMD CSW Source

implements [Application Profile ISO 19115/ISO19119](#). Supports searching collections of descriptive information (metadata) for data, services, and related information objects.

OpenSearch Source

Uses [OpenSearch](#) to query for metadata from Content Discovery and Retrieval (CDR) Search V1.1 compliant sources.

WFS v1.0.0 Source

Allows for requests for geographical features across the web using platform-independent calls.

[WFS v2.0.0 Source](#)

Queries a WFS version 2.0.0 compliant service.

25.3.1. Federated Source for Atlassian Confluence®

The Confluence source provides a Federated Source to retrieve pages, comments, and attachments from an Atlassian Confluence® REST API and turns the results into Metacards the system can use. The Confluence source does provide a Connected Source interface but its functionality has not been verified.

Confluence Source has been tested against the following versions of Confluence with REST API v2

- Confluence 1000.444.5 (Cloud)
- Confluence 5.10.6 (Server)
- Confluence 5.10.7 (Server)

Installing the Confluence Federated Source

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

Add a New CSW Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Add a New source.
5. Name the New source.
6. Select **Confluence Federated Source** from **Binding Configurations**.

Configuring the Confluence Federated Source

Configure an Existing CSW Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Select the name of the source to edit.

Table 53. Confluence Federated Source

Name	Property	Type	Description	Default Value	Required
Source Name	shortname	String			Yes

Name	Property	Type	Description	Default Value	Required
Confluence Rest URL	endpointUrl	String	The Confluence Rest API endpoint URL. Example: <a href="https://<host>:<port>/rest/api/content">https://<host>:<port>/rest/api/content		Yes
Username	username	String	Username to use with HTTP Basic Authentication. This auth info will overwrite any federated auth info. Only set this if the Confluence endpoint requires basic authentication.		No
Password	password	Password	Password to use with HTTP Basic Authentication. This auth info will overwrite any federated auth info. Only set this if the Confluence endpoint requires basic authentication.		No
Include Page Contents In Results	includePageContent	Boolean	Flag indicating if Confluence page contents should be included in the returned results.	false	No
Include Archived Spaces	includeArchivedSpaces	Boolean	Flag indicating if archived confluence spaces should be included in search results.	false	No
Exclude Confluence Spaces	excludeSpaces	Boolean	Flag indicating if the list of Confluence Spaces should be excluded from searches instead of included.	false	No
Confluence Spaces	confluenceSpaces	String cardinality =1000	The confluence spaces to include/exclude from searches. If no spaces are specified, all visible spaces will be searched.		No
Additional Attributes	additionalAttributes	String cardinality =100	Additional attributes to add to confluence metacards returned from this source.		No
Availability Poll Interval	availabilityPollInterval	Long	Availability polling interval in milliseconds.	60000	No

Usage Limitations of the Confluence Federated Source

Most of the fields that can be queried on confluence have some sort of restriction on them. Most of the fields do not support the `like` aka `~` operation so the source will convert `like` queries to `equal` queries for attributes that don't support `like`. If the source receives a query with attributes it doesn't understand, it will just ignore them. If the query doesn't contain any attributes that map to

confluence search attributes, an empty result set will be returned.

Depending on your version of Confluence, when downloading attachments you might get redirected to a different download URL. If this happens the default URLResourceReader configuration does not allow redirects and will fail the download. This can be fixed by enabling redirects in the URLResourceReader configuration.

25.3.2. CSW Federated Source

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

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

Installing the CSW Federated Source

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

Add a New CSW Federated Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **CSW Federated Source** from **Binding Configurations**.

Configuring the CSW Federated Source

Configure an Existing CSW Federated Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

Table 54. CSW Specification Profile Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	null	true

Name	Id	Type	Description	Default Value	Required
CSW URL	<code>cswUrl</code>	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec	<code> \${org.codice.ddf.system.protocol }\${org.codice.ddf.system.hostname}: \${org.codice.ddf.system.port} \${org.codice.ddf.system.rootContext}/csw</code>	true
Event Service Address	<code>eventServiceAddress</code>	String	DDF Event Service endpoint.	<code> \${org.codice.ddf.system.protocol }\${org.codice.ddf.system.hostname}: \${org.codice.ddf.system.port} \${org.codice.ddf.system.rootContext}/csw/subscription</code>	false
Register for Events	<code>registerForEvents</code>	Boolean	Check to register for events from this source.	false	false
Username	<code>username</code>	String	Username for CSW Service (optional)	null	false
Password	<code>password</code>	Password	Password for CSW Service (optional)	null	false
Disable CN Check	<code>disableCnCheck</code>	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	<code>coordinateOrder</code>	String	Coordinate order that remote source expects and returns spatial data in	LON_LAT	true
Use posList in LinearRing	<code>usePosList</code>	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	false

Name	Id	Type	Description	Default Value	Required
Metocard Mappings	metocardMappings	String	Mapping of the Metocard 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	false
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Connection Timeout	connectionTimeout	Integer	Amount of time to attempt to establish a connection before timing out,in milliseconds.	30000	true
Receive Timeout	receiveTimeout	Integer	Amount of time to wait for a response before timing out,in milliseconds.	60000	true
Output Schema	outputSchema	String	Output Schema	http://www.opengis.net/cat/csw/2.0.2	true
Query Type Name	queryTypeName	String	Qualified Name for the Query Type used in the CSW GetRecords request	csw:Record	true
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	true
Force CQL Text as the Query Language	isCqlForce	Boolean	Force CQL Text	false	true
Forced Spatial Filter Type	forceSpatialFilter	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Security Attributes	securityAttributeStrings	String	Security attributes for this source	null	true

Usage Limitations of the CSW Federated Source

- The CSW Federated Source does not support text path searches.
- Nearest neighbor spatial searches are not supported.

25.3.3. CSW Federation Profile Source

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

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

Installing the CSW Federation Profile Source

The CSW Federation Profile Source is installed by default with a standard installation in the Spatial application.

Configure the CSW Federation Profile Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Add a New source.
- Name the New source.
- Select **CSW Specification Profile Federated Source** from **Binding Configurations**.

Configuring the CSW Federation Profile Source

Configure an Existing CSW Federated Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

Table 55. CSW Federation Profile Source

Name	Id	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	CSW	true

Name	Id	Type	Description	Default Value	Required
CSW URL	<code>cswUrl</code>	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec	<code> \${org.codice.ddf.system.protocol }\${org.codice.ddf.system.hostname}:\${org.codice.ddf.system.port}\${org.codice.ddf.system.rootContext}/csw</code>	true
CSW Event Service Address	<code>eventServiceAddress</code>	String	CSW Event Service endpoint.	<code> \${org.codice.ddf.system.protocol }\${org.codice.ddf.system.hostname}:\${org.codice.ddf.system.port}\${org.codice.ddf.system.rootContext}/csw/subscription</code>	false
Register for Events	<code>registerForEvents</code>	Boolean	Check to register for events from this connected source.	false	false
Username	<code>username</code>	String	Username for CSW Service (optional)	null	false
Password	<code>password</code>	String	Password for CSW Service (optional)	null	false

Usage Limitations of the CSW Federation Profile Source

- The CSW Federation Profile Source does not support text path searches.
- Nearest neighbor spatial searches are not supported.

25.3.4. 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](#).

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

Installing the GMD CSW APISO v2.0.2 Source

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

Configure a new GMD CSW APISO v2.0.2 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **GMD CSW ISO Federated Source** from **Binding Configurations**.

Configuring the GMD CSW APISO v2.0.2 Source

Configure an existing GMD CSW APISO v2.0.2 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

Table 56. GMD CSW ISO Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	null	true
CSW URL	cswUrl	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec	null	true
Username	username	String	Username for CSW Service (optional)	null	false
Password	password	Password	Password for CSW Service (optional)	null	false
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	coordinateOrder	String	Coordinate order that remote source expects and returns spatial data in	LON_LAT	true

Name	Id	Type	Description	Default Value	Required
Use posList in LinearRing	usePosList	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	false
Metocard Mappings	metocardMappings	String	Mapping of the Metocard 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	false
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Connection Timeout	connectionTimeout	Integer	Amount of time to attempt to establish a connection before timing out,in milliseconds.	30000	true
Receive Timeout	receiveTimeout	Integer	Amount of time to wait for a response before timing out,in milliseconds.	60000	true
Output Schema	outputSchema	String	Output Schema	http://www.isotc211.org/2005/gmd	true
Query Type Name	queryTypeName	String	Qualified Name for the Query Type used in the CSW GetRecords request	gmd:MD_Metadata	true
Query Type Namespace	queryTypeNamespace	String	Namespace for the Query Type used in the CSW GetRecords request	http://www.isotc211.org/2005/gmd	true

Name	<code>Id</code>	Type	Description	Default Value	Required
Force CQL Text as the Query Language	<code>isCqlForce d</code>	Boolean	Force CQL Text	false	true
Forced Spatial Filter Type	<code>forceSpatialFilter</code>	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Security Attributes	<code>securityAttributeStrings</code>	String	Security attributes for this source	null	true

25.3.5. OpenSearch Source

The OpenSearch source provides a [Federated Source](#) that has the capability to do [OpenSearch](#) queries for metadata from Content Discovery and Retrieval (CDR) Search V1.1 compliant sources. The OpenSearch source does not provide a [Connected Source](#) interface.

Installing an OpenSearch Source

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

Configure a new OpenSearch Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **OpenSearch Source** from **Binding Configurations**.

Configuring an OpenSearch Source

Configure an existing OpenSearch Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

Table 57. Catalog OpenSearch Federated Source

Name	Id	Type	Description	Default Value	Required
Source Name	<code>shortname</code>	String	null	DDF-OS	true
OpenSearch service URL	<code>endpointUrl</code>	String	The OpenSearch endpoint URL or DDF's OpenSearch endpoint (https://localhost:8993/services/catalog/query)	<code> \${org.codice.ddf.system.protocol} \${org.codice.ddf.system.hostname}: \${org.codice.ddf.system.port} \${org.codice.ddf.system.rootContext}/catalog/query</code>	true
Username	<code>username</code>	String	Username to use with HTTP Basic Authentication. This auth info will overwrite any federated auth info. Only set this if the OpenSearch endpoint requires basic authentication.		false
Password	<code>password</code>	Password	Password to use with HTTP Basic Authentication. This auth info will overwrite any federated auth info. Only set this if the OpenSearch endpoint requires basic authentication.		false
OpenSearch query parameters	<code>parameters</code>	String	Query parameters to use with the OpenSearch connection.	<code>q,src,mr,start,count,mt,dn,lat,lon,radius,bbx,polygon,dtstart,dte nd,dateName,filter,sort</code>	true
Always perform local query	<code>localQueryOnly</code>	Boolean	When federating with other DDFs, keep this checked. If checked, this source performs a local query on the remote site (by setting src=local in endpoint URL), as opposed to an enterprise search.	true	true

Name	Id	Type	Description	Default Value	Required
Convert to BBox	<code>shouldConvertToBBox</code>	Boolean	Converts Polygon and Point-Radius searches to a Bounding Box for compatibility with older interfaces. Generated bounding box is a very rough representation of the input geometry.	true	true

Using OpenSearch Source

Use the OpenSearch source if querying a CDR-compliant search service is desired.

OpenSearch Query Format

Table 58. OpenSearch Parameter to DDF Query Mapping

OpenSearch/CDR Parameter	DDF Data Location
<code>q={searchTerms}</code>	Pulled verbatim from DDF query.
<code>src={fs:routeTo?}</code>	Unused
<code>mr={fs:maxResults?}</code>	Pulled verbatim from DDF query.
<code>count={count?}</code>	Pulled verbatim from DDF query.
<code>mt={fs:maxTimeout?}</code>	Pulled verbatim from DDF query.
<code>dn={idn:userDN?}</code>	DDF Subject
<code>lat={geo:lat?}</code>	Pulled verbatim from DDF query.
<code>lon={geo:lon?}</code>	Pulled verbatim from DDF query.
<code>radius={geo:radius?}</code>	Pulled verbatim from DDF query.
<code>bbox={geo:box?}</code>	Converted from Point-Radius DDF query.
<code>polygon={geo:polygon?}</code>	Pulled verbatim from DDF query.
<code>dtstart={time:start?}</code>	Pulled verbatim from DDF query.
<code>dtend={time:end?}</code>	Pulled verbatim from DDF query.
<code>dateName={cat:dateName?}</code>	Unused
<code>filter={fsa:filter?}</code>	Unused
<code>sort={fsa:sort?}</code>	Translated from DDF query. Format: "relevance" or "date" Supports "asc" and "desc" using colon as delimiter.

Table 59. OpenSearch Source Exported Services

Registered Interface	Service Property	Value
<code>DDF.catalog.source.FederatedSource</code>		

Table 60. OpenSearch Source Imported Services

Registered Interface	Availability	Multiple	Filter
ddf.catalog.transform.InputTransformer	required	false	(&(mime-type=text/xml)(id=xml))

Usage Limitations of the OpenSearch Source

The OpenSearch source does not provide a [Connected Source](#) interface.

25.3.6. WFS v1.0.0 Source

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

A Web Feature Service (WFS) source is an implementation of the [FederatedSource](#) interface provided by the DDF Framework.

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

Installing the WFS v1.0.0 Source

The WFS v1.0.0 Source is installed by default with a standard installation in the Spatial application.

Configure a new WFS v1.0.0 Source through the Admin Console:

- Navigate to the [Admin Console](#).
- Select the [Catalog](#) application.
- Select the [Sources](#) tab.
- Add a New source.
- Name the New source.
- Select **WFS v1.0.0 Source** from **Binding Configurations**.

Configuring the WFS v1.0.0 Source

Configure an existing WFS v1.0.0 Source through the Admin Console:

- Navigate to the [Admin Console](#).
- Select the [Catalog](#) application.
- Select the [Sources](#) tab.
- Select the name of the source to edit.

Table 61. WFS v1.0.0 Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	WFS_v1_0_0	true
WFS URL	wfsUrl	String	URL to the endpoint implementing the Web Feature Service (WFS) spec	null	true
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Username	username	String	Username for WFS Service (optional)	null	false
Password	password	Password	Password for WFS Service (optional)	null	false
Forced Feature Type	forcedFeatureType	String	Force only a specific FeatureType to be queried instead of all featureTypes	null	false
Non Queryable Properties	nonQueryableProperties	String	Properties listed here will NOT be queryable and any attempt to filter on these properties will result in an exception.	null	false
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	forceSpatialFilter	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Connection Timeout	connectionTimeout	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	receiveTimeout	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

Table 62. WFS v1.0.0 Connected Source

Name	Id	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	WFS	true
WFS URL	wfsUrl	String	URL to the endpoint implementing the Web Feature Service (WFS) spec	null	true
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true

Name	Id	Type	Description	Default Value	Required
Username	<code>username</code>	String	Username for WFS Service (optional)	null	false
Password	<code>password</code>	Password	Password for WFS Service (optional)	null	false
Non Queryable Properties	<code>nonQueryableProperties</code>	String	Properties listed here will NOT be queryable and any attempt to filter on these properties will result in an exception.	null	false
Poll Interval	<code>pollInterval</code>	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	<code>forceSpatialFilter</code>	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Connection Timeout	<code>connectionTimeout</code>	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	<code>receiveTimeout</code>	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

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=GetCapabil
ities
```

In this case, the WFS URL would be: <http://www.example.org:8080/geoserver/ows>

25.3.7. WFS v2.0.0 Source

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

Use the WFS Source if querying a WFS version 2.0.0 compliant service. Also see [Working with WFS Sources](#).

Installing the WFS v2.0.0 Source

The WFS v2.0.0 Source is installed by default with a standard installation in the Spatial application.

Configure a new WFS v2.0.0 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **WFS v2.0.0 Source** from **Binding Configurations**.

Configuring the WFS v2.0.0 Source

Configure an existing WFS v2.0.0 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

Table 63. WFS 2.0.0 Federated Source

Name	ID	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	WFS_v2_0_0	true
WFS URL	wfsUrl	String	URL to the endpoint implementing the Web Feature Service (WFS) 2.0.0 spec	null	true
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	coordinateOrder	String	Coordinate order that remote source expects and returns spatial data in	LAT_LON	true
Forced Feature Type	forcedFeatureType	String	Force only a specific FeatureType to be queried instead of all featureTypes	null	false

Name	Id	Type	Description	Default Value	Required
Disable Sorting	<code>disableSorting</code>	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	true
Username	<code>username</code>	String	Username for the WFS Service (optional)	null	false
Password	<code>password</code>	Password	Password for the WFS Service (optional)	null	false
Non Queryable Properties	<code>nonQueryableProperties</code>	String	Properties listed here will NOT be queryable and any attempt to filter on these properties will result in an exception.	null	false
Poll Interval	<code>pollInterval</code>	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	<code>forceSpatialFilter</code>	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Connection Timeout	<code>connectionTimeout</code>	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	<code>receiveTimeout</code>	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

Table 64. WFS 2.0.0 Connected Source

Name	Id	Type	Description	Default Value	Required
Source ID	<code>id</code>	String	The unique name of the Source	WFS	true
WFS URL	<code>wfsUrl</code>	String	URL to the endpoint implementing the Web Feature Service (WFS) 2.0.0 spec	null	true
Disable CN Check	<code>disableCnCheck</code>	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true

Name	Id	Type	Description	Default Value	Required
Force Longitude/Latitude coordinate order	<code>isLonLatOrder</code>	Boolean	Force Longitude/Latitude coordinate order	false	true
Disable Sorting	<code>disableSorting</code>	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	true
Username	<code>username</code>	String	Username for the WFS Service (optional)	null	false
Password	<code>password</code>	Password	Password for the WFS Service (optional)	null	false
Non Queryable Properties	<code>nonQueryableProperties</code>	String	Properties listed here will NOT be queryable and any attempt to filter on these properties will result in an exception.	null	false
Poll Interval	<code>pollInterval</code>	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	<code>forceSpatialFilter</code>	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Connection Timeout	<code>connectionTimeout</code>	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	<code>receiveTimeout</code>	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

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=GetCapabil  
ities
```

In this case, the WFS URL would be

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

Mapping WFS Feature Properties to Metocard 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 metocard. The [MetocardMapper](#) was added to allow an administrator to configure which feature properties map to which metocard attributes.

Using the WFS [MetocardMapper](#)

Use the WFS [MetocardMapper](#) to configure which feature properties map to which metocard 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 metocard 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 metocard attributes can be achieved through the [MetocardMapper](#). The [MetocardMapper](#) is set by creating a feature file configuration which specifies the appropriate mapping. The mappings are specific to a given feature type.

Installing the WFS [MetocardMapper](#)

The WFS [MetocardMapper](#) is not installed by default with a standard application in the Spatial application.

Configuring the WFS [MetocardMapper](#)

Table 65. WFS MetocardMapper Configurable Properties

Title	Property	Type	Description	Default Value	Required
Feature Type	featureType	String	Feature Type. Format is <code>{URI}local-name</code>		Yes
Metocard Attribute to WFS Feature Property Mapping	metocardAttrToFeaturePropMap	String	Metocard Attribute to WFS Feature Property Mapping. Format is <code>metocardAttribute=featureProperty</code>		Yes

Title	Property	Type	Description	Default Value	Required
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 **MetacardMapper**, 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 metocard attribute 'title'. In this particular case, since we mapped the state name to title in the metocard, 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="2.10.3"
    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>
```

25.3.8. WFS Services

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.

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](#) or [WFS v2.0.0 Source](#) for more information about how to configure a WFS source.

Converting 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 metocard. The OGC package contains a `GenericFeatureConverter` that attempts to populate mandatory metocard fields with properties from the WFS feature XML. All properties will be mapped directly to new attributes in the metocard. However, the `GenericFeatureConverter` may not be able to populate the default metocard fields with properties from the feature XML.

Creating a Custom Converter

To more accurately map WFS feature properties to fields in the metocard, 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.html> 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 Metocard unmarshal(HierarchicalStreamReader reader, UnmarshallingContext ctx) {
    MetocardImpl 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 Metocard.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>

```

25.4. Connected Sources

The following connected sources are included with a standard installation of DDF:

CSW Connected Source

Supports searching collections of descriptive information (metadata) for data, services, and

related information objects. *Implements CSW*.

[WFS Source v1.0.0](#)

Allows for requests for geographical features across the web using platform-independent calls.
Implements WFS v.1.0.0.

[WFS Source v2.0.0](#)

Enables querying a WFS version 2.0.0 compliant service. *Implements WFS v.2.0.0.*

25.4.1. CSW Connected Source

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

Installing the CSW Connected Source

The CSW Connected Source is not installed by default with a standard installation in the Spatial application.

Add a CSW Connected Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Add a New source.
- Name the New source.
- Select **CSW Connected Source** from **Binding Configurations**.

Configuring the CSW Connected Source

Configure an Existing CSW Federated Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the name of the source to edit.

Table 66. CSW Connected Source

Name	Id	Type	Description	Default Value	Required
Source ID	<code>id</code>	String	The unique name of the Source.	CSW	true
CSW URL	<code>csgetUrl</code>	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec.	null	true
Event Service Address	<code>eventServiceAddress</code>	String	DDF Event Service endpoint. Do NOT include .wsdl or ?wsdl.	null	false

Name	Id	Type	Description	Default Value	Required
Register for Events	<code>registerForEvents</code>	Boolean	Check to register for events from this connected source.	false	false
Username	<code>username</code>	String	Username for CSW Service (optional).	null	false
Password	<code>password</code>	String	Password for CSW Service (optional).	null	false
Disable CN Check	<code>disableCnCheck</code>	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Force Longitude/Latitude coordinate order	<code>isLonLatOrder</code>	Boolean	Force Longitude/Latitude coordinate order.	false	true
Use posList in LinearRing	<code>usePosList</code>	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing.	false	false
Metocard Mappings	<code>metocardMappings</code>	String	Mapping of the Metocard Attribute names to their CSW property names. The format should be ' <code>title=dc:title</code> '.	effective=c reated, created=da teSubmitte d, modified=mo dified, thumbna il=referenc es, content- type=type, id=identifi er, resource- uri=source	false
Poll Interval	<code>pollInterval</code>	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Connection Timeout	<code>connectionTimeout</code>	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	<code>receiveTimeout</code>	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

Name	<code>Id</code>	Type	Description	Default Value	Required
Output Schema	<code>outputSchema</code>	String	Output Schema	http://www.opengis.net/cat/csw/2.0.2	true
Query Type Name	<code>queryTypeName</code>	String	Qualified Name for the Query Type used in the CSW GetRecords request.	<code>csw:Record</code>	true
Query Type Namespace	<code>queryTypeNamespace</code>	String	Namespace prefix for the Query Type used in the CSW GetRecords request.	http://www.opengis.net/cat/csw/2.0.2	true
Force CQL Text as the Query Language	<code>isCqlForce</code>	Boolean	Force CQL Text.	false	true
Forced Spatial Filter Type	<code>forceSpatialFilter</code>	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	<code>NO_FILTER</code>	false

Using the CSW Connected Source

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

Usage Limitations of the CSW Connected Source

- The CSW Connected Source does not support text path searches.
- Nearest neighbor spatial searches are not supported.

25.5. Catalog Stores

The following catalog stores are included with a standard installation of DDF:

Registry Store

Provides a means of configuring connections to other registries and controlling update behavior.

25.5.1. Registry Store

NOTE

The Registry Store is currently marked **Experimental**. While functional and tested, it is subject to change or removal during the incubation period.

The Registry Store is the interface that allows CSW messages to be turned into usable Registry metacards and for those metacards to be turned back into CSW messages.

Installing Registry Store

The Registry Store is installed by default with the Registry application.

Configuring Registry Store

To configure the Registry store:

1. Navigate to the Admin Console.
2. Select **Registry**.
3. Select the **Remote Registries** Tab and click the **Add** button.
 - a. ALTERNATIVELY: Select the **Configuration** Tab and select **Registry Store**.

Table 67. CSW Registry Store

Name	Id	Type	Description	Default Value	Required
Registry ID	<code>id</code>	String	The unique name of the store	null	true
Registry Service URL	<code>cswUrl</code>	String	URL to the endpoint implementing CSW spec capable of returning ebrim formatted records	null	true
Username	<code>username</code>	String	Username for CSW Service (optional)	null	false
Password	<code>password</code>	Password	Password for CSW Service (optional)	null	false
Allow Push	<code>pushAllowed</code>	Boolean	Enable push (write) to this registry	true	true
Allow Pull	<code>pullAllowed</code>	Boolean	Enable pull (read) from this registry	true	true
Push Identity Node	<code>autoPush</code>	Boolean	Enable an automatic publish from the local identity node to this registry. Setting this to Off will have the effect of unpublishing the identity from this registry.	true	true

25.6. Catalog Providers

Catalog Providers are used to interface with the data of DDF.

The Solr Catalog Provider

An implementation of Catalog Provider using [Apache Solr](#) as a data store.

25.6.1. Solr Catalog Provider

The Solr Catalog Provider is included with a standard installation of DDF. The following are the three ways to configure the Solr Catalog Provider:

HTTP Solr (default)

Used with an external (HTTP) Solr server.

Embedded Solr

Used with an embedded Solr server with no HTTP interface.

Solr Cloud

Used with a cluster of Solr servers.

If not using Solr Cloud, whenever the DDF Solr ingests data, it creates an index for it in the data directory. This data indexing enables the data to be searchable by the DDF Solr. The data in the data directory can then be persisted by performing a **commit** on the DDF Solr. When the DDF Solr is restarted, it reloads the data to get back the index.

HTTP Solr

The HTTP Solr is an external Apache Solr server instance. There are two possible installations for the HTTP Solr configuration:

Internal (default)

Points to an Apache Solr server that runs in the same container as DDF.

External

Points to an Apache Solr server in a different container or on a different system.

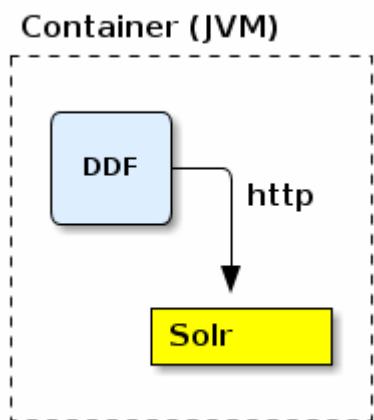


Figure 15. HTTP Solr Internal Deployment

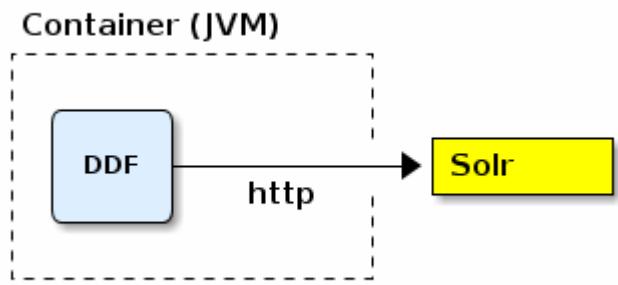


Figure 16. HTTP Solr External Deployment

When to Use HTTP Solr

The HTTP Solr configuration is good for most use cases. See below for details.

HTTP Solr Tradeoffs

- Solr Admin UI can be used.
- Backup can be performed manually or using the Backup command.
- External installation allows isolation of data storage.
- Requires additional protection to be secured.
- Not scalable.
- External installation has network overhead

Installing HTTP Solr

By default, the Solr Catalog application installs a Solr server inside the same container as DDF. To install an External Solr server, refer to [Apache Solr Reference Guide](#).

Configuring HTTP Solr

1. Edit the `<INSTALLATION_DIR>/etc/system.properties`

- a. Comment out the Solr Client Configuration for **Cloud Solr Client** and **Embedded Solr Client** sections.
- b. Uncomment the section for the **Http Solr Client**.

NOTE To point to an External Solr server instead of the default Internal one, put the url in `solr.http.url` (see below for example). Also, two jar files need to be added to the Solr installation as described in [Installing Solr Cloud](#).

`system.properties`

```

solr.client = HttpSolrClient
solr.http.url = https://solr-host-name:8993/solr
solr.data.dir = ${karaf.home}/data/solr

```

Embedded Solr

The Embedded Solr is an embedded, local Apache Solr server instance. It is a local instance that is a lightweight Catalog Provider solution. However, it does not provide a Solr Admin UI or a REST-like HTTP/XML and JSON API. If that is necessary, see [HTTP Solr](#) or [Solr Cloud](#).

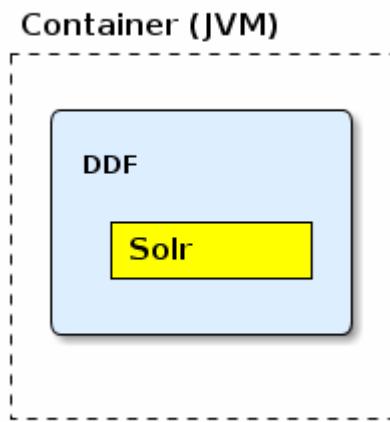


Figure 17. Embedded Solr Deployment

When to Use the Embedded Solr

The Embedded Solr configuration is great for demonstrations, training exercises, or for sparse querying and ingesting. See below for details.

Embedded Solr Tradeoffs

- Requires no installation and very little configuration.
- No HTTP connection required. As a result, there is no network overhead.
- Not scalable.
- There is no Solr Admin UI.
- Backup must be performed manually.

Installing Embedded Solr

No installation required. Follow the Configuring section below and restart the DDF.

Configuring Embedded Solr

1. Edit the `<INSTALLATION_DIR>/etc/system.properties`
 - a. Comment out the Solr Client Configuration for **Cloud Solr Client** and **Http Solr Client** sections.
 - b. Uncomment the section for the **Embedded Solr Client**:

system.properties

```
solr.client = EmbeddedSolrServer  
solr.data.dir = ${karaf.home}/data/solr
```

Solr Cloud

Solr Cloud is a cluster of Solr Server instances that are fault tolerant and highly available. Each Solr Server instance in Solr Cloud provides a Solr Admin UI.

Configuration shared between Solr Server instances is managed by Zookeeper. Zookeeper helps manage the overall structure.

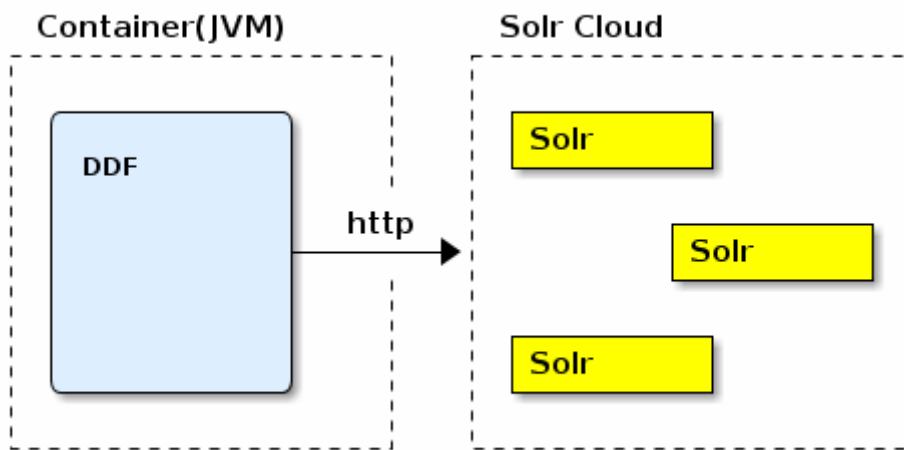


Figure 18. Solr Cloud Deployment

NOTE

Solr Cloud is currently in Beta version. Bugs and feature changes may affect the final release.

When to Use Solr Cloud

If the DDF needs to be accessible for a higher than normal period with low downtime, then Solr Cloud should be used. See below for details.

Solr Cloud Tradeoffs

- Scalable. Can exceed ~2 billion indexed documents.
- Solr Admin UI can be used.
- Has network overhead and requires additional protection to be secure.
- Installation is more involved (requires Zookeeper)
- Configuration and administration is more complex due to replicate, sharding, etc.
- No way to backup currently, but will automatically recover from system failure.

NOTE

The instructions on setting up Solr Cloud for DDF only include setup in a *NIX environment.

Solr Cloud Prerequisites

Before Solr Cloud can be installed:

- ZooKeeper 3.4.5 (Refer to https://zookeeper.apache.org/doc/r3.1.2/zookeeperStarted.html#sc_Download for installation instructions.)
 - *NIX environment
 - JDK 8 or greater

NOTE

A minimum of three Zookeeper nodes required. Three Zookeeper nodes are needed to form a quorum. A three Zookeeper ensemble allows for a single server to fail and the service will still be available. More Zookeeper nodes can be added to achieve greater fault tolerance. The total number of nodes must always be an odd number.

Installing Solr Cloud

Repeat the following procedure for each Solr server instance that will be part of the Solr Cloud cluster:

1. Refer to <https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide> for installation instructions.
 2. Download jar files. The jars are needed to support geospatial and xpath queries and need to be installed on every Solr server instance after the Solr Cloud installation instructions have been followed.
 - a. http://artifacts.codice.org/service/local/repositories/releases/content/org/codice/thirdparty/jts/1.12_1/jts-1.12_1.jar
 - b. <http://artifacts.codice.org/service/local/artifact/maven/content?r=public&g=ddf.platform.solr&a=solr-xpath&v=2.10.3>
 3. Copy downloaded jar files to: <SOLR INSTALL DIR>/server/solr-webapp/webapp/WEB-INF/lib/

NOTE

A minimum of two Solr server instances is required. Each Solr server instance must have a minimum of two shards. Having two Solr server instances guarantees that at least one Solr server is available if one fails. The two shards enables the document mapping to be restored if one shard becomes unavailable.

Configuring Solr Cloud

1. On the DDF server, edit `<INSTALLATION_DIRECTORY>/etc/system.properties`:
 - a. Comment out the Solr Client Configuration for **Http Solr Client** and **Embedded Solr Client** sections.
 - b. Uncomment the section for the **Cloud Solr Client**:

- c. Set `solr.cloud.zookeeper` to `<ZOOKEEPER_1_HOSTNAME>:<PORT_NUMBER>, <ZOOKEEPER_2_HOSTNAME>:<PORT_NUMBER>, <ZOOKEEPER_n_HOSTNAME>:<PORT_NUMBER>`
- d. Set `solr.data.dir` to the desired data directory.

system.properties

```
solr.client = CloudSolrClient
solr.data.dir = ${karaf.home}/data/solr
solr.cloud.zookeeper = zk1:2181,zk2:2181,zk3:2181
```

25.7. Storage Providers

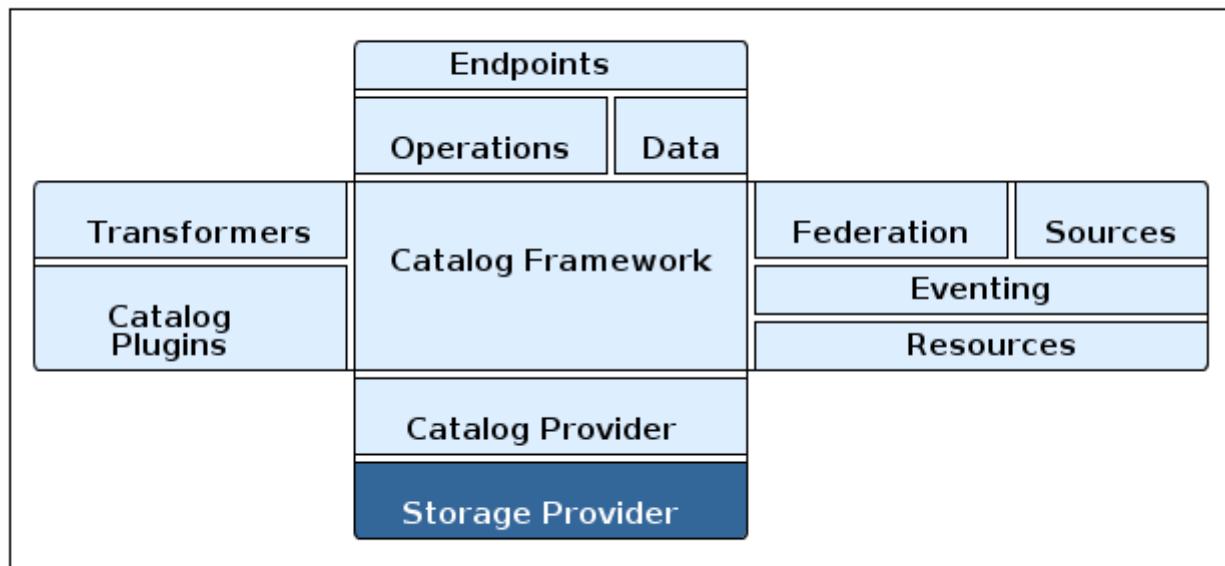


Figure 19. Storage Providers

26. Security Services

26.1. Encryption Service

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

Encryption Command

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

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

security:encrypt Command Example

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

security:encrypt Command Output

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

26.2. Expansion Service

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

Configuring Expansion Service

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

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

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

Expansion Service Instances and Configuration

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

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

Each instance of the expansion service can be configured using a configuration file. The configuration file can have three different types of lines: comments:: any line prefixed with the # character is ignored as a comment (for readability, blank lines are also ignored) attribute separator:: a line starting with `separator=` defines the attribute separator string. rule:: all other lines are assumed to be rules defined in a string format <key>:<original value>:<new value>

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

```

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

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

```

Table 68. Expansion Commands

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

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

Expansion Command Examples and Explanation

security:expansions

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

```

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

```

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

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

security:expand

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

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

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

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

26.3. Security IDP

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

Table 69. Security IdP Components

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

Limitations

NOTE 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](#).

26.4. Security STS

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

Table 70. Security STS 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 and exposes it as a service for the STS.
security-sts-x509validator	security-sts-server	Contains the default CXF x509 validator and exposes it as a service for the STS.

26.4.1. Security STS Client Config

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

Installing the Security STS Client Config

This bundle is installed by default.

Configuring the Security STS Client Config

Configure the Security STS Client Config from the Admin Console:

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

Table 71. Security STS Client

Name	Id	Type	Description	Default Value	Required
SAML Assertion Type:	assertionType	String	The version of SAML to use. Most services require SAML v2.0. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0	true
SAML Key Type:	keyType	String	The key type to use with SAML. Most services require Bearer. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/wss-sx/ws-trust/200512/Bearer	true
SAML Key Size:	keySize	String	The key size to use with SAML. The default key size is 256 and this is fine for most applications. Changing this value from the default could cause services to stop responding.	256	true
Use Key:	useKey	Boolean	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.	true	true
STS WSDL Address:	address	String	STS WSDL Address	<code> \${org.codice.ddf.system.protocol }\${org.codice.ddf.system.hostname}: \${org.codice.ddf.system.port }\${org.codice.ddf.system.rootContext }/SecurityTokenService?wsdl</code>	true

Name	Id	Type	Description	Default Value	Required
STS Endpoint Name:	<code>endpointName</code>	String	STS Endpoint Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}STS_Port	false
STS Service Name:	<code>serviceName</code>	String	STS Service Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}SecurityTokenService	false
Signature Properties:	<code>signatureProperties</code>	String	Path to Signature crypto properties. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/signature.properties	true
Encryption Properties:	<code>encryptionProperties</code>	String	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.	etc/ws-security/server/encryption.properties	true
STS Properties:	<code>tokenProperties</code>	String	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.	etc/ws-security/server/signature.properties	true

Name	Id	Type	Description	Default Value	Required
Claims:	<code>claims</code>	String	List of claims that should be requested by the STS Client.	<code>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role</code>	true

Table 72. Security STS WSS

Name	Id	Type	Description	Default Value	Required
SAML Assertion Type:	<code>assertionType</code>	String	The version of SAML to use. Most services require SAML v2.0. Changing this value from the default could cause services to stop responding.	<code>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</code>	true

Name	Id	Type	Description	Default Value	Required
SAML Key Type:	<code>keyType</code>	String	The key type to use with SAML. Most services require Bearer. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/wss-xs/ws-trust/200512/Bearer	true
SAML Key Size:	<code>keySize</code>	String	The key size to use with SAML. The default key size is 256 and this is fine for most applications. Changing this value from the default could cause services to stop responding.	256	true
Use Key:	<code>useKey</code>	Boolean	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.	true	true
STS WSDL Address:	<code>address</code>	String	STS WSDL Address	<code> \${org.codice.ddf.system.protocol} \${org.codice.ddf.system.hostname}: \${org.codice.ddf.system.httpsPort} \${org.codice.ddf.system.rootContext} /SecurityTokenService?wsdl</code>	true
STS Endpoint Name:	<code>endpointName</code>	String	STS Endpoint Name.	<code>{http://docs.oasis-open.org/wss-xs/ws-trust/200512}STS_Port</code>	false
STS Service Name:	<code>serviceName</code>	String	STS Service Name.	<code>{http://docs.oasis-open.org/wss-xs/ws-trust/200512}SecurityTokenService</code>	false

Name	Id	Type	Description	Default Value	Required
Signature Properties:	<code>signatureProperties</code>	String	Path to Signature crypto properties. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	<code>etc/ws-security/server/signature.properties</code>	true
Encryption Properties:	<code>encryptionProperties</code>	String	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.	<code>etc/ws-security/server/encryption.properties</code>	true
STS Properties:	<code>tokenProperties</code>	String	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.	<code>etc/ws-security/server/signature.properties</code>	true
Claims:	<code>claims</code>	String	Comma-delimited list of claims that should be requested by the STS.	<code>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role</code>	true

Table 73. Security STS Address Provider

Name	Id	Type	Description	Default Value	Required
Use WSS STS	<code>useWss</code>	Boolean	If you have a WSS STS configured, you may prefer to use it for services that need the STS address, such as SOAP sources.	false	true

Table 74. Security STS Client Config Imported Services

Registered Interface	Availability	Multiple
<code>org.osgi.service.cm.ConfigurationAdmin</code>	required	false

Security STS Client Config Exported Services

None.

26.4.2. 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).

26.4.3. Security STS LDAP Login

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

Installing the Security STS LDAP Claims Handler

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

Configuring the Security STS LDAP Claims Handler

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

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

Table 75. Security STS LDAP Claims Handler Settings

Configuration Name	Default Value	Additional Information
LDAP URL	<code>ldaps://\${org.codice.ddf.system.hostname}:1636</code>	
StartTLS	<code>false</code>	Ignored if the URL uses ldaps.
LDAP Bind User DN	<code>cn=admin</code>	This user should have the ability to verify passwords and read attributes for any user.
LDAP Bind User Password	<code>secret</code>	This password value is encrypted by default using the Security Encryption application.
LDAP Username Attribute	<code>uid</code>	
LDAP Base User DN	<code>ou=users,dc=example,dc=com</code>	
LDAP Base Group DN	<code>ou=groups,dc=example,dc=com</code>	

Security STS LDAP Claims Handler Imported Services

None.

Security STS LDAP Claims Handler Exported Services

None.

26.4.4. Security STS LDAP Claims Handler

The Security STS LDAP Claims Handler bundle adds functionality to the STS server that allows it to retrieve claims from an LDAP server. It also adds mappings for the LDAP attributes to the STS SAML claims.

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

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 the Security STS LDAP Claims Handler

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

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

4. Select Security STS LDAP and Roles Claims Handler.

Table 76. Security STS LDAP Claims Handler Settings

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

Implementation Details

Imported Services

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

Exported Services

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

26.4.5. Security STS Server

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

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

Installing the Security STS Server

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

Configuring the Security STS Server

Configure the Security STS Server from the Admin Console:

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

Table 77. Security STS Server Settings

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.

Table 78. Security STS Server Imported Services

Registered Interface	Availability	Multiple
<code>org.apache.cxf.sts.claims.ClaimsHandler</code>	optional	true
<code>org.apache.cxf.sts.token.validator.TokenValidator</code>	optional	true

Security STS Server Exported Services

None.

26.4.6. Security STS Service

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

Installing the Security STS Realm

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

Configuring the Security STS Realm

The Security STS Realm has no configurable properties.

Table 79. Security STS Realm Imported Services

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

Table 80. Security STS Realm Exported Services

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

Developing

Developers will build or extend the functionality of the applications.

27. System Data Flow

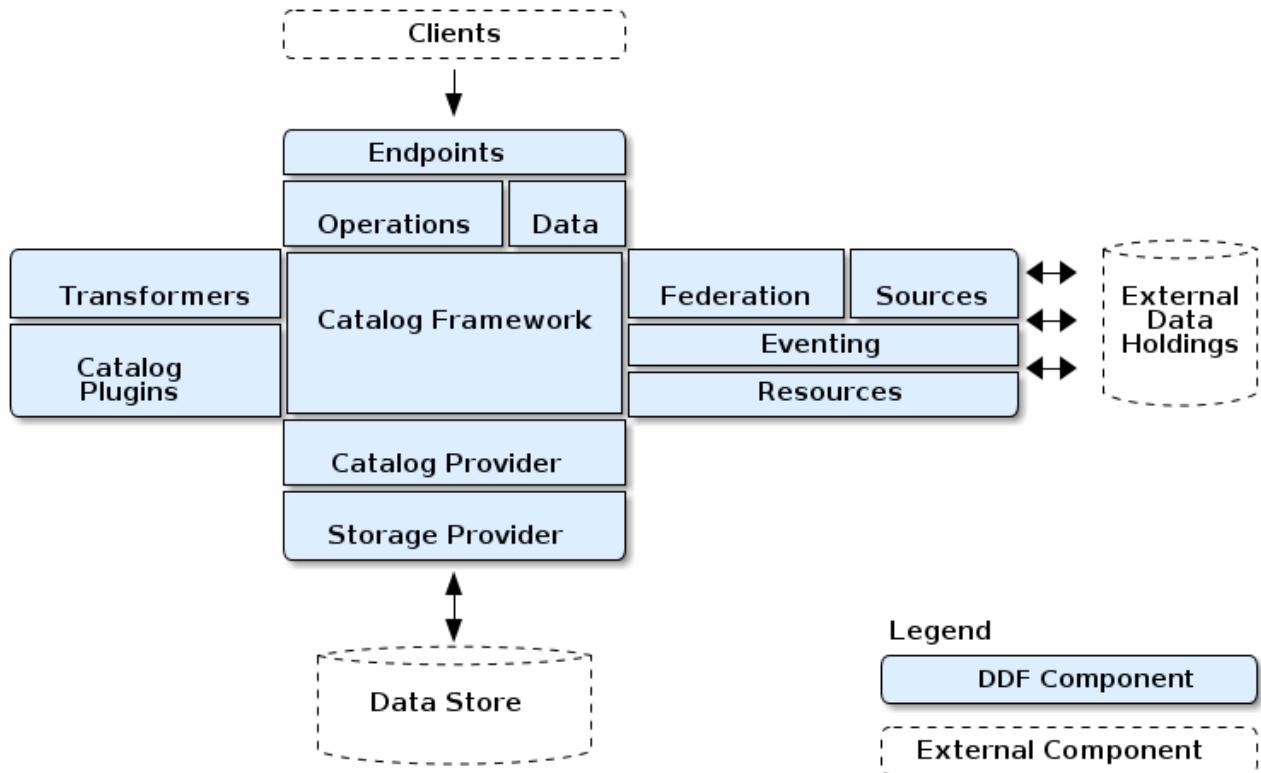


Figure 20. Catalog Architecture

28. Catalog Framework API

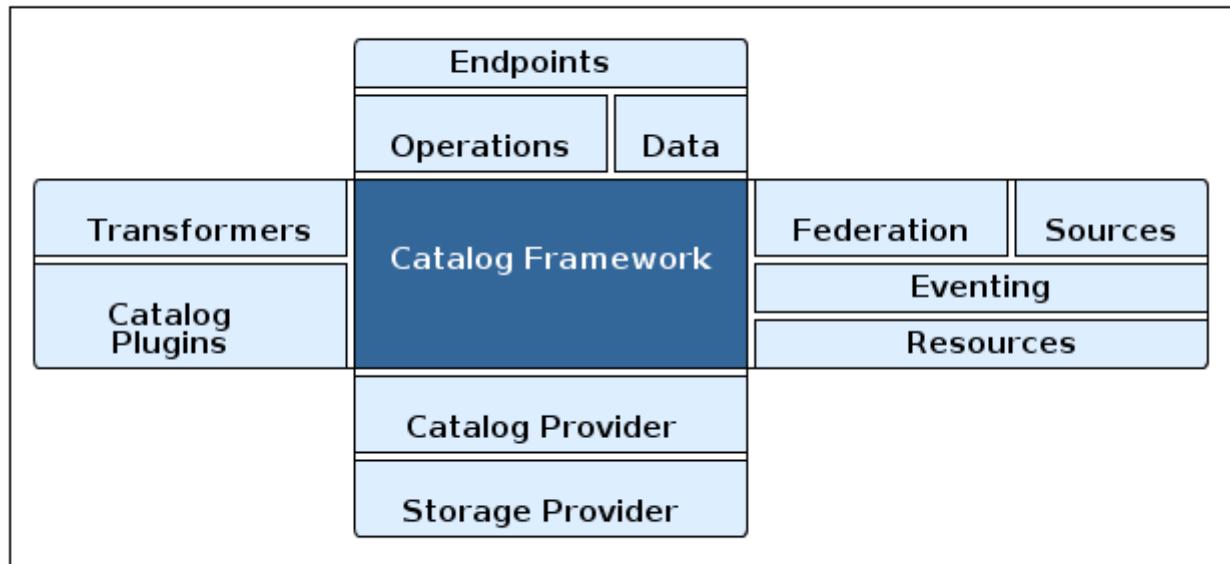


Figure 21. Catalog Framework Architecture

The **CatalogFramework** is the routing mechanism between catalog components that provides integration points for the Catalog Plugins. An **endpoint** invokes the active Catalog Framework,

which calls any configured [Pre-query](#) or [Pre-ingest plug-ins](#). The selected [federation strategy](#) calls the active [Catalog Provider](#) and any connected or federated sources. Then, any Post-query or Post-ingest plug-ins are invoked. Finally, the appropriate response is returned to the calling endpoint.

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](#).

The Catalog Framework decouples clients from service implementations and provides integration points for Catalog Plugins and convenience methods for Endpoint developers.

28.1. Catalog API 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.

28.1.1. 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.

28.1.2. 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.

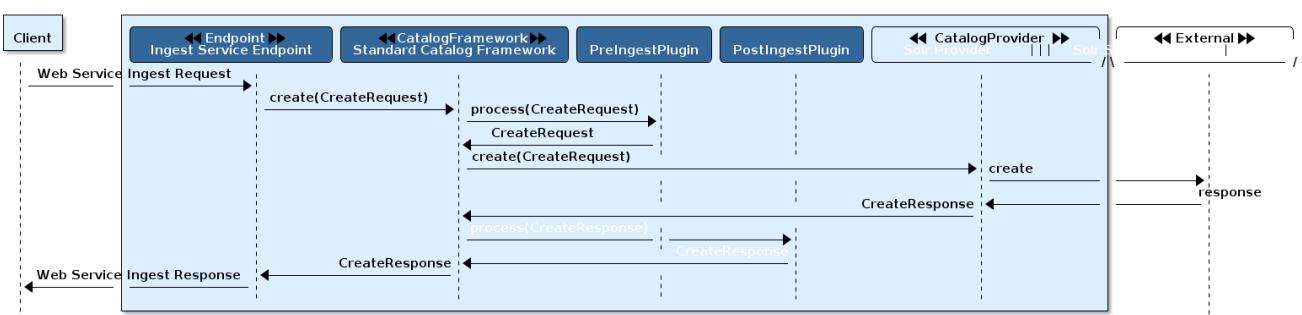


Figure 22. Ingest Request

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).

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

TIP

```
log:set
ERROR ingestLogger
```

Query

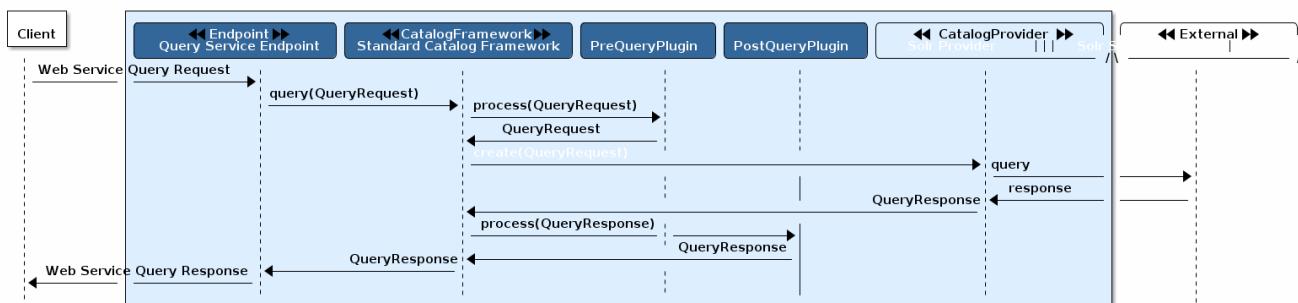


Figure 23. Query Request

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.

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 `<DDF_HOME>/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.

28.1.3. Catalog API

The Catalog API is an OSGi bundle ([catalog-core-api](#)) that contains the Java interfaces for the Catalog components and implementation classes for the Catalog Framework, Operations, and Data components.

Catalog API Search Interfaces

The Catalog API includes two different search interfaces.

Search UI Application Search Interface

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

SSH Search Interface

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

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.

Search Programmatic Flow

Searching the catalog involves three basic steps:

1. Define the search criteria (contextual, spatial, or temporal).
 - 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 Catalog 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 81. 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
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.

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 82. 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	Catalog
Canceling Product Downloads	The ability to cancel a download in progress	Catalog

Capability	Description	Endpoint Integration
Activities	Activities * <code>download</code> * <code>retry</code> * <code>cancel</code> * <code>pause</code> * <code>remove</code> * <code>resume</code>	Catalog, CometD endpoint
Notifications	Time-stamped messages of an action	Catalog, Search UI/CometD endpoint
Workspaces	Ability to save and manage queries and save metacards	Platform, Search UI/CometD endpoint
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 Metocard. Resources may also be known as products or data. Resources may exist either locally or on a remote data store.

Examples of Resources

- NITF image
- MPEG video
- Live video stream
- Audio recording
- Document

Product Retrieval Services

- 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.

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 enabled by default.

NOTE

To configure product caching:

1. Navigate to the **Admin Console**.
2. Select Catalog.
3. Select **Configuration**.
4. Select **Resource Download Settings**.

Table 83. Resource Download Settings

Name	Id	Type	Description	Default Value	Required
Product Cache Directory	productCacheDirectory	String	Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist	true	false
Enable Product Caching	cacheEnabled	Boolean	Product Download feature will attempt to create those directories. Out of the box (without configuration), the product cache directory is <DDF_HOME>/data/product-cache. If a relative path is provided it will be relative to the DDF_HOME. It is recommended to enter an absolute directory path such as /opt/product-cache in Linux or C:\product-cache in Windows.	10	false
Delay (in seconds) between product retrieval retry attempts	delayBetweenRetryAttempts	Integer	Check to enable caching of retrieved products.	3	false
Max product retrieval retry attempts	maxRetryAttempts	Integer	The time to wait (in seconds) between attempting to retry retrieving a product.	5	false
Product Retrieval Monitor Period	retrievalMonitorPeriod	Integer	The maximum number of attempts to retry retrieving a product.	false	false
Always Cache Product	cacheWhenCanceled	Boolean	How many seconds to wait and not receive product data before retrying to retrieve a product.	null	false

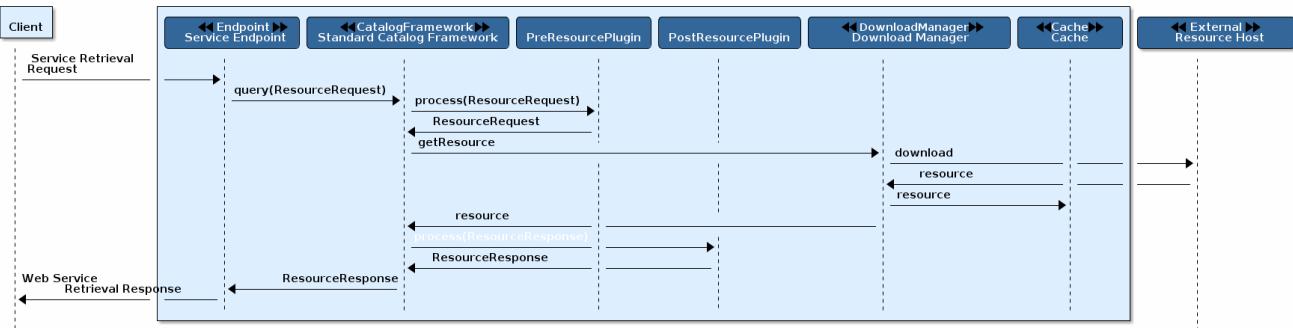


Figure 24. Product Retrieval Request

Notifications and Activities

DDF can send/receive notifications of "Activities" occurring in the system.

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

Activity events include the status and progress of actions that are being performed by the user, such as searches and downloads. Activities can be enabled by selecting "Show tasks" in the Standard Search UI configuration. 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.

28.2. Included Catalog Frameworks, Associated Components, and Configurations

These catalog frameworks are available in a standard DDF installation:

Catalog Fanout Framework

The Fanout Catalog Framework provides an implementation of the Catalog Framework that acts as a proxy, federating requests to all available sources.

Catalog Framework Camel Component

The catalog framework camel component supports creating, updating, and deleting metacards using the Catalog Framework from a Camel route.

Standard Catalog Framework

The Standard Catalog Framework provides the reference implementation of a Catalog Framework that implements all requirements of the Catalog API.

28.2.1. Standard Catalog Framework

The Standard Catalog Framework provides the reference implementation of a Catalog Framework that implements all requirements of the Catalog API. `CatalogFrameworkImpl` is the implementation of the DDF Standard Catalog Framework.

The Standard Catalog Framework is the core class of DDF. It provides the methods for create, update, delete, and resource retrieval (CRUD) operations on the `Sources`. By contrast, the Fanout Catalog Framework only allows for query and resource retrieval operations, no catalog modifications, and all queries are enterprise-wide.

Use this framework if:

- access to a catalog provider is required to create, update, and delete catalog entries.
- queries to specific sites are required.
- queries to only the local provider are required.

It is possible to have only remote Sources configured with no local `CatalogProvider` configured and be able to execute queries to specific remote sources by specifying the site name(s) in the query request.

The Standard Catalog Framework also maintains a list of `ResourceReaders` for resource retrieval operations. A resource reader is matched to the scheme (i.e., protocol, such as `file://`) in the URI of the resource specified in the request to be retrieved.

Site information about the catalog provider and/or any federated source(s) can be retrieved using the Standard Catalog Framework. Site information includes the source's name, version, availability, and the list of unique content types currently stored in the source (e.g., NITF). If no local catalog provider is configured, the site information returned includes site info for the catalog framework with no content types included.

Installing the Standard Catalog Framework

The Standard Catalog Framework is bundled as the `catalog-core-standardframework` feature and can be installed and uninstalled using the normal processes described in Configuration.

When this feature is installed, the Catalog Fanout Framework App feature `catalog-core-fanoutframework` should be uninstalled, as both catalog frameworks should not be installed simultaneously.

Configuring the Standard Catalog Framework

These are the configurable properties on the Standard Catalog Framework.

Table 84. Catalog Standard Framework

Name	ID	Type	Description	Default Value	Required
Enable Fanout Proxy	fanoutEnabled	Boolean	When enabled the Framework acts as a proxy, federating requests to all available sources. All requests are executed as federated queries and resource retrievals, allowing the framework to be the sole component exposing the functionality of all of its Federated Sources.	false	true
Enable Notifications	notificationEnabled	Boolean	Check to enable notifications.	true	false
Fanout tag blacklist	fanoutTagBlacklist	String	Ingest operations with tags in this list will be rejected.		true

Table 85. Standard Catalog Framework Exported Services

Registered Interface	Service Property	Value
ddf.catalog.federation.FederationStrategy	shortname	sorted
org.osgi.service.event.EventHandler	event.topics	ddf/catalog/event/CREATED, ddf/catalog/event/UPDATED, ddf/catalog/event/DELETED
ddf.catalog.CatalogFramework		
ddf.catalog.event.EventProcessor		
ddf.catalog.plugin.PostIngestPlugin		

Table 86. Standard Catalog Framework Imported Services

Registered Interface	Availability	Multiple
ddf.catalog.plugin.PostFederatedQueryPlugin	optional	true
ddf.catalog.plugin.PostIngestPlugin	optional	true
ddf.catalog.plugin.PostQueryPlugin	optional	true
ddf.catalog.plugin.PostResourcePlugin	optional	true
ddf.catalog.plugin.PreDeliveryPlugin	optional	true
ddf.catalog.plugin.PreFederatedQueryPlugin	optional	true
ddf.catalog.plugin.PreIngestPlugin	optional	true
ddf.catalog.plugin.PreQueryPlugin	optional	true
ddf.catalog.plugin.PreResourcePlugin	optional	true
ddf.catalog.plugin.PreSubscriptionPlugin	optional	true

Registered Interface	Availability	Multiple
<code>ddf.catalog.plugin.PolicyPlugin</code>	optional	true
<code>ddf.catalog.plugin.AccessPlugin</code>	optional	true
<code>ddf.catalog.resource.ResourceReader</code>	optional	true
<code>ddf.catalog.source.CatalogProvider</code>	optional	true
<code>ddf.catalog.source.ConnectedSource</code>	optional	true
<code>ddf.catalog.source.FederatedSource</code>	optional	true
<code>ddf.cache.CacheManager</code>		false
<code>org.osgi.service.event.EventAdmin</code>		false

Known Issues with Standard Catalog Framework

None.

28.2.2. Catalog Framework Camel Component

The Catalog Framework Camel Component supports creating, updating, and deleting metacards using the Catalog Framework from a Camel route.

URI Format

`catalog:framework`

Message Headers

Catalog Framework Producer

Header	Description
<code>operation</code>	the operation to perform using the Catalog Framework (possible values are CREATE UPDATE DELETE)

Sending Messages to Catalog Framework Endpoint

Catalog Framework Producer

In Producer mode, the component provides the ability to provide different inputs and have the Catalog framework perform different operations based upon the header values.

For the CREATE and UPDATE operation, the message body can contain a list of metacards or a single metocard object.

For the DELETE operation, the message body can contain a list of strings or a single string object. The string objects represent the IDs of metacards to be deleted. The exchange's "in" message will be set with the affected metacards. In the case of a CREATE, it will be updated with the created

metacards. In the case of the UPDATE, it will be updated with the updated metacards and with the DELETE it will contain the deleted metacards.

Table 87. Catalog Framework Camel Component Operations

Header	Message Body (Input)	Exchange Modification (Output)
operation = CREATE	List<Metocard> or Metocard	exchange.getIn().getBody() updated with List of Metacards created
operation = UPDATE	List<Metocard> or Metocard	exchange.getIn().getBody() updated with List of Metacards updated
operation = DELETE	List<String> or String (representing metocard IDs)	exchange.getIn().getBody() updated with List of Metacards deleted

Samples

This example demonstrates:

1. Reading in some sample data from the file system.
2. Using a Java bean to convert the data into a metocard.
3. Setting a header value on the Exchange.
4. Sending the Metocard to the Catalog Framework component for ingesting.

```
<route>
  <from uri="file:data/sampleData?noop=true"/>
    <bean ref="sampleDataToMetocardConverter" method="convertToMetocard"/>\ 
      <setHeader headerName="operation">
        <constant>CREATE</constant>
      </setHeader>
    <to uri="catalog:framework"/>
</route>
```

28.3. Developing Complementary Catalog Frameworks

DDF and the underlying OSGi technology can serve as a robust infrastructure for developing frameworks that complement the Catalog.

28.3.1. Recommendations for Framework Development

- Provide extensibility similar to that of the Catalog.
 - Provide a stable API with interfaces and simple implementations (refer to http://www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html).
- Make use of the Catalog wherever possible to store, search, and transform information.

- Utilize OSGi standards wherever possible.
 - ConfigurationAdmin
 - MetaType
- Utilize the sub-frameworks available in DDF.
 - Karaf
 - CXF
 - PAX Web and Jetty

28.3.2. Catalog Framework Reference

The Catalog Framework can be requested from the OSGi Service Registry.

Blueprint Service Reference

```
<reference id="catalogFramework" interface="DDF.catalog.CatalogFramework" />
```

Methods

The `CatalogFramework` provides convenient methods to transform `Metacards` and `QueryResponses` using a reference to the `CatalogFramework`.

Create, Update, and Delete Methods

Create, Update, and Delete (CUD) methods add, change, or remove stored metadata in the local Catalog Provider.

Example Create, Update, Delete Methods

```
public CreateResponse create(CreateRequest createRequest) throws IngestException,  
SourceUnavailableException;  
public UpdateResponse update(UpdateRequest updateRequest) throws IngestException,  
SourceUnavailableException;  
public DeleteResponse delete(DeleteRequest deleteRequest) throws IngestException,  
SourceUnavailableException;
```

CUD operations process `PolicyPlugin`, `AccessPlugin`, and `PreIngestPlugin` instances before execution and `PostIngestPlugin` instances after execution.

Query Methods

Query methods search metadata from available Sources based on the `QueryRequest` properties and Federation Strategy. Sources could include Catalog Provider, Connected Sources, and Federated Sources.

Example Query Methods

```
public QueryResponse query(QueryRequest query) throws UnsupportedQueryException,  
,SourceUnavailableException, FederationException;  
public QueryResponse query(QueryRequest queryRequest, FederationStrategy strategy)  
throws SourceUnavailableException, UnsupportedQueryException, FederationException;
```

Query requests process [PolicyPlugin](#), [AccessPlugin](#), and [PreQueryPlugin](#) instances before execution and [PolicyPlugin](#), [AccessPlugin](#), and [PostQueryPlugin](#) instances after execution.

Resource Methods

Resource methods retrieve products from Sources.

Example Resource Methods

```
public ResourceResponse getEnterpriseResource(ResourceRequest request)  
throws IOException, ResourceNotFoundException, ResourceNotSupportedException;  
public ResourceResponse getLocalResource(ResourceRequest request) throws IOException,  
ResourceNotFoundException, ResourceNotSupportedException;  
public ResourceResponse getResource(ResourceRequest request, String resourceSiteName)  
throws IOException, ResourceNotFoundException, ResourceNotSupportedException;
```

Resource requests process `PreResourcePlugin`'s before execution and `PostResourcePlugin`'s after execution.

Source Methods

Source methods can get a list of Source identifiers or request descriptions about Sources.

Example Source Methods

```
public Set<String> getSourceIds();  
public SourceInfoResponse getSourceInfo(SourceInfoRequest sourceInfoRequest) throws  
SourceUnavailableException;
```

Transform Methods

Transform methods provide convenience methods for using Metocard Transformers and Query Response Transformers.

Transform Methods

```
// Metocard Transformer  
public BinaryContent transform(Metocard metocard, String transformerId, Map<String  
, Serializable> requestProperties) throws CatalogTransformerException;  
  
// Query Response Transformer  
public BinaryContent transform(SourceResponse response, String transformerId, Map  
<String, Serializable> requestProperties) throws CatalogTransformerException;
```

Included Catalog Frameworks

Catalog API

description

Implementing Catalog Methods

Query Response Transform Example

```
// inject CatalogFramework instance or retrieve an instance  
private CatalogFramework catalogFramework;  
  
public RSSEndpoint(CatalogFramework catalogFramework)  
{  
    this.catalogFramework = catalogFramework ;  
    // implementation  
}  
  
// Other implementation details ...  
  
private void convert(QueryResponse queryResponse ) {  
    // ...  
    String transformerId = "rss";  
  
    BinaryContent content = catalogFramework.transform(queryResponse, transformerId,  
null);  
  
    // ...  
}
```

Dependency Injection

Using Blueprint or another injection framework, transformers can be injected from the OSGi Service Registry.

Blueprint Service Reference

```
<reference id="[[Reference Id]]" interface="DDF.catalog.transform.[[Transformer Interface Name]]" filter="(shortname=[[Transformer Identifier]])" />
```

Each transformer has one or more `transform` methods that can be used to get the desired output.

Input Transformer Example

```
DDF.catalog.transform.InputTransformer inputTransformer = retrieveInjectedInstance() ;  
  
Metocard entry = inputTransformer.transform(messageInputStream);
```

Metocard Transformer Example

```
DDF.catalog.transform.MetocardTransformer metocardTransformer =  
retrieveInjectedInstance() ;  
  
BinaryContent content = metocardTransformer.transform(metocard, arguments);
```

Query Response Transformer Example

```
DDF.catalog.transform.QueryResponseTransformer queryResponseTransformer =  
retrieveInjectedInstance() ;  
  
BinaryContent content = queryResponseTransformer.transform(sourceSesponse, arguments);
```

OSGi Service Registry

IMPORTANT In the vast majority of cases, working with the OSGi Service Reference directly should be avoided. Instead, dependencies should be injected via a dependency injection framework like Blueprint.

Transformers are registered with the OSGi Service Registry. Using a `BundleContext` and a filter, references to a registered service can be retrieved.

OSGi Service Registry Reference Example

```
ServiceReference[] refs =  
bundleContext.getServiceReferences(DDF.catalog.transform.InputTransformer.class  
.getName(), "(shortname=" + transformerId + ")");  
InputTransformer inputTransformer = (InputTransformer) context.getService(refs[0]);  
Metocard entry = inputTransformer.transform(messageInputStream);
```

28.3.3. Metocard Type Definition File

To define Metocard Types, the definition file must have a `metocardTypes` key in the root object.

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

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

```
{  
  "metocardTypes": [  
    {  
      "type": "my-metocard-type",  
      "attributes": {...}  
    }  
  ]  
}
```

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

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

NOTE

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

Multiple Metocard Types in a Single File

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

28.3.4. Global Attribute Validators File

To define Validators, the 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:

2. **validator** Possible Values

- **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]

Example Validator Definition

```
{  
  "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"]  
      }  
    ]  
  }  
}
```

28.3.5. Attribute Type Definition File

To define Attribute Types, the 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.

Attribute Types

```
{  
  "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.

3. Required Attribute Definitions

`stored`

If true, the value of the attribute should be stored in the underlying datastore. Some attributes may only be indexed or used in transit and do not need to be persisted.

`indexed`

If true, then the value of the attribute should be included in the datastore's index and therefore be part of query evaluation.

`tokenized`

Only applicable to STRING_TYPE attributes, if true then stopwords and punctuation will be stripped prior to storing and/or indexing. If false, only an exact string will match.

`multi-valued`

If true, then the attribute values will be Lists of the attribute type rather than single values.

The `type` attribute must also be included and must have one of the allowed values:

4. type Attribute Possible 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:

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  
    }  
  }  
}
```

28.3.6. Default Attribute Values

To define default attribute values, the 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 `metocardTypes`.

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

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 `metocardTypes` key is optional. If it is left out, then the default attribute value will be applied to every metocard that has that attribute. It can be thought of as a 'global' default value. If the `metocardTypes` key is included, then its value must be a list of strings where each string is the name of a metocard 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 metocard type, the default value for the specific metocard 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"
    }
  ]
}
```

28.3.7. Attribute Injection Definition

Attribute injections are defined attributes that will be injected into all metocard types or into specific metocard types. This capability allows metocard types to be extended with new attributes.

To define attribute injections, the definition file must have an `inject` key in the root object.

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

The value of `inject` is simply a list of objects where each object contains the key `attribute` and optionally `metacardTypes`.

```
{
  "inject": [
    {
      "attribute": ....,
      "metacardTypes": [...]
    }
  ]
}
```

The value corresponding to the `attribute` key is the name of the attribute to inject.

The `metocardTypes` key is optional. If it is left out, then the attribute will be injected into every metocard type. In that case it can be thought of as a 'global' attribute injection. If the `metocardTypes` key is included, then its value must be a list of strings where each string is the name of a metocard type. In this case, the attribute will be injected only into metocard types that match one of the types given in the list.

Example:

```
{  
  "inject": [  
    {  
      "attribute": "rating"  
    },  
    {  
      "attribute": "cloud-cover",  
      "metocardTypes": ["nitf", "video"]  
    }  
  ]  
}
```

NOTE Attributes must be registered in the attribute registry (see the `AttributeRegistry` interface) in order to be injected into metocard types. For example, attributes defined in JSON definition files are placed in the registry, so they can be injected.

28.4. Catalog Development Fundamentals

This section introduces the fundamentals of working with the Catalog API and the OGC Filter for Queries.

28.4.1. Simple Catalog API Implementations

The Catalog API implementations, which are denoted with the suffix of `Impl` on the Java file names, have multiple purposes and uses:

- First, they provide a good starting point for other developers to extend functionality in the framework. For instance, extending the `MetocardImpl` allows developers to focus less on the inner workings of DDF and more on the developer's intended purposes and objectives.
- Second, the Catalog API Implementations display the proper usage of an interface and an interface's intentions. Also, they are good code examples for future implementations. If a developer does not want to extend the simple implementations, the developer can at least have a working code reference on which to base future development.

28.4.2. Use of the Whiteboard Design Pattern

The Catalog makes extensive use of the Whiteboard Design Pattern. Catalog Components are registered as services in the OSGi Service Registry, and the Catalog Framework or any other clients tracking the OSGi Service Registry are automatically notified by the OSGi Framework of additions

and removals of relevant services.

The Whiteboard Design Pattern is a common OSGi technique that is derived from a technical whitepaper provided by the OSGi Alliance in 2004. It is recommended to use the Whiteboard pattern over the Listener pattern in OSGi because it provides less complexity in code (both on the client and server sides), fewer deadlock possibilities than the Listener pattern, and closely models the intended usage of the OSGi framework.

29. Developing Sources

29.1. Developing Sources

Sources are components that enable DDF to talk to back-end services. They let DDF perform query and ingest operations on catalog stores and query operations on federated sources.

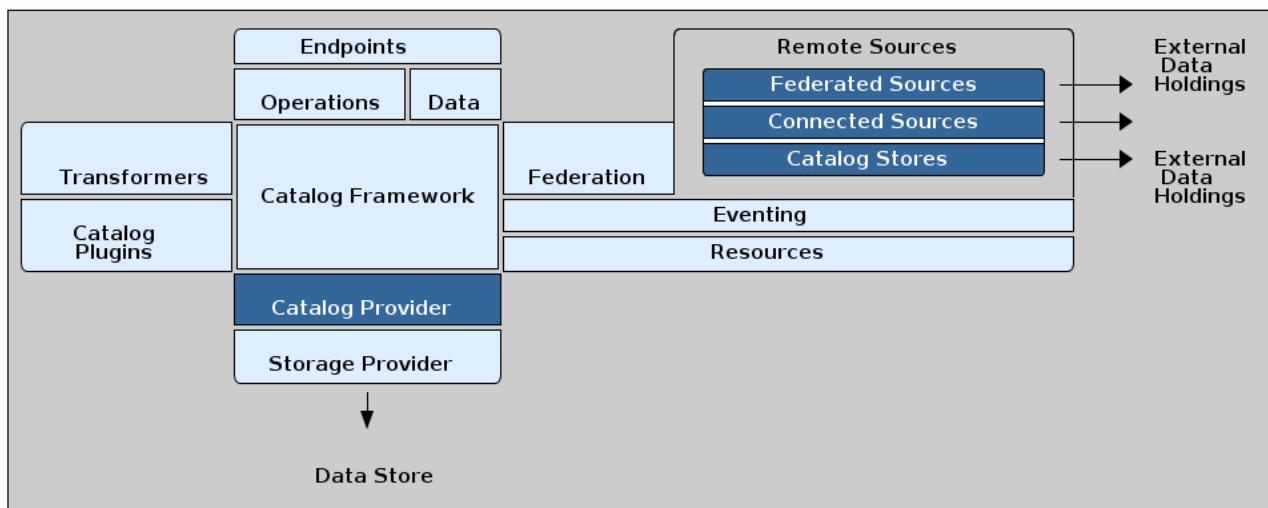


Figure 25. Source Architecture

29.1.1. Implement a Source Interface

There are three types of sources that can be created. All of these types of sources can perform a query operation. Operating on queries is the foundation for all sources. All of these sources must also be able to return their availability and the list of content types currently stored in their back-end data stores.

Catalog Provider

`ddf.catalog.source.CatalogProvider` is used to communicate with back-end storage and allows for Query and Create/Update/Delete operations.

Federated Source

`ddf.catalog.source.FederatedSource` is used to communicate with remote systems and only allows query operations.

Connected Source

`ddf.catalog.source.ConnectedSource` is similar to a Federated Source with the following exceptions:

- Queried on all local queries
- `SiteName` is hidden (masked with the DDF sourceId) in query results
- `SiteService` does not show this Source's information separate from DDF's.

Catalog Store

`catalog.store.interface` is used to

The procedure for implementing any of the source types follows a similar format:

1. Create a new class that implements the specified Source interface and `ConfiguredService`.
2. Implement the required methods.
3. Create an OSGi descriptor file to communicate with the OSGi registry. (Refer to [OSGi Services](#)).
 - a. Import DDF packages.
 - b. Register source class as service to the OSGi registry.
4. Deploy to DDF.

The `factory-pid` property of the metatype must contain one of the following in the name:
service, Service, source, Source

Developing Catalog Providers

Create a custom implementation of a catalog provider.

1. Create a Java class that implements `CatalogProvider`.

```
public class TestCatalogProvider implements ddf.catalog.source.CatalogProvider
```

2. Implement the required methods from the `ddf.catalog.source.CatalogProvider` interface.

```
public CreateResponse create(CreateRequest createRequest) throws IngestException; public  
UpdateResponse update(UpdateRequest updateRequest) throws IngestException; public  
DeleteResponse delete(DeleteRequest deleteRequest) throws IngestException;
```

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog, ddf.catalog.source
```

4. Export the service to the OSGi registry.

Catalog Provider Blueprint example

```
<service ref="[[TestCatalogProvider]]" interface="ddf.catalog.source.CatalogProvider"  
/>
```

See the existing [Catalog Provider list](#) for examples of Catalog Providers included in DDF.

Developing Federated Sources

1. Create a Java class that implements `FederatedSource` and `ConfiguredService`.

```
public class TestFederatedSource implements ddf.catalog.source.FederatedSource,  
ddf.catalog.service.ConfiguredService
```

2. Implement the required methods of the `ddf.catalog.source.FederatedSource` and `ddf.catalog.service.ConfiguredService` interfaces.

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

`Import-Package: ddf.catalog, ddf.catalog.source`

4. Export the service to the OSGi registry.

Federated Source Blueprint example

```
<service ref="[[TestFederatedSource]]" interface="ddf.catalog.source.FederatedSource"  
/>
```

NOTE

A code example of a Federated Source delivered with DDF can be found in [Solr Cache Source](#)

Developing Connected Sources

Create a custom implementation of a connected source.

1. Create a Java class that implements `ConnectedSource` and `ConfiguredService`.

```
public class TestConnectedSource implements ddf.catalog.source.ConnectedSource,  
ddf.catalog.service.ConfiguredService
```

2. Implement the required methods of the `ddf.catalog.source.ConnectedSource` and `ddf.catalog.service.ConfiguredService` interfaces.

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

`Import-Package: ddf.catalog, ddf.catalog.source`

4. Export the service to the OSGi registry.

Connected Source Blueprint example

```
<service ref="[[TestConnectedSource]]" interface="ddf.catalog.source.ConnectedSource"  
/>
```

IMPORTANT

In some Providers that are created, there is a need to make Web Service calls through JAXB clients. It is best NOT to create your JAXB client as a global variable. There may be intermittent failures with the creation of Providers and federated sources when clients are created in this manner. Create your JAXB clients every single time within the methods that require it in order to avoid this issue.

Exception Handling

In general, sources should only send information back related to the call, not implementation details.

Exception Examples

Follow these guidelines for effective exception handling:

- Use a "Site XYZ not found" message rather than the full stack trace with the original site not found exception.
- If the caller issues a malformed search request, return an error describing the right form, or specifically what was not recognized in the request. Do not return the exception and stack trace where the parsing broke.
- If the caller leaves something out, do not return the null pointer exception with a stack trace, rather return a generic exception with the message "xyz was missing."

External Resources for Developing Sources

- [Three Rules for Effective Exception Handling](#)

30. Transformers

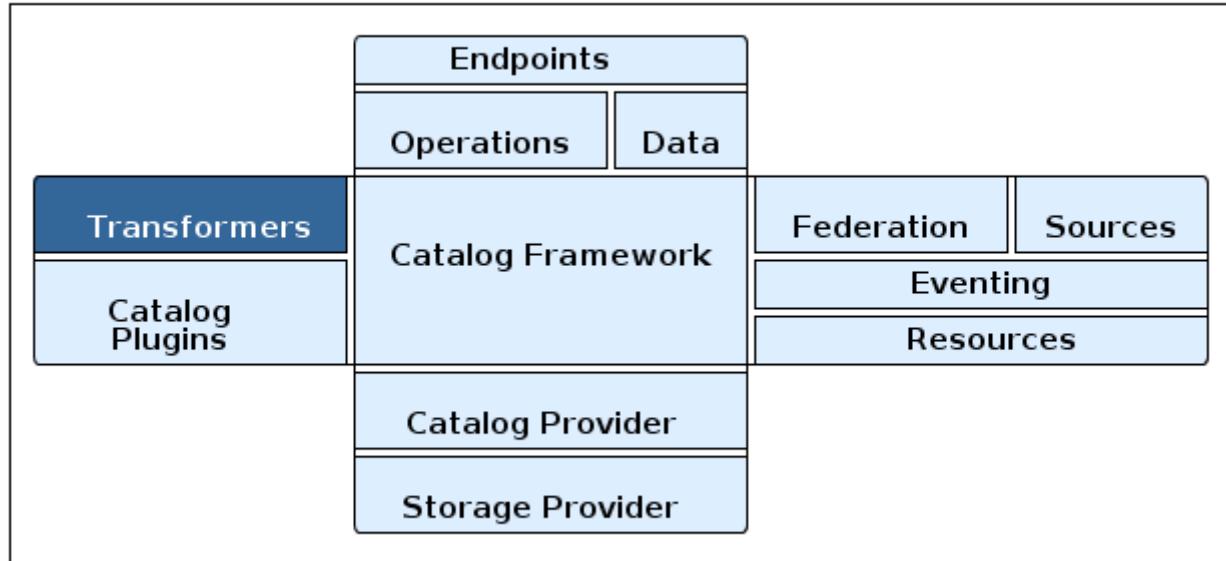


Figure 26. Transformers

Transformers transform data to and from various formats. Transformers can be categorized on the basis of when they are invoked and used. The existing types are Input transformers, Metocard transformers, and Query Response transformers. Additionally, XSLT transformers are provided to aid in developing custom, lightweight Metocard and Query Response transformers.

Transformers are utility objects used to transform a set of standard DDF components into a desired

format, such as into PDF, GeoJSON, XML, or any other format. For instance, a transformer can be used to convert a set of query results into an easy-to-read GeoJSON format ([GeoJSON Transformer](#)) or convert a set of results into a RSS feed that can be easily published to a URL for RSS feed subscription. A major benefit of transformers is that they can be registered in the OSGi Service Registry so that any other developer can access them based on their standard interface and self-assigned identifier, referred to as its "shortname." Transformers are often used by endpoints for data conversion in a system standard way. Multiple endpoints can use the same transformer, a different transformer, or their own published transformer.

WARNING

The current transformers only work for UTF-8 characters and do not support Non-Western Characters (e.g., Hebrew). It is recommended not to use international character sets as they may not be displayed properly.

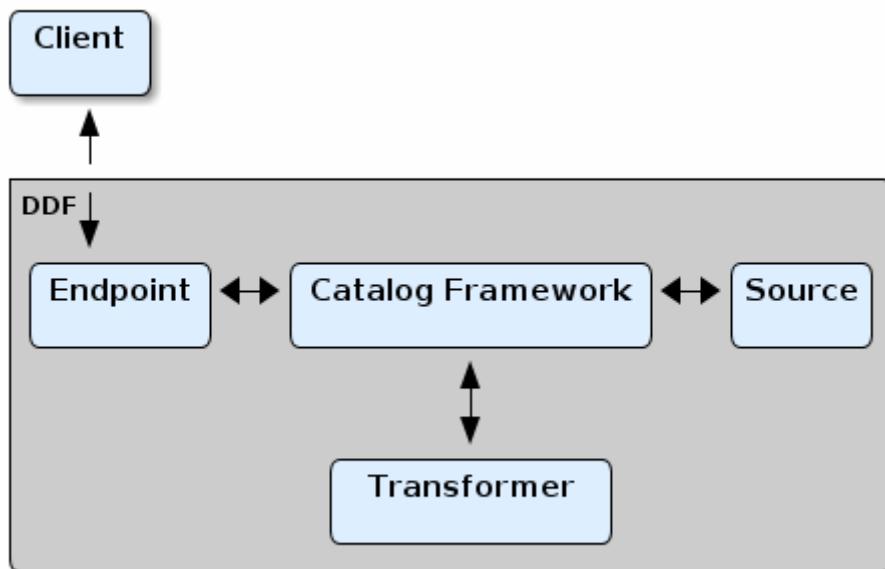


Figure 27. Communication Diagram

Transformers are used to alter the format of a resource or its metadata to or from the catalog's metocard format.

Types of Transformers

[Input Transformers](#)

Input Transformers create metacards from input. An input transformer transforms raw data (text/binary) into a Metocard. Once converted to a Metocard, the data can be used in a variety of ways, such as in an [UpdateRequest](#), [CreateResponse](#), or within Catalog Endpoints or Sources. For instance, an input transformer could be used to receive and translate XML into a Metocard so that it can be placed within a [CreateRequest](#) in order to be ingested within the Catalog. Input transformers should be registered within the Service Registry with the interface `ddf.catalog.transform.InputTransformer` in order to notify some Catalog components of any new transformers.

[Metocard Transformers](#)

Metocard Transformers translate a metocard from catalog metadata to a specific data format.

Query Response Transformers

Query Response transformers convert query responses into other data formats.

30.1. Input Transformers

The following input transformers are available with a standard installation of DDF:

GeoJSON Input Transformer

Translates specific GeoJSON into a Catalog metacard.

PdfInputTransformer

Translates PDF files into Catalog Metacards.

PPTX Input Transformer

Translates Microsoft PowerPoint (OOXML only) documents into Catalog Metacards.

RegistryTransformer

Creates Registry metacards from **ebrim** messages.

Tika Input Transformer

Translates Microsoft Word, Microsoft Excel, Microsoft PowerPoint, OpenOffice Writer, and PDF documents into Catalog Metacards.

Video Input Transformer

Creates Catalog metacards from certain video file types.

XmlInputTransformer

Translates XML documents into Catalog Metacards.

30.1.1. GeoJSON Input Transformer

The GeoJSON input transformer is responsible for translating specific GeoJSON into a Catalog metacard.

Installing the GeoJSON Input Transformer

The GeoJSON Input Transformer is installed by default with a standard installation.

Configuring the GeoJSON Input Transformer

The GeoJSON Input Transformer has no configurable properties.

Using the GeoJSON Input Transformer

Using the REST Endpoint, for example, HTTP POST a GeoJSON metacard to the Catalog. Once the REST Endpoint receives the GeoJSON Metacard, it is converted to a Catalog metacard.

Example HTTP POST of a Local `metocard.json` File Using the Curl Command

```
curl -X POST -i -H "Content-Type: application/json" -d "@metocard.json"  
https://localhost:8993/services/catalog
```

Conversion to a Metocard

A [GeoJSON object](#) consists of a single JSON object. The single JSON object can be a geometry, a feature, or a [FeatureCollection](#). This input transformer only converts "feature" objects into metacards. This is a natural choice since feature objects include geometry information and a list of properties. For instance, if only a geometry object is passed, such as only a [LineString](#), not enough information is available to create a metocard. This input transformer currently does not handle [FeatureCollections](#) either, but could be supported in the future.

IMPORTANT

Cannot create Metocard from this limited GeoJSON

```
{ "type": "LineString",  
  "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]  
}
```

The following sample *will* create a valid metocard:

Sample Parseable GeoJson (Point)

```
{  
  "properties": {  
    "title": "myTitle",  
    "thumbnail": "CA==",  
    "resource-uri": "http://example.com",  
    "created": "2012-09-01T00:09:19.368+0000",  
    "metadata-content-type-version": "myVersion",  
    "metadata-content-type": "myType",  
    "metadata": "<xml></xml>",  
    "modified": "2012-09-01T00:09:19.368+0000"  
  },  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [  
      30.0,  
      10.0  
    ]  
  }  
}
```

In the current implementation, [Metocard.LOCATION](#) is not taken from the properties list as WKT, but instead interpreted from the [geometry](#) JSON object. The geometry object is formatted according to the [GeoJSON](#) standard. Dates are in the ISO 8601 standard. White space is ignored, as in most cases

with JSON. Binary data is accepted as Base64. XML must be properly escaped, such as what is proper for normal JSON.

Only Required Attributes are recognized in the properties currently.

Metocard Extensibility

GeoJSON supports custom, extensible properties on the incoming GeoJSON using DDF's extensible metocard support. To have those customized attributes understood by the system, a corresponding [MetocardType](#) must be registered with the [MetocardTypeRegistry](#). That [MetocardType](#) must be specified by name in the metocard-type property of the incoming GeoJSON. If a [MetocardType](#) is specified on the GeoJSON input, the customized properties can be processed, cataloged, and indexed.

Sample GeoJSON input

```
{  
  "properties": {  
    "title": "myTitle",  
    "thumbnail": "CA==",  
    "resource-uri": "http://example.com",  
    "created": "2012-09-01T00:09:19.368+0000",  
    "metadata-content-type-version": "myVersion",  
    "metadata-content-type": "myType",  
    "metadata": "<xml></xml>",  
    "modified": "2012-09-01T00:09:19.368+0000",  
    "min-frequency": "10000000",  
    "max-frequency": "20000000",  
    "metocard-type": "ddf.metocard.custom.type"  
  },  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [  

```

When the GeoJSON Input Transformer gets GeoJSON with the [MetocardType](#) specified, it will perform a lookup in the [MetocardTypeRegistry](#) to obtain the specified [MetocardType](#) in order to understand how to parse the GeoJSON. If no [MetocardType](#) is specified, the GeoJSON Input Transformer will assume the default [MetocardType](#). If an unregistered [MetocardType](#) is specified, an exception will be returned to the client indicating that the [MetocardType](#) was not found.

Usage Limitations of the GeoJSON Input Transformer

The GeoJSON Input Transformer does not handle multiple geometries.

30.1.2. PDF Input Transformer

The PDF Input Transformer is responsible for translating a PDF document into a Catalog Metacard.

Installing the PDF Input Transformer

The PDF Transformer is installed by default with a standard installation in the Catalog application.

Configuring the PDF Input Transformer

The PDF Input Transformer has no configurable properties.

30.1.3. PPTX Input Transformer

The PPTX Input Transformer is the input transformer responsible for translating Microsoft PowerPoint (OOXML only) documents into a Catalog Metacard. This input transformer utilizes [Apache Tika](#) for basic metadata and [Apache POI](#) for thumbnail creation. The PPTX Input Transformer's main purpose is to ingest PPTX documents into the DDF Content Repository and the Metadata Catalog.

The PPTX Input Transformer will take precedence over the Tika Input Transformer for PPTX documents.

Installing the PPTX Input Transformer

This transformer is installed by default with a standard installation in the Catalog application.

Configuring the PPTX Input Transformer

The PPTX Input Transformer has no configurable properties.

Using the PPTX Input Transformer

Use the PPTX Input Transformer for ingesting Microsoft PowerPoint (OOXML only) documents into the DDF Content Repository and/or the Metadata Catalog.

Table 88. PPTX Input Transformer Service Properties

Key	Value
mime-type	* application/vnd.openxmlformats-officedocument.presentationml.presentation
id	pptx
title	PPTX Input Transformer
description	Default Input Transformer for the application/vnd.openxmlformats-officedocument.presentationml.presentation mime type.

PPTX Input Transformer Implementation Details

This input transformer maps the metadata common across all mime types to applicable metocard

attributes in the default **MetacardType** and adds a thumbnail of the first page in the PPTX document.

30.1.4. Registry Transformer

The Registry Transformer creates Registry metacards from **ebrim** messages. It also returns the **ebrim** message from the metocard metadata.

Installing the Registry Transformer

The Registry Transformer is not installed by default with a standard installation, but with the Registry application.

To install:

1. [Install Registry](#) application.

Configuring the Registry Transformer

The Registry Transformer has no configurable properties.

30.1.5. Tika Input Transformer

The Tika Input Transformer is the default input transformer responsible for translating Microsoft Word, Microsoft Excel, Microsoft PowerPoint, OpenOffice Writer, and PDF documents into a Catalog Metacard. This input transformer utilizes <https://tika.apache.org>[Apache Tika] to provide basic support for these mime types. As such, the metadata extracted from these types of documents is the metadata that is common across all of these document types, e.g., creation date, author, last modified date, etc. The Tika Input Transformer's main purpose is to ingest these types of content into the Metadata Catalog.

The Tika input transformer is given a service ranking (priority) of -1 so that it is guaranteed to be the last input transformer that is invoked. This allows any registered input transformer that are more specific for any of these document types to be invoked instead of this rudimentary input transformer.

Installing the Tika Input Transformer

This transformer is installed by default with a standard installation in the Catalog.

Configuring the Tika Input Transformer

The Tika Input Transformer has no configurable properties.

Using the Tika Input Transformer

Use the Tika Input Transformer for ingesting Microsoft documents, OpenOffice documents, or PDF documents into the Catalog.

Table 89. Tika Input Transformer Service Properties

Key	Value
mime-type	* application/pdf * application/vnd.openxmlformats-officedocument.wordprocessingml.document * application/vnd.openxmlformats-officedocument.spreadsheetml.sheet * application/vnd.openxmlformats-officedocument.presentationml.presentation * application/vnd.openxmlformats-officedocument.presentationml.presentation * application/vnd.ms-powerpoint.presentation.macroenabled.12 * application/vnd.ms-powerpoint.slideshow.macroenabled.12 * application/vnd.openxmlformats-officedocument.presentationml.slideshow * application/vnd.ms-powerpoint.template.macroenabled.12 * application/vnd.oasis.opendocument.text
shortname	
id	tika
title	Tika Input Transformer
description	Default Input Transformer for all mime types.
service.ranking	-1

Tika Input Transformer Implementation Details

This input transformer maps the metadata common across all mime types to applicable metocard attributes in the default MetacardType.

30.1.6. Video Input Transformer

The video input transformer is responsible for creating Catalog metacards from certain video file types. Currently, it is responsible for handling MPEG-2 transport streams as well as MPEG-4, AVI, MOV, and WMV videos. This input transformer uses [Apache Tika](#) to extract basic metadata from the video files and applies more sophisticated methods to extract more meaningful metadata from these types of video.

Installing the Video Input Transformer

This transformer is installed by default with a standard installation in the Catalog application.

Configuring the Video Input Transformer

The Video Input Transformer has no configurable properties.

Using the Video Input Transformer

Use the video input transformer for ingesting video files into the Catalog.

Table 90. Video Input Transformer Service Properties

Key	Value
mime-type	* video/avi * video/msvideo * video/vnd.avi * video/x-msvideo * video/mp4 * video/MP2T * video/mpeg * video/quicktime * video/wmv * video/x-ms-wmv
shortname	video
id	video
description	Detects and extracts metadata from various video file formats.

30.1.7. XML Input Transformer

The XML Input Transformer is responsible for translating an XML document into a Catalog Metocard.

Installing the XML Input Transformer

The XML Input Transformer is installed by default with a standard installation in the Catalog application.

Configuring the XML Input Transformer

The XML Input Transformer has no configurable properties.

30.2. Developing Input Transformers

DDF supports the creation of custom input transformers for use cases not covered by the included implementations.

1. Create a new Java class that implements `ddf.catalog.transform.InputTransformer`.

```
public class SampleInputTransformer implements ddf.catalog.transform.InputTransformer
```

2. Implement the transform methods.

```
public Metocard transform(InputStream input) throws IOException,  
CatalogTransformerException  
public Metocard transform(InputStream input, String id) throws IOException,  
CatalogTransformerException
```

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.transform
```

4. Create an OSGi descriptor file to communicate with the OSGi Service Registry (described in the [Working with OSGi](#) section). Export the service to the OSGi Registry and declare service properties.

Input Transformer Blueprint Descriptor Example

```
...
<service ref="[[SampleInputTransformer]]" interface=
"ddf.catalog.transform.InputTransformer">
    <service-properties>
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Input Transformer]]" />
        <entry key="description" value="[[A new transformer for metocard input.]]" />
    </service-properties>
</service>
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Table 91. Input Transformer Variable Descriptions / Blueprint Service Properties

Key	Description of Value	Example
<code>shortname</code>	(Required) An abbreviation for the return-type of the BinaryContent being sent to the user.	<i>atom</i>
<code>title</code>	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	<i>Atom Entry Transformer Service</i>
<code>description</code>	(Optional) A short, human-readable description that describes the functionality of the service and the output.	<i>This service converts a single metocard xml document to an atom entry element.</i>

30.2.1. Create an XML Input Transformer using SaxEventHandlers

If the transformer will transform XML, (as opposed to JSON or a Word document, for example) there is a simpler solution than fully implementing a `MetacardValidator`. DDF includes an extensible, configurable `XmlInputTransformer`. This transformer can be instantiated via blueprint as a managed service factory and configured via metatype. The `XmlInputTransformer` takes a configuration of `SaxEventHandlers`. A `SaxEventHandler` is a class that handles SAX Events (a very fast XML parser) to parse metadata and create metacards. As many `SaxEventHandlers` as needed can be implemented and included in the `XmlInputTransformer` configuration. See the `catalog-transformer-streaming-impl` bundle for examples (`XmlSaxEventHandlerImpl` which parses the DDF Metacard XML Metadata and the `GmlHandler` which parses GML 2.0) Each `SaxEventHandler` implementation has a `SaxEventHandlerFactory` associated with it. The `SaxEventHandlerFactory` is responsible for instantiating new `SaxEventHandlers` - each transform request gets a new instance of `XmlInputTransformer` and set of `SaxEventHandlers` to be thread- and state-safe.

The following diagrams intend to clarify implementation details:

The `XmlInputTransformer` Configuration diagram shows the `XmlInputTransformer` configuration, which is configured using the metatype and has the `SaxEventHandlerFactory` ids. Then, when a transform request is received, the `ManagedServiceFactory` instantiates a new `XmlInputTransformer`. This `XmlInputTransformer` then instantiates a new `SaxEventHandlerDelegate` with the configured

`SaxEventHandlersFactory` ids. The factories all in turn instantiate a `SaxEventHandler`. Then, the `SaxEventHandlerDelegate` begins parsing the XML input document, handing the SAX Events off to each `SaxEventHandler`, which handle them if they can. After parsing is finished, each `SaxEventHandler` returns a list of `Attributes` to the `SaxEventHandlerDelegate` and `XmlInputTransformer` which add the attributes to the metocard and then return the fully constructed metocard.

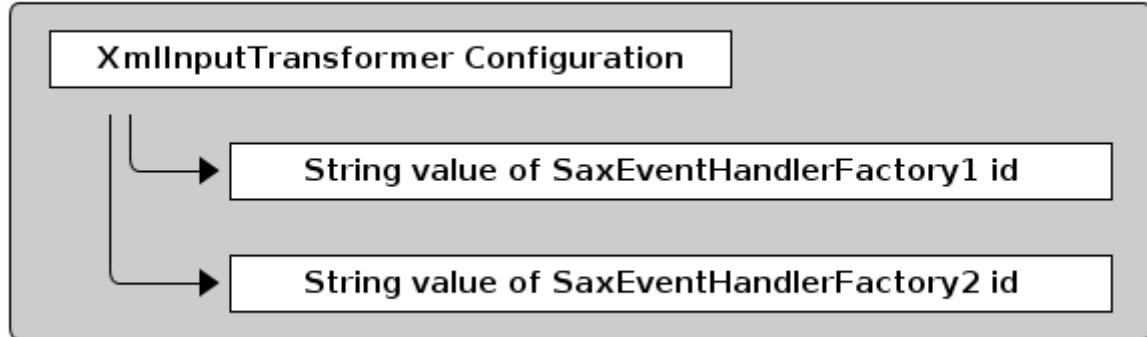


Figure 28. `XMLInputTransformer Configuration`

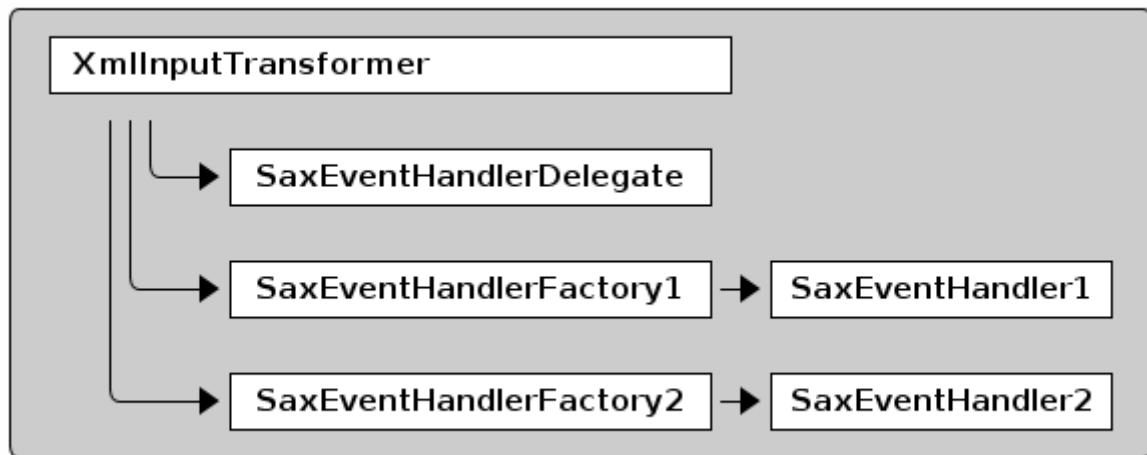


Figure 29. `XMLInputTransformer SaxEventHandlerDelegate Configuration`

For more specific details, see the Javadoc for the `org.codice.ddf.transformer.xml.streaming.*` package. Additionally, see the source code for the `org.codice.ddf.transformer.xml.streaming.impl.GmlHandler.java`, `org.codice.ddf.transformer.xml.streaming.impl.GmlHandlerFactory`, `org.codice.ddf.transformer.xml.streaming.impl.XmlInputTransformerImpl`, and `org.codice.ddf.transformer.xml.streaming.impl.XmlInputTransformerImplFactory`.

NOTE

1. The `XmlInputTransformer` & `SaxEventHandlerDelegate` create and configure themselves based on String matches of the configuration ids with the `SaxEventHandlerFactory` ids, so ensure these match.
2. The `XmlInputTransformer` uses a `DynamicMetacardType`. This is pertinent because a metacards attributes are only stored in Solr if they are declared on the `MetacardType`. Since the `DynamicMetacardType` is constructed dynamically, attributes are declared by the `SaxEventHandlerFactory` that parses them, as opposed to the `MetacardType`. See `org.codice.ddf.transformer.xml.streaming.impl.XmlSaxEventHandlerFactoryImpl.java` vs `ddf.catalog.data.impl.BasicTypes.java`

30.2.2. Create an Input Transformer Using Apache Camel

Alternatively, make an Apache Camel route in a blueprint file and deploy it using a feature file or via hot deploy.

30.2.3. Input Transformer Design Pattern

Follow this design pattern for compatibility:

From

When using `from catalog:inputtransformer?id=text/xml`, an Input Transformer will be created and registered in the OSGi registry with an id of `text/xml`.

To

When using `to catalog:inputtransformer?id=text/xml`, an Input Transformer with an id matching `text/xml` will be discovered from the OSGi registry and invoked.

Table 92. InputTransformer Message Formats

Exchange Type	Field	Type
Request (comes from <code><from></code> in the route)	body	<code>java.io.InputStream</code>
Response (returned after called via <code><to></code> in the route)	body	<code>ddf.catalog.data.Metacard</code>

InputTransformer Creation Example

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <route>
            <from uri="catalog:inputtransformer?mimeType=RAW(id=text/xml;id=vehicle)" />
            <to uri="xslt:vehicle.xslt" /> <!-- must be on classpath for this bundle
-->
            <to uri=
"catalog:inputtransformer?mimeType=RAW(id=application/json;id=geojson)" />
        </route>
    </camelContext>
</blueprint>
```

TIP Its always a good idea to wrap the `mimeType` value with the `RAW` parameter as shown in the example above. This will ensure that the value is taken exactly as is, and is especially useful when you are using special characters.

Table 93. InputTransformer Creation Details

Line Number	Description
1	Defines this as an Apache Aries blueprint file.
2	Defines the Apache Camel context that contains the route.
3	Defines start of an Apache Camel route.
4	Defines the endpoint/consumer for the route. In this case it is the DDF custom catalog component that is an InputTransformer registered with an id of <code>text/xml;id=vehicle</code> meaning it can transform an InputStream of vehicle data into a metocard. Note that the specified XSL stylesheet must be on the classpath of the bundle that this blueprint file is packaged in.
5	Defines the XSLT to be used to transform the vehicle input into GeoJSON format using the Apache Camel provided XSLT component.

NOTE An example of using an Apache Camel route to define an `InputTransformer` in a blueprint file and deploying it as a bundle to an OSGi container can be found in the DDF SDK examples at [DDF/sdk/sample-transformers/xslt-identity-input-transformer](#)

30.3. Metocard Transformers

The following metocard transformers are available with a standard installation of DDF.

GeoJSON Metocard Transformer

Translates a Metocard into GeoJSON.

HTML Metacard Transformer

Translates a metacard into an HTML formatted document.

KML Metacard Transformer

implements [KML]. Translates a metacard into a KML-formatted document.

KML Style Mapper

Provides the ability for the **KmlTransformer** to map a KML Style URL to a metacard based on that metacard's attributes.

Metadata Metacard Transformer

Returns the **Metacard.METADATA** attribute when given a metacard.

RegistryTransformer

Translates a Registry metacard.

Resource Metacard Transformer

Retrieves the resource bytes of a metacard by returning the product associated with the metacard.

Thumbnail Metacard Transformer

Retrieves the thumbnail bytes of a Metacard by returning the **Metacard.THUMBNAIL** attribute value.

XML Metacard Transformer

Translates a metacard into an XML-formatted document.

30.3.1. GeoJSON Metacard Transformer

GeoJSON Metacard Transformer translates a metacard into GeoJSON.

Installing the GeoJSON Metacard Transformer

The GeoJSON Metacard Transformer is not installed by default on a standard installation.

To install:

1. Navigate to the Admin Console
2. Select the **Catalog** application.
3. Select the **Features** tab.
4. Install the **catalog-transformer-json** feature.

Configuring the GeoJSON Metacard Transformer

The GeoJSON Metacard Transformer has no configurable properties.

Using the GeoJSON Metacard Transformer

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 Metacard Transformer

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

Example REST GET Output from the GeoJSON Metacard Transformer

```
{
  "properties": {
    "title": "myTitle",
    "thumbnail": "CA==",
    "resource-uri": "http://example.com",
    "created": "2012-08-31T23:55:19.518+0000",
    "metadata-content-type-version": "myVersion",
    "metadata-content-type": "myType",
    "metadata": "<xml>text</xml>",
    "modified": "2012-08-31T23:55:19.518+0000",
    "metacard-type": "ddf.metacard"
  },
  "type": "Feature",
  "geometry": {
    "type": "LineString",
    "coordinates": [
      [
        [
          30.0,
          10.0
        ],
        [
          [
            10.0,
            30.0
          ],
          [
            40.0,
            40.0
          ]
        ]
      ]
    }
  }
}
```

Table 94. Implementation Details

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

30.3.2. HTML Metocard Transformer

The HTML metocard transformer is responsible for translating a metocard into an HTML-formatted document.

Installing the HTML Metocard Transformer

The HTML Metocard Transformer is installed by default with standard installation in the Search UI application.

Configuring the HTML Metocard Transformer

The HTML Metocard Transformer has no configurable properties.

Using the HTML Metocard Transformer

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

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

HTML Metocard Transformer Example Output

```
html metocard.png
```

Table 95. HTML Metocard Transformer Implementation Details

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

30.3.3. KML Metocard Transformer

The KML Metocard Transformer is responsible for translating a metocard 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.

Installing the KML Metocard Transformer

The KML Metocard Transformer is installed by default with a standard installation in the Spatial Application.

Configuring the KML Metocard Transformer

The KML Metocard Transformer has no configurable properties.

Using the KML Metocard Transformer

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

KML Metocard Transformer 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>&lt;!DOCTYPE html&gt;
      &lt;html&gt;
        &lt;head&gt;
          &lt;meta content="text/html; charset=windows-1252" http-equiv="content-
type"&gt;
            &lt;style media="screen" type="text/css"&gt;
              .label {
                font-weight: bold
              }
              .linkTable {
                width: 100%
              }
              .thumbnailDiv {
                text-align: center
              }
              img {
                max-width: 100px;
                max-height: 100px;
                border-style:none
              }
            &lt;/style&gt;
          &lt;/head&gt;
          &lt;body&gt;
            &lt;div class="thumbnailDiv"&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource"&gt;&lt;img alt="Thumbnail" src="data:image/jpeg; charset=utf-8; base64, CA=="&gt;&lt;/a&gt;&lt;/div&gt;
          &lt;/table&gt;
        &lt;/body&gt;
      &lt;/html&gt;
    &lt;/description>
  </Placemark>
</kml>
```

```

<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/0103c77e66d9428d8f48fab939da528e?transform=html">View Details...</a></td>
        <td><a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?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>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;&lt;td
width="400"&gt;$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</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>&lt;h3&gt;&lt;b&gt;#[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr
&gt;&lt;td width="400"&gt;#[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</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>
      <coordinates>100.8,0.2
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>

```

Table 96. KML Metocard Transformer Implementation Details

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

30.3.4. KML Style Mapper

The KML Style Mapper provides the ability for the [KmlTransformer](#) to map a KML Style URL to a metocard based on that metocard'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 metocard 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](#).

Installing the KML Style Mapper

The KML Style Mapper is installed by default with a standard installation in the [Spatial Application](#) in the [spatial-kml-transformer](#) feature.

Configuring the KML Style Mapper

The properties below describe how to configure a style mapping. The configuration name is [Spatial KML Style Map Entry](#).

Table 97. Spatial KML Style Map Entry

Name	Id	Type	Description	Default Value	Required
Attribute Name	<code>attributeName</code>	String	The name of the Metocard Attribute to match against. e.g. title, metadata-content-type, etc	null	true
Attribute Value	<code>attributeValue</code>	String	The value of the Metocard Attribute.	null	true
Style URL	<code>styleUrl</code>	String	The full qualified URL to the KML Style. e.g. http://example.com/styles#myStyle	null	true

KML Style Mapper 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>&lt;!DOCTYPE html&gt;
&lt;html&gt;
  &lt;head&gt;
    &lt;meta content="text/html; charset=windows-1252" http-equiv="content-type"&gt;
    &lt;style media="screen" type="text/css"&gt;
      .label {
        font-weight: bold
      }
      .linkTable {
        width: 100%
      }
      .thumbnailDiv {
        text-align: center
      }
      img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    &lt;/style&gt;
  &lt;/head&gt;
  &lt;body&gt;
    &lt;div class="thumbnailDiv"&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource"&gt;&lt;img alt="Thumbnail" src="data:image/jpeg;charset=utf-8;base64, CA=="&gt;&lt;/a&gt;&lt;/div&gt;
    &lt;table&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Source:&lt;/td&gt;
        &lt;td&gt;ddf.distribution&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Created:&lt;/td&gt;
        &lt;td&gt;Wed Oct 30 09:46:29 MDT 2013&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Effective:&lt;/td&gt;
        &lt;td&gt;2014-01-07T14:58:16-0700&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
    &lt;table class="linkTable"&gt;
      &lt;tr&gt;
        &lt;td&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=html"&gt;View Details...&lt;/a&gt;&lt;/td&gt;
        &lt;td&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource"&gt;Download...&lt;/a&gt;&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
  &lt;/body&gt;
&lt;/html&gt;
</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>
      <coordinates>100.8,0.2 100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2
100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
    </LinearRing>
  </outerBoundaryIs>
    </Polygon>
  </MultiGeometry>
</MultiGeometry>
</Placemark>
</kml>

```

Table 98. KML Style Mapper Implementation Details

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

30.3.5. Metadata Metocard Transformer

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

Installing the Metadata Metocard Transformer

The Metadata Metocard Transformer is installed by default with a standard installation in the Catalog application.

This transformer's feature, `catalog-transformer-metadata`, can be uninstalled.

Configuring the Metadata Metacard Transformer

The Metadata Metacard Transformer has no configurable properties.

Using the Metadata Metacard Transformer

The Metadata Metacard Transformer can be used programmatically by requesting a metacard transformer 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 Metacard Transformer

```
http://localhost:8181/services/catalog/0123456789abcdef0123456789abcdef?transform=meta  
data
```

Table 99. Metadata Metacard Transformer Exported Services

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

30.3.6. Resource Metacard Transformer

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

Installing the Resource Metacard Transformer

The Resource Metacard Transformer is installed by default with a standard installation in the Catalog application as the feature `catalog-transformer-resource`.

Configuring the Resource Metacard Transformer

The Resource Metacard Transformer has no configurable properties.

Using the Resource Metacard Transformer

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

Sample Resource Metacard Transformer Blueprint Reference Snippet

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

Table 100. Resource Metacard Transformer Implementation Details

Service Property	Value	id
resource	shortname	resource
mime-type	application/octet-stream	title

30.3.7. Thumbnail Metacard Transformer

The Thumbnail Metacard Transformer retrieves the thumbnail bytes of a Metacard by returning the [Metacard.THUMBNAIL](#) attribute value.

Installing the Thumbnail Metacard Transformer

This transformer is installed by default with a standard installation in the Catalog application.

Configuring the Thumbnail Metacard Transformer

The Thumbnail Metacard Transformer has no configurable properties.

Using the Thumbnail Metacard Transformer

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](#).

Table 101. Thumbnail Metacard Transformer Implementation Details

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

30.3.8. XML Metacard Transformer

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.

Installing the XML Metacard Transformer

This transformer comes installed by default with a standard installation in the Catalog application.

To install or uninstall manually, use the `catalog-transformer-xml` feature.

Configuring the XML Metocard Transformer

the XML Metocard Transformer has no configurable properties.

Using the XML Metocard Transformer

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

XML Metocard Transformer URL

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

Table 102. Metocard to XML Mappings

Metocard Variables	XML Element
<code>id</code>	<code>metocard/@gml:id</code>
<code>metocardType</code>	<code>metocard/type</code>
<code>sourceId</code>	<code>metocard/source</code>
all other attributes	<code>metocard/<AttributeType>[name='<AttributeName>']/value</code> For instance, the value for the metocard attribute named "title" would be found at <code>metocard/string[@name='title']/value</code>

XML Adapted Attributes (AttributeTypes)

- boolean
- base64Binary
- dateTime
- double
- float
- geometry
- int
- long
- object
- short
- string
- stringxml

30.4. Developing Metocard Transformers

In general, a `MetocardTransformer` is used to transform a `Metocard` into some desired format useful to

the end user or as input to another process. Programmatically, a `MetacardTransformer` transforms a `Metacard` into a `BinaryContent` instance, which contains the translated `Metacard` into the desired final format. Metacard transformers can be used through the Catalog Framework `transform` convenience method or requested from the OSGi Service Registry by endpoints or other bundles.

Creating a New Metacard Transformer

Existing input transformers are written as Java classes, and these steps walk through the steps to create a custom input transformer.

1. Create a new Java class that implements `ddf.catalog.transform.MetacardTransformer`.

```
public class SampleMetacardTransformer implements ddf.catalog.transform.MetacardTransformer
```

2. Implement the `transform` method.

```
public BinaryContent transform(Metacard metocard, Map<String, Serializable> arguments)
throws CatalogTransformerException
```

- a. `transform` must return a `Metacard` or throw an exception. It cannot return null.

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

`Import-Package: ddf.catalog,ddf.catalog.transform`

4. Create an OSGi descriptor file to communicate with the OSGi Service registry (described in the Working with OSGi section). Export the service to the OSGi registry and declare service properties.

Metacard Transformer Blueprint Descriptor Example

```
...
<service ref="[[SampleMetacardTransformer]]" interface=
"ddf.catalog.transform.MetacardTransformer">
    <service-properties>
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Metacard Transformer]]" />
        <entry key="description" value="[[A new transformer for metacards.]]" />
    </service-properties>
</service>
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Table 103. Metacard Transformer Blueprint Service Properties / Variable Descriptions

Key	Description of Value	Example
<code>shortname</code>	(Required) An abbreviation for the return type of the <code>BinaryContent</code> being sent to the user.	atom

Key	Description of Value	Example
<code>title</code>	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	Atom Entry Transformer Service
<code>description</code>	(Optional) A short, human-readable description that describes the functionality of the service and the output.	This service converts a single metocard xml document to an atom entry element.

30.5. Query Response Transformers

The following query response transformers are available with a standard installation of DDF.

Atom Query Response Transformer

Transforms a query response into an Atom 1.0 feed.

Query Response Transformer

Transforms a query response into a CSW formatted document.

GeoJSON Query Response Transformer

Translates a query response into a GeoJSON formatted document.

KML Query Response Transformer

Translates a query response into a KML-formatted document.

Query Response Transformer Consumer

`description`

SearchUI

The SearchUI is a `QueryResponseTransformer` that not only provides results in html format but also provides a convenient, simple querying user interface.

XML Query Response Transformer

Translates a query response into an XML-formatted document.

30.5.1. Atom Query Response Transformer

The Atom Query Response Transformer transforms a query response into an Atom 1.0 feed. The Atom transformer maps a `QueryResponse` object as described in the Query Result Mapping.

Installing the Atom Query Response Transformer

The Atom Query Response Transformer is installed by default with a standard installation.

Configuring the Atom Query Response Transformer

The Atom Query Response Transformer has no configurable properties.

Using the Atom Query Response Transformer

Use this transformer when Atom is the preferred medium of communicating information, such as for feed readers or federation. An integrator could use this with an endpoint to transform query responses into an Atom feed.

For example, clients can use the [OpenSearch Endpoint](#). The client can query with the format option set to the shortname, atom.

Sample OpenSearch Query with Atom Specified as Return Format

```
http://localhost:8181/services/catalog/query?q=ddf?format=atom
```

Developers could use this transformer to programmatically transform `QueryResponse` objects on the fly.

Sample Atom Feed from `QueryResponse` object

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:os="http://a9.com/-spec/opensearch/1.1/">
  <title type="text">Query Response</title>
  <updated>2013-01-31T23:22:37.298Z</updated>
  <id>urn:uuid:a27352c9-f935-45f0-9b8c-5803095164bb</id>
  <link href="#" rel="self" />
  <author>
    <name>Lockheed Martin</name>
  </author>
  <generator version="2.1.0.20130129-1341">ddf123</generator>
  <os:totalResults>1</os:totalResults>
  <os:itemsPerPage>10</os:itemsPerPage>
  <os:startIndex>1</os:startIndex>
  <entry xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/">
    <id>urn:catalog:id:ee7a161e01754b9db1872bfe39d1ea09</id>
    <title type="text">F-15 lands in Libya; Crew Picked Up</title>
    <updated>2013-01-31T23:22:31.648Z</updated>
    <published>2013-01-31T23:22:31.648Z</published>
    <link href="http://123.45.67.123:8181/services/catalog/ddf123/ee7a161e01754b9db1872bfe39d1ea09" rel="alternate" title="View Complete Metocard" />
    <category term="Resource" />
    <georss:where xmlns:gml="http://www.opengis.net/gml">
      <gml:Point>
        <gml:pos>32.8751900768792 13.1874561309814</gml:pos>
```

```

</gml:Point>
</georss:where>
<content type="application/xml">
    <ns3:metacard xmlns:ns3="urn:catalog:metacard" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ns1="http://www.opengis.net/gml"
    xmlns:ns4="http://www.w3.org/2001/SMIL20/" xmlns:ns5=
"http://www.w3.org/2001/SMIL20/Language" ns1:id="4535c53fc8bc4404a1d32a5ce7a29585">
        <ns3:type>ddf.metacard</ns3:type>
        <ns3:source>ddf.distribution</ns3:source>
        <ns3:geometry name="location">
            <ns3:value>
                <ns1:Point>
                    <ns1:pos>32.8751900768792 13.1874561309814</ns1:pos>
                </ns1:Point>
            </ns3:value>
        </ns3:geometry>
        <ns3:dateTime name="created">
            <ns3:value>2013-01-31T16:22:31.648-07:00</ns3:value>
        </ns3:dateTime>
        <ns3:dateTime name="modified">
            <ns3:value>2013-01-31T16:22:31.648-07:00</ns3:value>
        </ns3:dateTime>
        <ns3:stringxml name="metadata">
            <ns3:value>
                <ns6:xml xmlns:ns6="urn:sample:namespace" xmlns=
"urn:sample:namespace">Example description.</ns6:xml>
            </ns3:value>
        </ns3:stringxml>
        <ns3:string name="metadata-content-type-version">
            <ns3:value>myVersion</ns3:value>
        </ns3:string>
        <ns3:string name="metadata-content-type">
            <ns3:value>myType</ns3:value>
        </ns3:string>
        <ns3:string name="title">
            <ns3:value>Example title</ns3:value>
        </ns3:string>
    </ns3:metacard>
</content>
</entry>
</feed>

```

Table 104. Atom Query Response Transformer Result Mapping

XPath to Atom XML	Value
/feed/title	"Query Response"
/feed/updated	ISO 8601 dateTime of when the feed was generated
/feed/id	Generated (UUID URN

XPath to Atom XML	Value
/feed/author/name	Platform Global Configuration organization
/feed/generator	Platform Global Configuration site name
/feed/generator/@version	Platform Global Configuration version
/feed/os:totalResults	SourceResponse Number of Hits
/feed/os:itemsPerPage	Request's Page Size
/feed/os:startIndex	Request's Start Index
/feed/entry/fs:resultSource/@fs:souceId	Source Id from which the Result came. <code>Metacard.getSourceId()</code>
/feed/entry/relevance:score	Result's relevance score if applicable. <code>Result.getRelevanceScore()</code>
/feed/entry/id	urn:catalog:id:<Metacard.ID>
/feed/entry/title	Metacard.TITLE
/feed/entry/updated	ISO 8601 dateTime of Metacard.MODIFIED
/feed/entry/published	ISO 8601 dateTime of Metacard.CREATED
/feed/entry/link[@rel='related']	URL to retrieve underlying resource (if applicable and link is available)
/feed/entry/link[@rel='alternate']	Link to alternate view of the Metacard (if a link is available)
/feed/entry/category	Metacard.CONTENT_TYPE
/feed/entry//georss:where	GeoRSS GML of every Metacard attribute with format <code>AttributeFormat.GEOMETRY</code>
/feed/entry/content	Metacard XML generated by <code>DDF.catalog.transform.MetacardTransformer</code> with <code>shortname=xml</code> . If no transformer found, <code>/feed/entry/content/@type</code> will be text and Metacard.ID is displayed <code><content type="text">4e1f38d1913b4e93ac622e6c1b258f89</content></code>

30.5.2. CSW Query Response Transformer

The CSW Query Response Transformer transforms a query response into a ([CSW-formatted](#) document.

Installing the CSW Query Response Transformer

The CSW Query Response Transformer is installed by default with a standard installation in the Spatial application.

Configuring the CSW Query Response Transformer

The CSW Query Response Transformer has no configurable properties.

30.5.3. GeoJSON Query Response Transformer

The GeoJSON Query Response Transformer translates a query response into a GeoJSON-formatted document.

Installing the GeoJSON Query Response Transformer

The GeoJSON Query Response Transformer is installed by default with a standard installation in the Catalog application.

Configuring the GeoJSON Query Response Transformer

The GeoJSON Query Response Transformer has no configurable properties.

30.5.4. 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 metocard 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.

Installing the KML Query Response Transformer

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

Configuring the KML Query Response Transformer

The KML Query Response Transformer has no configurable properties.

Using the KML Query Response Transformer

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

KML Query Response Transformer URL

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

KML Query Response Transformer 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>&lt;h3&gt;&lt;b&gt;[$name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr
            &gt;&lt;td width="400"&gt;[$description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</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>&lt;h3&gt;&lt;b&gt;[$name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr
            &gt;&lt;td width="400"&gt;[$description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</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>&lt;!DOCTYPE html&gt;
&lt;html&gt;
    &lt;head&gt;
        &lt;meta content="text/html; charset=windows-1252" http-equiv="content-type"&gt;
        &lt;style media="screen" type="text/css"&gt;

```

```

.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/0103c77e66d9428d8f48fab939da528e?transform=resource"></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/0103c77e66d9428d8f48fab939da528e?transform=html">View Details...</a></td>
<td><a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?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>

```

Table 105. KML Query Response Implementation Details

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

30.5.5. Query Response Transformer Consumer

The Query Response Transformer Consumer is responsible for translating a query response into a Catalog Metocard.

Installing the Query Response Transformer Consumer

The Query Response Transformer Consumer is installed by default with a standard installation in the Catalog application.

Configuring the Query Response Transformer Consumer

The Query Response Transformer Consumer has no configurable properties.

30.5.6. Simple SearchUI

The Simple SearchUI is a [QueryResponseTransformer](#) that not only provides results in html format but also provides a convenient, simple querying user interface. It is primarily used as a test tool and verification of configuration. The left pane of the Simple SearchUI contains basic fields to query the Catalog and other Sources. The right pane consists of the results returned from the query.

Installing Simple SearchUI

This transformer is installed by default with a standard installation in the Catalog application as the feature, [catalog-transformer-ui](#).

Configuring Simple SearchUI

From the Admin Console, the Simple SearchUI can be configured under the [Catalog HTML Query Response Transformer](#).

Table 106. Simple Search UI

Name	Id	Type	Description	Default Value	Required
Header	header	String	Specifies the header text to be rendered on the generated Query Page		true
Footer	footer	String	Specifies the footer text to be rendered on the generated Query Page		true
Text Color	color	String	Specifies the Text Color of the Header and Footer. Use html css colors or #rrggbb.		true
Background Color	background	String	Specifies the Background Color of the Header and Footer. Use html css colors or #rrggbb.	null	true

Using Simple SearchUI

In order to obtain the Simple SearchUI, a user must use the transformer with an endpoint that queries the Catalog such as the OpenSearch Endpoint. If a distribution is running locally, <https://localhost:8993/search/simple> should bring up the Simple Search UI. After the page has loaded, enter the desired search criteria in the appropriate fields. Then click the "Search" button in order to execute the search on the Catalog.

The "Clear" button will reset the query criteria specified.

Table 107. Simple SearchUI Query Response Result Mapping

Simple SearchUI Column Title	Catalog Result	Notes
Title	Metocard.TITLE	The title may be hyperlinked to view the full Metocard

Simple SearchUI Column Title	Catalog Result	Notes
Source	<code>Metacard.getSourceId()</code>	Source where the Metacard was discovered
Location	<code>Metacard.LOCATION</code>	Geographical location of the Metacard
Time	<code>Metacard.CREATED or Metacard.EFFECTIVE</code>	Time received/created
Thumbnail	<code>Metacard.THUMBNAIL</code>	No column shown if no results have thumbnail
Resource	<code>Metacard.RESOURCE_URI</code>	No column shown if no results have a resource

Search Criteria

The Simple SearchUI allows for querying a Catalog in the following methods:

Keyword Search

searching with keywords using the grammar of the underlying endpoint/Catalog.

Temporal Search

searching based on relative or absolute time.

Spatial search

searching spatially with a Point-Radius or Bounding Box.

Content Type Search

searching for specific `Metacard.CONTENT_TYPE` values

Usage Limitations of the Simple SearchUI

If the Simple SearchUI results do not provide usable links on the metocard results, verify that a valid host has been entered in the Platform Global Configuration.

30.5.7. XML Query Response Transformer

The XML Query Response Transformer is responsible for translating a query response into an XML-formatted document. The metacards element that is generated is an extension of `gml:AbstractFeatureCollectionType`, which makes the output of this transformer [GML 3.1.1](#) compatible.

Installing the XML Query Response Transformer

This transformer is installed by default with a standard installation in the Catalog application. To uninstall, uninstall the `catalog-transformer-xml` feature.

Configuring the XML Query Response Transformer

The XML Query Response Transformer has no configurable properties.

Using the XML Query Response Transformer

Using the OpenSearch Endpoint, for example, query with the format option set to the XML shortname `xml`.

XML Query Response Transformer Query Example

```
http://localhost:8181/services/catalog/query?q=input?format=xml
```

XML Query Response Transformer Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:metacards xmlns:ns1="http://www.opengis.net/gml" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ns3="urn:catalog:metacard" xmlns:ns4=
"http://www.w3.org/2001/SMIL20/" xmlns:ns5="http://www.w3.org/2001/SMIL20/Language">
  <ns3:metocard ns1:id="000ba4dd7d974e258845a84966d766eb">
    <ns3:type>ddf.metacard</ns3:type>
    <ns3:source>southwestCatalog1</ns3:source>
    <ns3:dateTime name="created">
      <ns3:value>2013-04-10T15:30:05.702-07:00</ns3:value>
    </ns3:dateTime>
    <ns3:string name="title">
      <ns3:value>Input 1</ns3:value>
    </ns3:string>
  </ns3:metocard>
  <ns3:metocard ns1:id="00c0eb4ba9b74f8b988ef7060e18a6a7">
    <ns3:type>ddf.metacard</ns3:type>
    <ns3:source>southwestCatalog1</ns3:source>
    <ns3:dateTime name="created">
      <ns3:value>2013-04-10T15:30:05.702-07:00</ns3:value>
    </ns3:dateTime>
    <ns3:string name="title">
      <ns3:value>Input 2</ns3:value>
    </ns3:string>
  </ns3:metocard>
</ns3:metacards>
```

Table 108. XML Query Response Transformer Implementation Details

Registered Interface	Service Property	Value
ddf.catalog.transform.QueryResponseTransformer	shortname	xml
	description	Transforms query results into xml
	title	View as XML...

30.6. 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.

Included Mime Type Mappers

[DDF Mime Type Mapper](#)

Resolves file extensions to mime types, and vice versa.

30.6.1. 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 the DDF Mime Type Mapper

The DDF Mime Type Mapper is installed by default in a standard installation.

Configuring DDF Mime Type Mapper

The DDF Mime Type Mapper has no configurable properties.

30.7. 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).

Included Mime Type Resolvers

Tika Mime Type Resolver

provides support for resolving over 1300 mime types.

Custom Mime Type Resolver

used when mime types need to be added that are not supported by default.

30.7.1. Tika Mime Type Resolver

The `TikaMimeTypeResolver` is a `MimeTypeResolver` that is implemented using the <https://tika.apache.org>[Apache Tika] open source product.

Using the Apache Tika content analysis toolkit, the `TikaMimeTypeResolver` provides support for resolving over 1300 mime types, but not all mime types yield the same quality metadata.

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`.

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

Installing the Tika Mime Type Resolver

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

This feature is installed by default.

Configuring the Tika Mime Type Resolver

The Tika Mime Type Resolver has no configurable properties.

Table 109. Tika Mime Type Resolver Exported Services

Registered Interface	Service Property	Value
<code>ddf.mime.MimeTypeResolver</code>		<code>tika-mimetypes.xml</code>

30.7.2. Custom Mime Type Resolver

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

Table 110. Custom Mime Type Resolver Default Supported Mime Types

File Extension	Mime Type
<code>nitf</code>	<code>image/nitf</code>
<code>ntf</code>	<code>image/ntf</code>

File Extension	Mime Type
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.

Installing the Custom Mime Type Resolver

One Custom Mime Type Resolver is configured and installed 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 Custom Mime Type Resolver

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

- Navigate to the Admin Console.
- Select the **Platform** application.
- Select **Configuration**.
- Select **MIME Custom Types**.

Managed Service Factory PID

- `Ddf_Custom_Mime_Type_Resolver`

Table 111. MIME Custom Types

Name	Id	Type	Description	Default Value	Required
Resolver Name	<code>name</code>	String	null	DDF Custom Resolver	false
Priority	<code>priority</code>	Integer	null	10	true
File Extensions to Mime Types	<code>customMimeTypes</code>	String	List of key/value pairs where key is the file extension and value is the mime type, e.g., nitf=image/nitf	null	true

Table 112. Custom Mime Type Resolver Imported Services

Registered Interface	Availability	Multiple
<code>ddf.catalog.transform.InputTransformer</code>	optional	true
<code>ddf.catalog.transform.QueryResponseTransformer</code>	optional	true

Registered Interface	Availability	Multiple
<code>ddf.mime.MimeTypeResolver</code>	optional	true

Table 113. Custom Mime Type Resolver Exported Services

Registered Interface	Service Property	Value
<code>ddf.mime.MimeTypeToTransformerMapper</code>		
<code>ddf.mime.MimeTypeMapper</code>		

30.8. Developing Query Response Transformers

A `QueryResponseTransformer` is used to transform a List of Results from a `SourceResponse`. Query Response Transformers can be used through the Catalog transform convenience method or requested from the OSGi Service Registry by endpoints or other bundles.

1. Create a new Java class that implements `ddf.catalog.transform.QueryResponseTransformer`.

```
public class SampleResponseTransformer implements
ddf.catalog.transform.QueryResponseTransformer
```

2. Implement the `transform` method.

```
public BinaryContent transform(SourceResponse upstreamResponse, Map<String, Serializable>
arguments) throws CatalogTransformerException
```

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog, ddf.catalog.transform
```

4. Create an OSGi descriptor file to communicate with the OSGi Service Registry (described in [Working with OSGi](#)). Export the service to the OSGi registry and declare service properties.

Query Response Transformer Blueprint Descriptor Example

```
...
<service ref="[[SampleResponseTransformer]]" interface=
"ddf.catalog.transform.QueryResponseTransformer">
    <service-properties>
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Response Transformer]]" />
        <entry key="description" value="[[A new transformer for response queues.]]"
    />
    </service-properties>
</service>
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Table 114. Query Response Transformer Blueprint Service Properties / Variable Descriptions

Key	Description of Value	Example
<code>shortname</code>	An abbreviation for the return type of the BinaryContent being sent to the user.	atom
<code>title</code>	A user-readable title that describes (in greater detail than the shortname) the service.	Atom Entry Transformer Service
<code>description</code>	A short, human-readable description that describes the functionality of the service and the output.	<i>This service converts a single metocard xml document to an atom entry element.</i>

31. Plugins

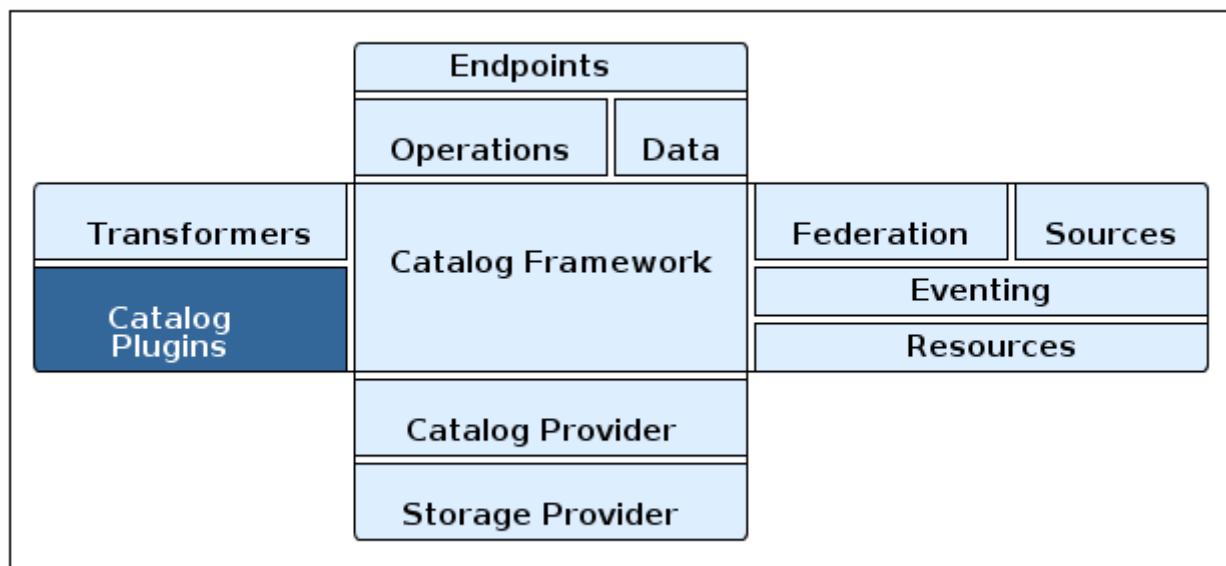


Figure 30. Catalog Architecture: Plugins

Plugins are additional tools to use to add additional business logic at certain points, depending on the type of plugin.

The Catalog Framework calls Catalog Plugins to process requests and responses as they enter and leave the Framework.

31.1. Types of Plugins

Plugins can be designed to run before or after certain processes. They are often used for validation, optimization, or logging. Many plugins are designed to be called at more than one time. See [Plugin Compatibility](#).

"Pre-" Plugins

These plugins are executed before an action is taken.

Pre-Ingest Plugins

Perform any changes to a resource prior to ingesting it.

Pre-Query Plugins

Perform any changes to query before executing.

Pre-Resource Plugins

Perform any changes to a resource associated with a metocard prior to download.

Pre-Subscription Plugins

Perform any changes before creating a subscription.

Pre-Delivery Plugins

Perform any changes before delivered a subscribed event.

Pre>Create Storage Plugins

Perform any changes before creating a resource.

Pre-Update Storage Plugins

Perform any changes before updating a resource.

"Post-" Plugins

These plugins execute after an action is taken.

Post-Ingest Plugins

Perform actions after ingest is completed.

Post-Query Plugins

Perform any changes to response after query completes.

Post-Resource Plugins

Perform any changes to a resource after download.

Post-Create Storage Plugins

Perform any changes after creating a resource.

Post-Update Storage Plugins

Perform any changes after updating a resource.

Other Plugins

Policy Plugins

used to build policy information for requests.

Access Plugins

allows or denies access to the Catalog operation or response.

31.1.1. Plugin Invocation

Plugins are called in a specific order during different operations. [Custom Plugins](#) can be added to the chain for special use cases.

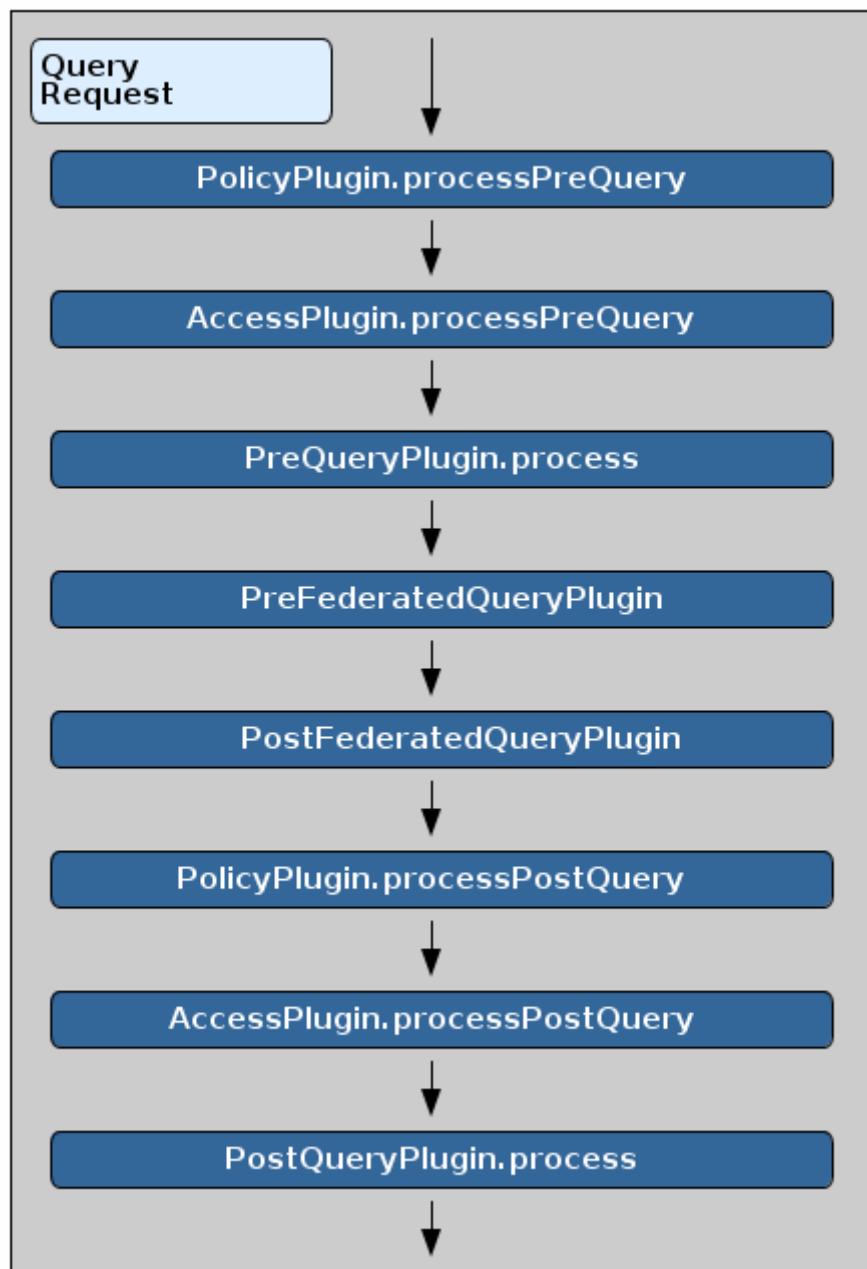


Figure 31. Query Request Plugin Call Order

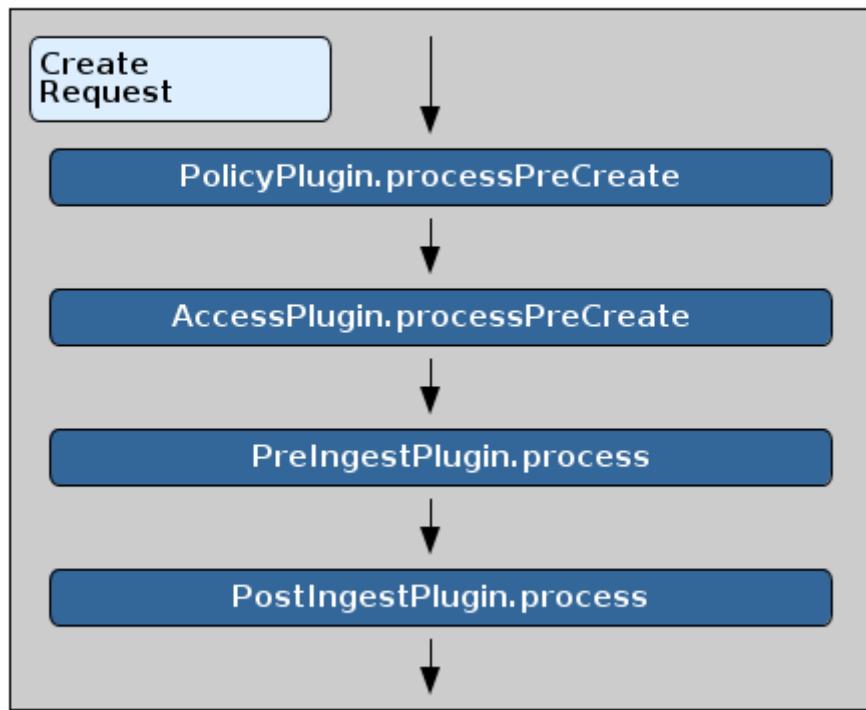


Figure 32. Create Request Plugin Call Order

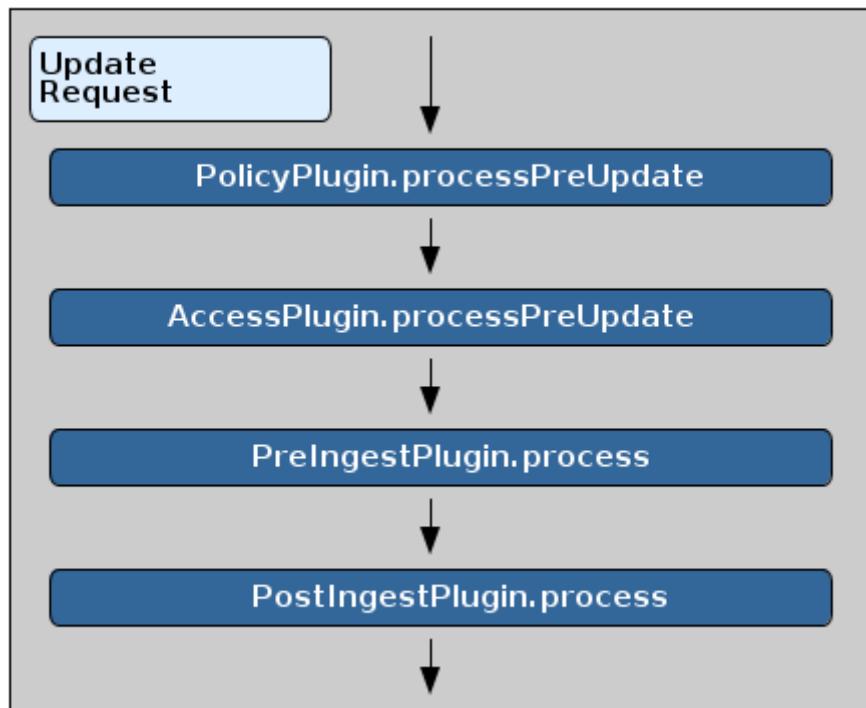


Figure 33. Update Request Plugin Call Order

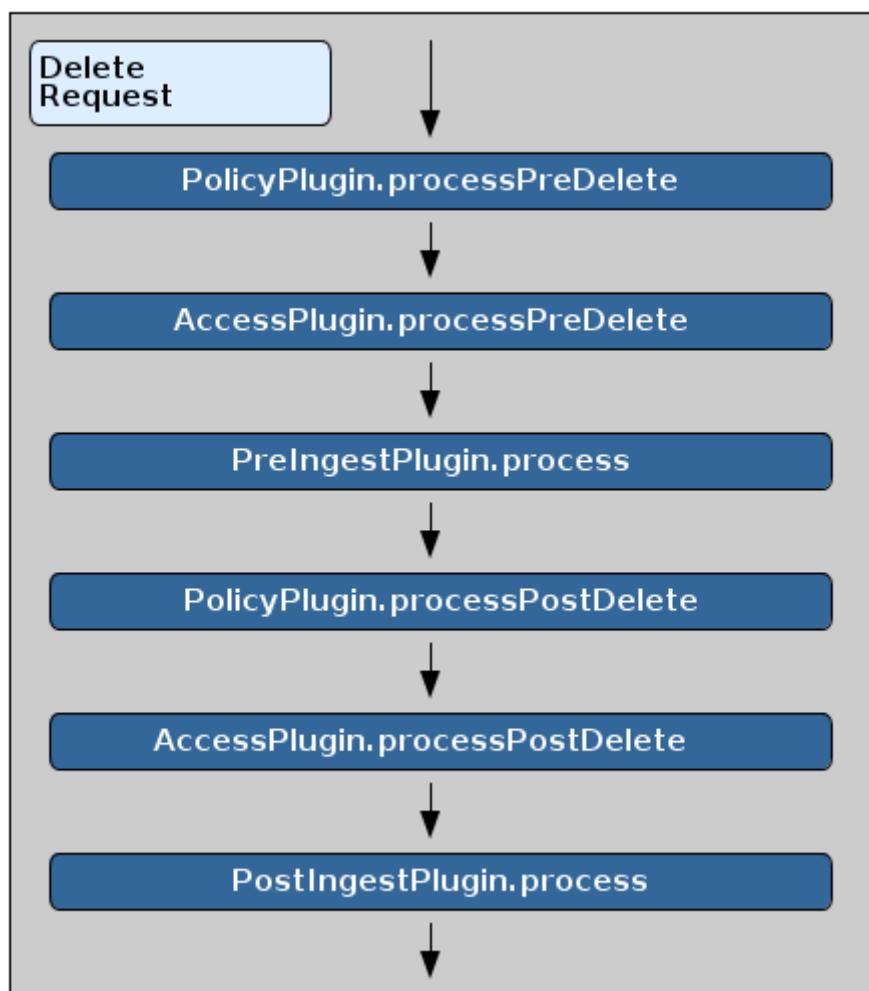


Figure 34. Delete Request Plugin Call Order

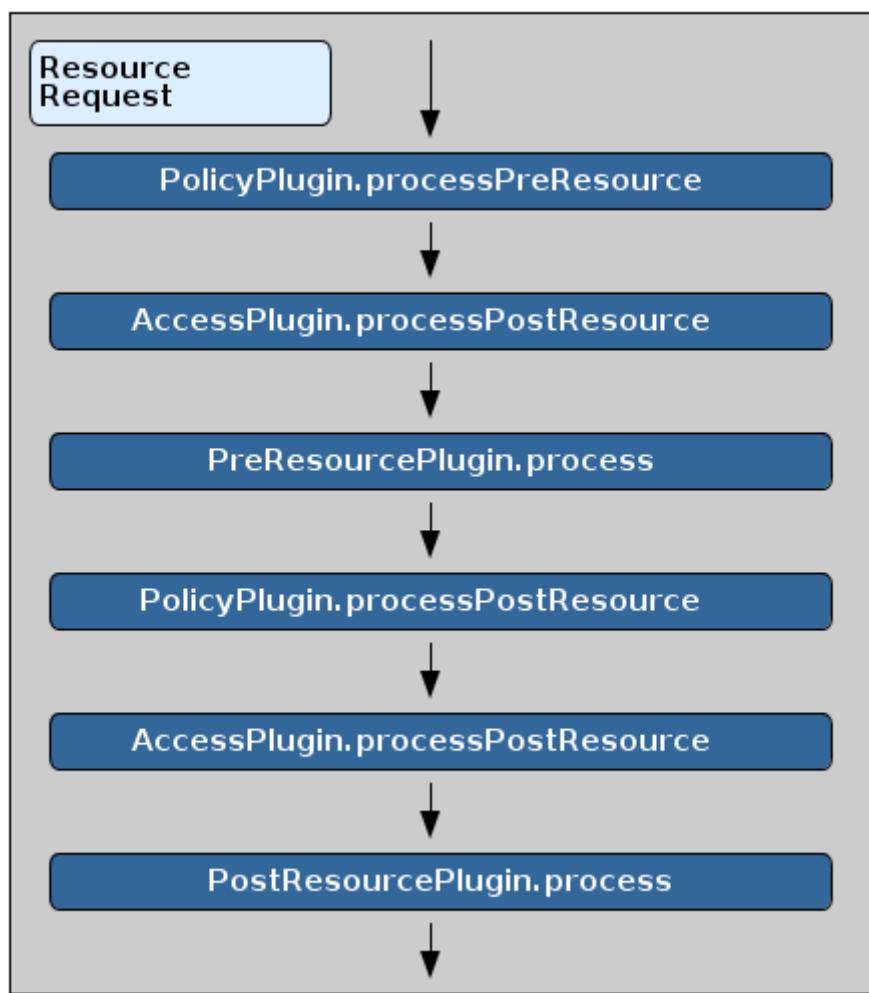


Figure 35. Resource Request Plugin Call Order

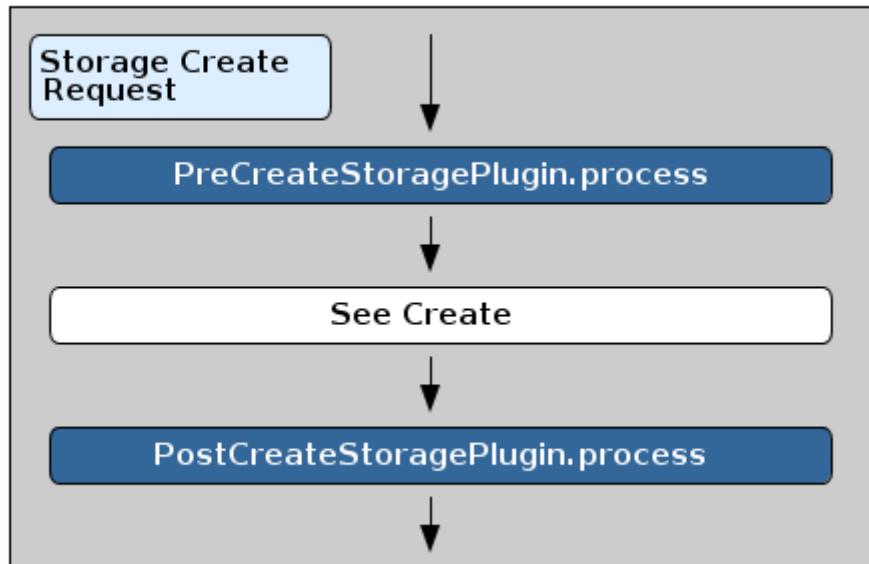


Figure 36. Storage Create Request Plugin Call Order

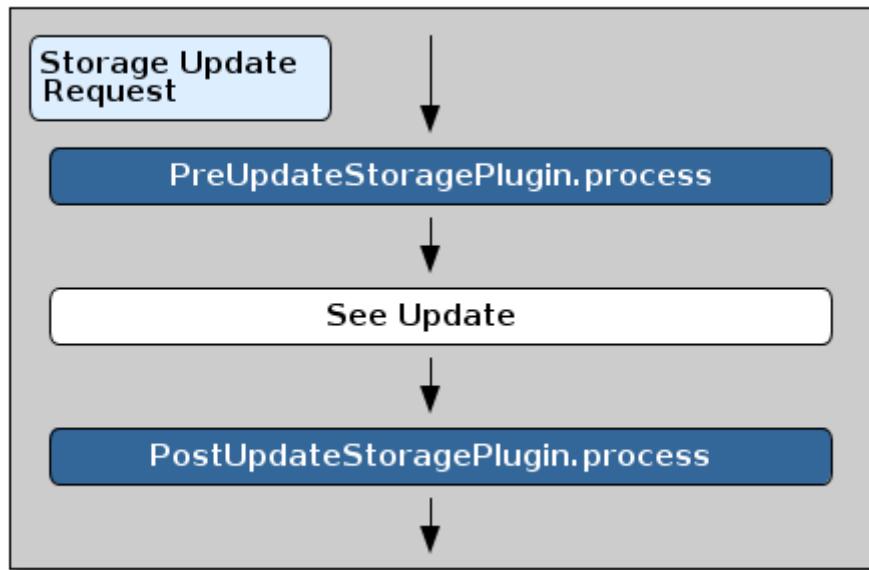


Figure 37. Storage Update Request Plugin Call Order

Table 115. DDF "Pre" Plugin Compatibility

Plugin	Pre-Ingest	Pre-Query	Pre-Resource	Pre-Subscription	Pre-Delivery	Pre-Create Storage	Pre-Update Storage
Catalog Backup Plugin							
Caching Federation Strategy							
Catalog Metrics Plugin		x					
Checksum Plugin						x	x
Dummy Pre-Ingest Plugin	x						
Event Processor							
Expiration Date Pre-Ingest Plugin	x						

Plugin	Pre-Ingest	Pre-Query	Pre-Resource	Pre-Subscription	Pre-Delivery	Pre-Create Storage	Pre-Update Storage
Fanout Event Processor							
Filter Plugin							
Identification Plugin	x						
Jpeg2000 Thumbnail Converter							
Metocard Groomer	x						
Metocard Resource Size Plugin							
Metocard Resource Status							
Metocard Validity Marker	x						
Operation Plugin							
Preview Storage						x	x
Resource Usage Plugin			x				
REST Replicator Plugin							
Security Logging Plugin	x	x	x			x	x
Security Plugin							
Video Thumbnail							

Table 116. DDF "Post", Policy, and Access Plugin Compatibility

Plugin	Post-Ingest	Post-Query	Post-Resource	Post-Create Storage	Post-Update Storage	Policy	Access
Catalog Backup Plugin	x						
Caching Federation Strategy	x						
Catalog Metrics Plugin	x	x	x				
Catalog Policy Plugin						x	
Resource URI Policy Plugin						x	
Checksum Plugin							
Event Processor	x						
Fanout Event Processor	x						
Filter Plugin							x
Filter Post Query Plugin						x	
Historian Policy Plugin						x	
Identification Plugin	x						
Jpeg2000 Thumbnail Converter		x					
Metocard Attribute Security Policy Plugin						x	

Plugin	Post-Ingest	Post-Query	Post-Resource	Post-Create Storage	Post-Update Storage	Policy	Access
Metocard Resource Size Plugin		x					
Metocard Resource Status		x					
Metocard Validity Filter Plugin						x	
Operation Plugin							x
Registry Policy Plugin						x	
Resource Usage Plugin		x	x				
REST Replicator Plugin	x						
Security Logging Plugin	x	x	x	x	x		
Security Plugin							x
Video Thumbnail				x	x		
XML Attribute Security Policy Plugin							x

31.2. Pre-Ingest Plugins

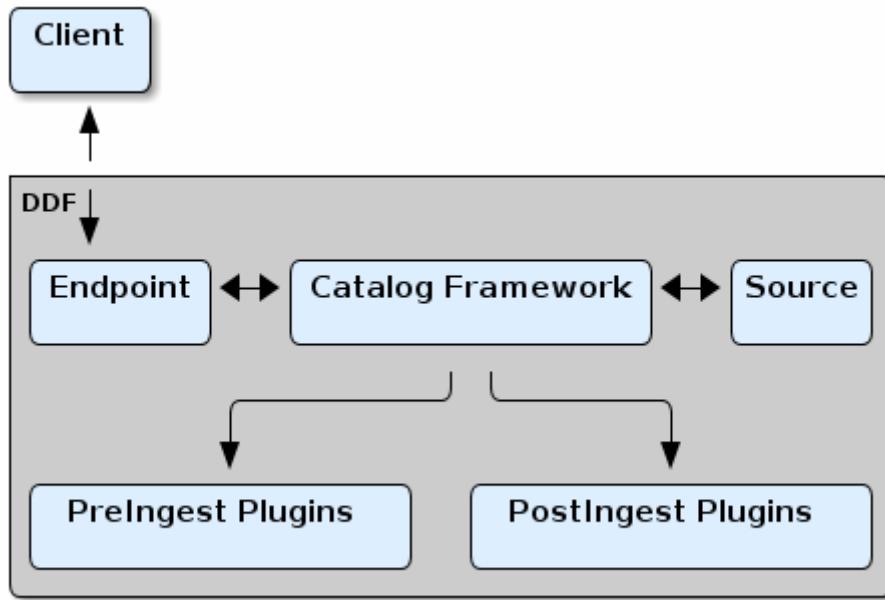


Figure 38. Ingest Plugin Flow

Pre-Ingest plugins are invoked before an ingest operation is sent to the catalog. They are not run on a query. This is an opportunity to take any action on the ingest request, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

Included Pre-Ingest Plugins

Metocard Groomer

The Metocard Groomer Pre-Ingest plugin makes modifications to `CreateRequest` and `UpdateRequest` metacards.

Metocard Validity Marker

The Metocard Validity Marker Pre-Ingest plugin modifies the metacards contained in `CreateRequest`s and `UpdateRequest`s.

Expiration Date Pre-Ingest Plugin

The Expiration Date plugin updates metocard expiration dates.

Identification Plugin

The Identification Plugin assigns IDs to a metocard as it is added to the catalog.

Security Logging Plugin

The Security Logging Plugin logs operations to the security log.

31.2.1. Metacard Groomer

The Metacard Groomer Pre-Ingest plugin makes modifications to [CreateRequest](#) and [UpdateRequest](#) metacards.

Use this pre-ingest plugin as a convenience to apply basic rules for your metacards.

This plugin makes the following modifications when metacards are in a [CreateRequest](#):

- Overwrites the [Metacard.ID](#) field with a generated, unique, 32 character hexadecimal value if missing or if the resource URI is not a catalog resource URI.
- Sets [Metacard.CREATED](#) to the current time stamp if not already set.
- Sets [Metacard.MODIFIED](#) to the current time stamp if not already set.
- Sets [Metacard.TAGS](#) to the default [Metacard.DEFAULT_TAG](#) if no tags are present.
- Sets [Core.METACARD_CREATED](#) to the current time stamp if not present.
- Sets [Core.METACARD_MODIFIED](#) to the current time stamp.

In an [UpdateRequest](#), the same operations are performed as a [CreateRequest](#), except:

- If no value is provided for [Metacard.ID](#) in the new metocard, it will be set using the [UpdateRequest](#) ID if applicable.

Installing the Metacard Groomer

The Metacard Groomer is installed by default with a standard installation in the Catalog application.

Configuring the Metacard Groomer

The Metacard Groomer has no configurable properties.

31.2.2. Metacard Validity Marker

The Metacard Validity Marker Pre-Ingest plugin modifies the metacards contained in [CreateRequests](#) and [UpdateRequests](#).

The plugin runs each metocard in the [CreateRequest](#) and [UpdateRequest](#) against each registered [MetacardValidator](#) service.

NOTE

This plugin can make it seem like ingested products are not successfully ingested. If after ingest, the ingest did not fail and there are no errors in the ingest log, but the expected results do not show up after a query, verify either that the ingested data is valid or that the [Catalog Federation Strategy](#) is configured to show warnings and/or errors.

Installing Metacard Validity Marker

This plugin is installed by default with a standard installation in the {ddf-catalog} application.

Configuring Metocard Validity Marker

By default, it will mark metacards with validation errors and warnings as they are reported by each metocard validator and then allow the ingest. To prevent the ingest of certain invalid metacards, the **Metocard Validity Marker** plugin can be configured to "enforce" one or more validators. Metacards that are invalid according to an "enforced" validator will not be ingested.

Table 117. Metocard Validation Filter Plugin

Name	Id	Type	Description	Default Value	Required
Attribute map	attributeMap	String	Mapping of Metocard SECURITY attribute to user attribute.	invalid-state=system-admin	false
Show errors	showErrors	Boolean	Sets whether metacards with validation errors are filtered.	true	false
Show warnings	showWarnings	Boolean	Sets whether metacards with validation warnings are filtered.	false	false

Table 118. Metocard Validation Marker Plugin

Name	Id	Type	Description	Default Value	Required
Enforced Validators	enforcedMetocardValidators	String	ID of Metocard Validator to enforce. Metacards that fail these validators will NOT be ingested.	true	false
Enforce errors	enforceErrors	Boolean	Sets whether validation errors are enforced	false	false
Enforce warnings	enforceWarnings	Boolean	Sets whether validation warnings are enforced	null	false

Using Metocard Validity Marker

Use this pre-ingest plugin to validate metacards against metocard validators, which can check schemas, schematron, or any other logic.

31.2.3. Expiration Date Pre-Ingest Plugin

The Expiration Date plugin updates metocard expiration dates.

Installing the Expiration Date Pre-Ingest Plugin

The Expiration Date Pre-Ingest Plugin is installed by default with a standard installation in the Catalog application.

Configuring the Expiration Date Pre-Ingest Plugin

The Expiration Date Pre-Ingest Plugin has no configurable properties.

31.2.4. Identification Plugin

The Identification Plugin assigns IDs to registry metacards and adds/updates IDs on create and update.

Installing the Identification Plugin

The Identification Plugin is not installed by default in a standard installation. It is installed by default with the [Registry](#) application.

Configuring the Identification Plugin

The Identification Plugin has no configurable properties.

31.2.5. Security Logging Plugin

The Security Logging Plugin logs operations to the security log.

Installing Security Logging Plugin

The Security Logging Plugin is installed by default in a standard installation in the Security application.

Configuring the Security Logging Plugin

The Security Logging Plugin has no configurable properties.

31.3. Pre-Query Plugins

Pre-query plugins are invoked before a query operation is sent to any of the Sources. This is an opportunity to take any action on the query, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

Included Pre-Query Plugins

[*Catalog Metrics Plugin*](#)

The Catalog Metrics Plugin captures metrics on catalog operations.

[*Security Logging Plugin*](#)

The Security Logging Plugin logs operations to the security log.

31.3.1. Catalog Metrics Plugin

The Catalog Metrics Plugin captures metrics on catalog operations.

Installing the Catalog Metrics Plugin

The Catalog Metrics Plugin is installed by default with a standard installation in the Catalog application.

Configuring the Catalog Metrics Plugin

The Catalog Metrics Plugin has no configurable properties.

31.4. Pre-Resource Plugins

Pre-Resource plugins are invoked before a request to retrieve a resource is sent to a Source. This is an opportunity to take any action on the request, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

Included Pre-Resource Plugins

[Resource Usage Plugin](#)

Monitors and limits data usage, as well as enable monitoring and cancelling running queries.

[Security Logging Plugin](#)

Logs operations to the security log.

31.4.1. Resource Usage Plugin

The Resource Usage Plugin monitors and limits data usage, and enables monitoring and cancelling running queries.

Installing the Resource Usage Plugin

The Resource Usage Plugin is not installed by default with a standard installation. It is installed with the \${resource-management} application.

Configuring the Resource Usage Plugin

The Resource Usage Plugin can be configured from the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **\${resource-management}** application.

3. Select the **Configuration** tab.

4. Select **Data Usage**.

Table 119. Data Usage

Name	Id	Type	Description	Default Value	Required
Monitor Local Sources	<code>monitorLocalSources</code>	Boolean	When checked, the Data Usage Plugin will also consider data usage from local sources.	false	true

31.5. Pre-Subscription Plugins

Pre-subscription plugins are invoked before a Subscription is activated by an Event Processor. This is an opportunity to take any action on the Subscription, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

31.6. Pre-Delivery Plugins

Pre-delivery plugins are invoked before a Delivery Method is invoked on a Subscription. This is an opportunity to take any action before notification, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.

31.7. Pre-Create Storage Plugins

Included Pre-Delivery Plugins

Pre-Create storage plugins are invoked immediately before an item is created in the content repository.

Included Pre-Create Storage Plugins

Checksum Plugin

Creates a unique checksum for resources input into the system to identify updated content.

Security Logging Plugin

Logs operations to the security log.

31.7.1. Checksum Plugin

The Checksum plugin creates a unique checksum for resources input into the system to identify updated content.

Installing the Checksum Plugin

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

Configuring the Checksum Plugin

The Checksum Plugin has no configurable properties.

31.8. Pre-Update Storage Plugins

Pre-Update storage plugins are invoked immediately before an item is updated in the content repository.

Included Pre-Update Storage Plugins

[Checksum Plugin](#)

Creates a unique checksum for resources input into the system to identify updated content.

[Security Logging Plugin](#)

Logs operations to the security log.

31.9. Post-Ingest Plugins

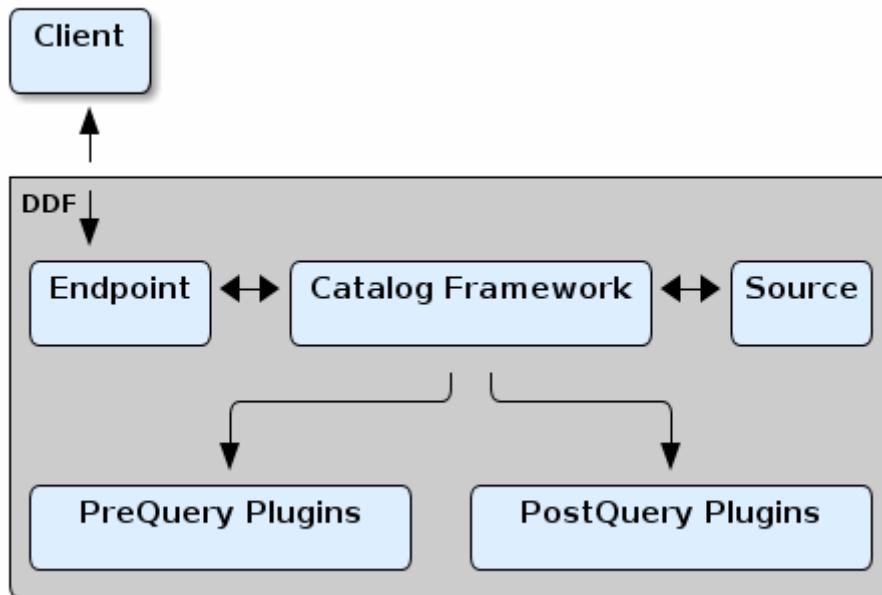


Figure 39. Query Plugin Flow

Post-ingest plugins are invoked after data has been created, updated, or deleted in a Catalog

Provider.

Included Post-Ingest Plugins

[*Catalog Backup Plugin*](#)

Enables data backup of the catalog and the metacards it contains.

[*Catalog Metrics Plugin*](#)

The Catalog Metrics Plugin captures metrics on catalog operations.

[*Event Processor*](#)

Creates, updates, and deletes subscriptions for event notification.

[*Fanout Event Processor*](#)

Creates, updates, and deletes subscriptions for event notification.

[*Security Logging Plugin*](#)

The Security Logging Plugin logs operations to the security log.

31.9.1. Catalog Backup Plugin

The Catalog Backup Plugin is used to enable data backup of the catalog and the metacards it contains.

Installing the Catalog Backup Plugin

The Catalog Backup Plugin is installed by default with a standard installation in the Catalog application.

Configuring the Catalog Backup Plugin

To configure the Catalog Backup Plugin:

1. Navigate to the **Admin Console**.
2. Select Catalog application.
3. Select **Configuration** tab.
4. Select **Backup Post-Ingest Plugin**.

Table 120. Backup Post-Ingest Plugin

Name	ID	Type	Description	Default Value	Required
Enable Backup Plugin	enableBackupPlugin	Boolean	Enable the Backup Ingest plugin which will write each result to a directory	true	false
Root backup directory path	rootBackupDir	String	Root backup directory for Metacards. A relative path is relative to ddf.home.	data/backup	true

Name	Id	Type	Description	Default Value	Required
Subdirectory levels	<code>subDirLeve ls</code>	Integer	Number of subdirectory levels to create. Two characters from the ID will be used to name each subdirectory level.	2	true

31.9.2. Event Processor

The Event Processor provides an engine that creates, updates, and deletes subscriptions for event notification. These subscriptions optionally specify a filter criteria so that only events of interest to the subscriber are posted for notification.

An internal subscription tracker monitors the OSGi registry, looking for subscriptions to be added (or deleted). When it detects a subscription being added, it informs the Event Processor, which sets up the subscription's filtering and is responsible for posting event notifications to the subscriber when events satisfying their criteria are met.

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

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

- invokes each pre-delivery plugin on the metocard in the event.
- invokes the Delivery Method's operation corresponding to the type of event being processed, e.g., created operation for the creation of a metocard.

Installing the Event Processor

The Event Processor is installed by default with a standard installation in the Catalog application.

Configuring the Event Processor

The Event Processor has no configurable properties.

Event Processing and Notification

As metacards are created, updated, and deleted, the Catalog's Event Processor is invoked (as a post-ingest plugin) for each of these events. The Event Processor applies the filter criteria for each registered subscription to each of these ingest events to determine if they match the criteria. If an event matches a subscription's criteria, any pre-delivery plugins that are installed are invoked, the subscription's Delivery Method is retrieved, and its operation corresponding to the type of ingest event is invoked. For example, the DeliveryMethod's `created()` function is called when a metocard is created. The Delivery Method's operations subsequently invoke the corresponding operation in the client's event consumer service, which is specified by the callback URL provided when the Delivery Method was created.

Available Event Processors

- Standard Event Processor
- Fanout Event Processor

Using DDF Implementation

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

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

Creating a Subscription

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

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

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

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

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

Usage Limitations of the Event Processor

The Standard Event processor currently broadcasts federated events and should not. It should only broadcast events that were generated locally, all other events should be dropped.

31.10. Post-Query Plugins

Post-query plugins are invoked after a query has been executed successfully, but before the response is returned to the endpoint. This is an opportunity to take any action on the query response, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.

deduplication.

Included Post-Query Plugins

Catalog Metrics Plugin

The Catalog Metrics Plugin captures metrics on catalog operations.

Resource Usage Plugin

Monitors and limits data usage, and enables monitoring and cancelling running queries.

Security Logging Plugin

The Security Logging Plugin logs operations to the security log.

Jpeg2000 Thumbnail Converter

Creates thumbnails from images ingested in jpeg2000 format.

Metocard Resource Size Plugin

The Metocard Resource Size Plugin post-query plugin updates the resource size attribute of each metocard in the query results.

31.10.1. JPEG2000 Thumbnail Converter

The JPEG2000 Thumbnail converter creates thumbnails from images ingested in jpeg2000 format.

Installing the JPEG2000 Thumbnail Converter

The JPEG2000 Thumbnail Converter is installed by default with a standard installation in the Catalog application.

Configuring the JPEG2000 Thumbnail Converter

The JPEG2000 Thumbnail Converter has no configurable properties.

31.10.2. Metocard Resource Size Plugin

This post-query plugin updates the resource size attribute of each metocard in the query results if there is a cached file for the product and it has a size greater than zero; otherwise, the resource size is unmodified and the original result is returned.

Use this post-query plugin as a convenience to return query results with accurate resource sizes for cached products.

Installing the Metocard Resource Size Plugin

The Metocard Resource Size Plugin is installed by default with a standard installation.

Configuring the Metocard Resource Size Plugin

The Metocard Resource Size Plugin has no configurable properties.

•

31.11. Post-Resource Plugins

Post-resource plugins are invoked after a resource has been retrieved, but before it is returned to the endpoint. This is an opportunity to take any action on the response, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.

Post-Resource Plugins

Catalog Metrics Plugin

The Catalog Metrics Plugin captures metrics on catalog operations.

Resource Usage Plugin

Monitors and limits data usage, and enables monitoring and cancelling running queries.

Security Logging Plugin

The Security Logging Plugin logs operations to the security log.

31.12. Post-Create Storage Plugins

Post-Create storage plugins are invoked immediately after an item is created in the content repository.

Included Post-Create Storage Plugins

Video Thumbnail

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

Security Logging Plugin

Logs operations to the security log.

31.12.1. 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_HOME>/bin_third_party/ffmpeg`. When invoked, this plugin runs the FFmpeg binary in a separate process to generate the thumbnail. The `<DDF_HOME>/bin_third_party/ffmpeg` directory is deleted when the plugin is uninstalled.

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

Installing the Video Thumbnail Plugin

The Video Thumbnail Plugin is not installed by default with a standard installation.

Configuring the Video Thumbnail Plugin

The Video Thumbnail Plugin has no configurable properties.

31.13. Post-Update Storage Plugins

Post-Update storage plugins are invoked immediately after an item is updated in the content repository.

Included Post-Update Storage Plugins

Video Thumbnail

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

Security Logging Plugin

Logs operations to the security log.

31.14. Policy Plugins

Policy plugins are invoked before all other plugin types to set up the policy for a request/response. This provides an opportunity to attach custom requirements on operations or individual metacards. All the 'requirements' from each Policy plugin will be combined into a single policy that will be included in the request/response. Access plugins will be used to act on this combined policy.

Included Policy Plugins

Catalog Policy Plugin

Configures the attributes required for users to perform Create, Read, Update, and Delete operations on the catalog.

Resource URI Policy Plugin

Configures the attributes required for users to perform Create or Update operations on resource URIs.

Historian Policy Plugin

tracks updates to metacards to enable viewing the history of metadata.

Metocard Attribute Security Policy Plugin

pulls attributes directly off of a metocard and combine these attributes into a security field for the metocard.

Metocard Validity Filter Plugin

aps the attributes between user and metocard and determines whether metacards with validation errors or warnings are filtered from query results.

[Registry Policy Plugin](#)

Defines the policies for user access to registry entries and operations.

[XML Attribute Security Policy Plugin](#)

Parses XML metadata contained within a metocard for security attributes on any number of XML elements in the metadata.

31.14.1. Catalog Policy Plugin

The Catalog Policy Plugin configures the attributes required for users to perform Create, Read, Update, and Delete operations on the catalog.

Installing the Catalog Policy Plugin

The Catalog Policy Plugin is installed by default with a standard installation in the \${app-name} application.

Configuring the Catalog Policy Plugin

To configure the Catalog Policy Plugin:

1. Navigate to the **Admin Console**.
2. Select Catalog application.
3. Select **Configuration** tab.
4. Select **Catalog Policy Plugin**.

Table 121. Catalog Policy Plugin

Name	Id	Type	Description	Default Value	Required
Create Required Attributes	createPermissions	String	Roles/attributes required for the create operations. Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Update Required Attributes	updatePermissions	String	Roles/attributes required for the update operation. Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true

Name	Id	Type	Description	Default Value	Required
Delete Required Attributes	<code>deletePermissions</code>	String	Roles/attributes required for the delete operation. Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role =guest	true
Read Required Attributes	<code>readPermissions</code>	String	Roles/attributes required for the read operations (query and resource). Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role =guest	true

31.14.2. Metocard Attribute Security Policy Plugin

The **Metocard Attribute Security Policy Plugin** pulls attributes directly off of a metocard and combine these attributes into a security field for the metocard. This plugin assumes that the pertinent information has already been parsed out of the metadata and placed directly on the metocard itself. The plugin supports mapping attributes from their names on the metocard to a different security policy name. Attributes can be mapped by union or intersection. E.g. A union mapping of `src1=dest` and `src2=dest` will result in `dest` containing the union of all attributes from `src1` and `src2`; an intersection mapping of `src1=dest` and `src2=dest` will result in `dest` containing the intersection of common attributes from `src1` and `src2`.

Installing the Metocard Attribute Security Policy Plugin

The Metocard Attribute Security Policy Plugin is installed by default with a standard installation in the Catalog application.

Configuring the Metocard Attribute Security Policy Plugin

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application tile
3. Select the **Configuration** tab
4. Select the **Metocard Attribute Security Policy Plugin**.

Table 122. Metocard Attribute Security Policy Plugin

Name	Id	Type	Description	Default Value	Required
Metocard Intersect Attributes:	<code>intersectMetocardAttributes</code>	String	Metocard attributes that will be collected and mapped to security information. Example: security.classification=classification Any duplicate destinations in this group will result in an intersection of values. E.g. source1=dest and source2=val where source1 is the set of		true
Metocard Union Attributes:	<code>unionMetocardAttributes</code>	String	'a', 'b', 'c'	null	true

31.14.3. Metocard Validity Filter Plugin

The Metocard Validity Filter Plugin determines whether metacards with validation errors or warnings are filtered from query results.

Installing the Metocard Validity Filter Plugin

The Metocard Validity Filter Plugin is installed by default with a standard installation in the Catalog application.

Configuring the Metocard Validity Filter Plugin

The Metocard Validity Filter Plugin can be configured:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **Metocard Validity Filter Plugin**.

Table 123. Metocard Validation Filter Plugin

Name	Id	Type	Description	Default Value	Required
Attribute map	<code>attributeMap</code>	String	Mapping of Metocard SECURITY attribute to user attribute.	invalid-state=system-admin	false
Show errors	<code>showErrors</code>	Boolean	Sets whether metacards with validation errors are filtered.	true	false
Show warnings	<code>showWarnings</code>	Boolean	Sets whether metacards with validation warnings are filtered.	false	false

Table 124. Metocard Validation Marker Plugin

Name	Id	Type	Description	Default Value	Required
Enforced Validators	<code>enforcedMetacardValidators</code>	String	ID of Metocard Validator to enforce. Metacards that fail these validators will NOT be ingested.	true	false
Enforce errors	<code>enforceErrors</code>	Boolean	Sets whether validation errors are enforced	false	false
Encorce warnings	<code>enforceWarnings</code>	Boolean	Sets whether validation warnings are enforced	null	false

31.14.4. Registry Policy Plugin

The Registry Policy Plugin defines the policies for user access to registry entries and operations.

Installing the Registry Policy Plugin

The Registry Policy Plugin is not installed by default on a standard installation. It is installed with the [Registry](#) application.

Configuring the Registry Policy Plugin

The Registry Policy Plugin can be configured from the Admin Console:

1. Navigate to the [Admin Console](#).
2. Select the [Registry](#) application.
3. Select the [Configuration](#) tab.
4. Select **Registry Policy Plugin**.

Table 125. Registry Policy Plugin

Name	Id	Type	Description	Default Value	Required
Registry Create Attributes	<code>createAccessPolicyStrings</code>	String	Roles/attributes required for Create operations on registry entries. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Registry Update Attributes	<code>updateAccessPolicyStrings</code>	String	Roles/attributes required for Update operations on registry entries. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true

Name	Id	Type	Description	Default Value	Required
Registry Delete Attributes	<code>deleteAccessPolicyStrings</code>	String	Roles/attributes required for Delete operations on registry entries. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Registry Read Attributes	<code>readAccessPolicyStrings</code>	String	Roles/attributes required for reading registry entries. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Registry Admin Attributes	<code>registryByPassPolicyStrings</code>	String	Roles/attributes required for an admin to bypass all filtering/access controls. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin	true
Disable Registry Write Access	<code>registryDisabled</code>	Boolean	Disables all write access to registry entries in the catalog. Only users with Registry Admin Attributes will be able to write registry entries	null	false
Entries are White List	<code>whiteList</code>	Boolean	A flag indicating whether or not the Registry Entry Ids represent a 'white list' (allowed - checked) or a 'black list' (blocked - unchecked) ids	null	false
Registry Entries Ids	<code>registryEntryIds</code>	String	List of registry entry ids to be used in the white/black list.	null	false

31.14.5. XML Attribute Security Policy Plugin

The **XML Attribute Security Policy Plugin** parses XML metadata contained within a metocard for security attributes on any number of XML elements in the metadata. The configuration for the plugin contains one field for setting the XML elements that will be parsed for security attributes and the other two configurations contain the XML attributes that will be pulled off of those elements. The **Security Attributes (union)** field will compute the union of values for each attribute defined. While the **Security Attributes (intersection)** field will compute the intersection of values for each attribute defined.

Installing the XML Attribute Security Policy Plugin

The XML Attribute Security Policy Plugin is installed by default with a standard installation in the

`${app-name}` application.

Configuring the XML Attribute Security Policy Plugin

1. Navigate to the Admin Console.
2. Select the **Catalog** application tile.
3. Select the **Configuration** tab.
4. Select the **XML Attribute Security Policy Plugin** configuration.

Table 126. XML Attribute Security Policy Plugin

Name	Id	Type	Description	Default Value	Required
XML Elements:	<code>xmlElements</code>	String	XML elements within the metadata that will be searched for security attributes. If these elements contain matching attributes, the values of the attributes will be combined.		true
Security Attributes (union):	<code>securityAttributeUnions</code>	String	Security Attributes. These attributes, if they exist on any of the XML elements listed above, will have their values extracted and the union of all of the values will be saved to the metocard. For example: if element1 and element2 both contain the attribute 'attr' and that attribute has values X,Y and X,Z, respectively, then the final result will be the union of those values: X,Y,Z. The X,Y,Z value will be the value that is placed within the security attribute on the metocard.		false
Security Attributes (intersection):	<code>securityAttributeIntersections</code>	String	Security Attributes. These attributes, if they exist on any of the XML elements listed above, will have their values extracted and the intersection of all of the values will be saved to the metocard. For example: if element1 and element2 both contain the attribute 'attr' and that attribute has values X,Y and X,Z, respectively, then the final result will be the intersection of those values: X. The X value will be the value that is placed within the security attribute on the metocard.	null	false

31.15. Access Plugins

Access plugins are invoked directly after the [Policy plugins](#) have been successfully executed. This is an opportunity to either stop processing or modify the request/response based on policy information.

Included Access Plugins

[Operation Plugin](#)

Validates the subject's security attributes are adequate to make the request.

31.15.1. Operation Plugin

The operation plugin validates the subject's security attributes to ensure they are adequate to make the request.

Installing the Operation Plugin

The Operation Plugin is installed by default with a standard installation in the Catalog application.

Configuring the Operation Plugin

The Operation Plugin has no configurable properties.

31.15.2. Metocard Ingest Network Plugin

The Metocard Ingest Network Plugin allows the conditional insertion of new attributes on metacards during ingest based on network information from the ingest request; including IP address and hostname.

For the extent of this section, a 'rule' will refer to a configured, single instance of this plugin.

Installing the Metocard Ingest Network Plugin

The Metocard Ingest Network Plugin is installed by default during a standard installation in the Catalog application.

Configuring the Metocard Ingest Network Plugin

To configure the Metocard Ingest Network Plugin:

- Click on the Catalog application
- Click on the **Configuration** tab
- Click on the label *Metocard Ingest Network Plugin* to setup a network rule

Multiple instances of the plugin can be configured by clicking on its configuration title within the configuration tab of the catalog app. Each instance represents a conditional statement, or a 'rule', that gets evaluated for each ingest request. For any request that meets the configured criteria of a rule, that rule will attempt to transform its list of key-value pairs to become new attributes on all metacards in that request.

The rule is divided into two fields: "Criteria" and "Expected Value". The "Criteria" field features a drop-down list containing the four elements for which equality can be tested:

- IP Address of where the ingest request came from
- Host Name of where the ingest request came from
- Scheme that the ingest request arrived on, for example, *http* vs *https*
- Context Path that the ingest request arrived on, for example, */services/catalog*

In order for a rule to evaluate to true and the attributes be applied, the value in the "Expected Value" field must be an exact match to the actual value of the selected criteria. For example, if the selected criteria is "IP Address" with an expected value of "192.168.0.1", the rule only evaluates to true for ingest requests coming from "192.168.0.1" and nowhere else.

Check for IPv6

IMPORTANT Verify your system's IP configuration. Rules using "IP Address" may need to be written in IPv6 format.

The key-value pairs within each rule should take the following form: "key = value" where the "key" is the name of the attribute and the "value" is the value assigned to that attribute. Whitespace is ignored unless it is within the key or value. Multi-valued attributes can be expressed in comma-separated format if necessary.

Examples of Valid Attribute Assignments

```
contact.contributor-name = John Doe  
contact.contributor-email = john.doe@example.net  
language = English  
language = English, French, German  
security.access-groups = SJ202, SR 101, JS2201
```

Useful Attributes

The following table provides some useful attributes that may commonly be set by this plugin:

Table 127. Useful Attributes

Attribute Name	Expected Format	Multi-Valued
expiration	ISO DateTime	no
description	Any String	no
metocard.owner	Any String	no
language	Any String	yes
security.access-groups	Any String	yes
security.access-individuals	Any String	yes

Usage Limitations of the Metocard Ingest Network Plugin

- This plugin only works for ingest (create requests) performed over a network; data ingested via command line does not get processed by this plugin.
- Any attribute that is already set on the metocard will not be overwritten by the plugin.
- The order of execution is not guaranteed. For any rule configuration where two or more rules add different values for the same attribute, it is undefined what the final value for that attribute will be in the case where more than one of those rules evaluates to true.

31.16. Developing Catalog Plugins

Plugins extend the functionality of the Catalog Framework by performing actions at specified times during a transaction. Plugin interfaces are located in the Catalog Core API. By implementing a plugin interface, actions can be performed at the desired time.

NOTE Catalog plugin interfaces are located in the catalog core API.

The following types of plugins can be created:

Table 128. Plugin Interfaces

Plugin Type	Plugin Interface	Description	Example
Pre-Ingest	<code>ddf.catalog.plugin.PreIngestPlugin</code>	Runs before the Create/Update/Delete method is sent to the CatalogProvider	Metadata validation services
Post-Ingest	<code>ddf.catalog.plugin.PostIngestPlugin</code>	Runs after the Create/Update/Delete method is sent to the CatalogProvider	EventProcessor for processing and publishing event notifications to subscribers
Pre-Query	<code>ddf.catalog.plugin.PreQueryPlugin</code>	Runs prior to the Query/Read method being sent to the Source	An example is not included with DDF
Post-Query	<code>ddf.catalog.plugin.PostQueryPlugin</code>	Runs after results have been retrieved from the query but before they are posted to the Endpoint	An example is not included with DDF
Pre-Subscription	<code>ddf.catalog.plugin.PreSubscription</code>	Runs prior to a Subscription being created or updated	Modify a query prior to creating a subscription

Plugin Type	Plugin Interface	Description	Example
Pre-Delivery	<code>ddf.catalog.plugin.PreDeliveryPlugin</code>	Runs prior to the delivery of a Metacard when an event is posted	Inspect a metocard prior to delivering it to the Event Consumer
Pre-Resource	<code>ddf.catalog.plugin.PreResource</code>	Runs prior to a Resource being retrieved	An example is not included with DDF
Post-Resource	<code>ddf.catalog.plugin.PostResource</code>	Runs after a Resource is retrieved, but before it is sent to the Endpoint	Verification of a resource prior to returning to a client
Policy	<code>ddf.catalog.plugin.PolicyPlugin</code>	Runs prior to all other catalog plugins to establish the policy for requests/responses	An example is MetacardValidityFilterPlugin
Access	<code>ddf.catalog.plugin.AccessPlugin</code>	Runs directly after the PolicyPlugin	An examples are the FilterPlugin and OperationPlugin

31.16.1. Implement Plugins

The procedure for implementing any of the plugins follows a similar format:

1. Create a new class that implements the specified plugin interface.
2. Implement the required methods.
3. Create OSGi descriptor file to communicate with the OSGi registry.
 - a. Register plugin class as service to OSGi registry.
4. Deploy to DDF.

Plugin Performance Concerns

NOTE Plugins should include a check to determine if requests are local or not. It is usually preferable to take no action on non-local requests.

TIP Refer to the Javadoc for more information on all Requests and Responses in the `ddf.catalog.operation` and `ddf.catalog.event` packages.

Implementing Pre-Ingest Plugins

Develop a custom Pre-Ingest Plugin.

1. Create a Java class that implements `PreIngestPlugin`.

```
public class SamplePreIngestPlugin implements ddf.catalog.plugin.PreIngestPlugin
```

2. Implement the required methods.

- `public CreateRequest process(CreateRequest input) throws PluginExecutionException, StopProcessingException;`
- `public UpdateRequest process(UpdateRequest input) throws PluginExecutionException, StopProcessingException;`
- `public DeleteRequest process(DeleteRequest input) throws PluginExecutionException, StopProcessingException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin
```

4. Export the service to the OSGi registry.

```
Blueprint descriptor example <service ref="[[SamplePreIngestPlugin]]">interface="ddf.catalog.plugin.PreIngestPlugin" />
```

Pre-Ingest Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a `StopProcessingException` will be thrown.

Invoking Pre-Ingest Plugins

Pre-Ingest plugins are invoked serially, prioritized by descending OSGi service ranking. The plugin with the highest service ranking will be executed first.

The output of a Pre-Ingest plugin is sent to the next Pre-Ingest plugin, until all have executed and the ingest operation is sent to the requested Source.

Implementing Post-Ingest Plugins

Develop a custom Post-Ingest Plugin.

1. Create a Java class that implements `PostIngestPlugin`.

```
public class SamplePostIngestPlugin implements ddf.catalog.plugin.PostIngestPlugin
```

2. Implement the required methods.

- `public CreateResponse process(CreateResponse input) throws PluginExecutionException;`
- `public UpdateResponse process(UpdateResponse input) throws PluginExecutionException;`
- `public DeleteResponse process(DeleteResponse input) throws PluginExecutionException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin
```

4. Export the service to the OSGi registry.

```
Blueprint descriptor example <service ref="[[SamplePostIngestPlugin]]">interface="ddf.catalog.plugin.PostIngestPlugin" />
```

Post-Ingest Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` will be thrown.

Invoking Post-Ingest Plugins

Post-Ingest plugins are invoked serially in rank order.

Implementing Pre-Query Plugins

Develop a custom Pre-Query Plugin

1. Create a Java class that implements `PreQueryPlugin`.

```
public class SamplePreQueryPlugin implements ddf.catalog.plugin.PreQueryPlugin
```

2. Implement the required method.

```
public QueryRequest process(QueryRequest input) throws PluginExecutionException,  
StopProcessingException;
```

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin
```

4. Export the service to the OSGi registry.

```
<service ref="" interface="ddf.catalog.plugin.PreQueryPlugin" />
```

Pre-Query Plugins Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a `StopProcessingException` will be thrown.

Invoking Pre-Query Plugins

Pre-query plugins are invoked serially, prioritized by descending OSGi service ranking. The plugin with the highest service ranking will be executed first. The output of a pre-query plugin is sent to the next pre-query plugin, until all have executed and the query operation is sent to the requested Source.

Implementing Post-Query Plugins

Develop a custom Post-Query Plugin

1. Create a Java class that implements `PostQueryPlugin`.

```
public class SamplePostQueryPlugin implements ddf.catalog.plugin.PostQueryPlugin
```

2. Implement the required method.

```
public QueryResponse process(QueryResponse input) throws PluginExecutionException,  
StopProcessingException;
```

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin
```

4. Export the service to the OSGi registry.

```
<service ref="" interface="ddf.catalog.plugin.PostQueryPlugin" />
```

Post-Query Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a **PluginExecutionException** will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a **StopProcessingException** will be thrown.

Invoking Post-Query Plugins

Post-query plugins are invoked serially, prioritized by descending OSGi service ranking. The plugin with the highest service ranking will be executed first. The output of the first plugin is sent to the next plugin, until all have executed and the response is returned to the requesting endpoint.

Implementing Pre-Delivery Plugins

Develop a custom Pre-Delivery Plugin.

1. Create a Java class that implements **PreDeliveryPlugin**.

```
public class SamplePreDeliveryPlugin implements ddf.catalog.plugin.PreDeliveryPlugin
```

2. Implement the required methods.

```
public Metocard processCreate(Metocard metocard) throws PluginExecutionException,  
StopProcessingException; public Update processUpdateMiss(Update update) throws  
PluginExecutionException, StopProcessingException;  
• public Update processUpdateHit(Update update) throws PluginExecutionException,  
StopProcessingException;  
• public Metocard processCreate(Metocard metocard) throws PluginExecutionException,  
StopProcessingException;
```

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation,ddf.catalog.event
```

4. Export the service to the OSGi registry.

Blueprint descriptor example

```
<service ref="" interface="ddf.catalog.plugin.PreDeliveryPlugin" />
```

Pre-Delivery Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a **PluginExecutionException** will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a **StopProcessingException** will be thrown.

Invoking Pre-Delivery Plugins

Pre-delivery plugins are invoked serially, prioritized by descending OSGi service ranking. The plugin with the highest service ranking will be executed first.

The output of a pre-delivery plugin is sent to the next pre-delivery plugin, until all have executed and the Delivery Method is invoked on the associated Subscription.

Implementing Pre-Subscription Plugins

Develop a custom Pre-Subscription Plugin.

1. Create a Java class that implements `PreSubscriptionPlugin`.

```
`public class SamplePreSubscriptionPlugin implements  
ddf.catalog.plugin.PreSubscriptionPlugin
```

2. Implement the required method.

- `public Subscription process(Subscription input) throws PluginExecutionException,
StopProcessingException;`

Pre-Subscription Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a `StopProcessingException` will be thrown.

Invoking Pre-Subscription Plugin

Pre-subscription plugins are invoked serially, prioritized by descending OSGi service ranking. That is, the plugin with the highest service ranking will be executed first.

The output of a pre-subscription plugin is sent to the next pre-subscription plugin, until all have executed and the create Subscription operation is sent to the Event Processor.

Implementing Pre-Resource Plugins

Develop a custom Pre-Resource Plugin.

1. Create a Java class that implements `PreResourcePlugin`. `public class SamplePreResourcePlugin
implements ddf.catalog.plugin.PreResourcePlugin`

2. Implement the required method.

- `public ResourceRequest process(ResourceRequest input) throws PluginExecutionException,
StopProcessingException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

`Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation`

4. Export the service to the OSGi registry. .Blueprint descriptor example

```
<service ref="[[SamplePreResourcePlugin]]"  
interface="ddf.catalog.plugin.PreResourcePlugin" />
```

Pre-Resource Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a `StopProcessingException` will be thrown.

Invoking Pre-Resource Plugins

Pre-Resource plugins are invoked serially, prioritized by descending OSGi service ranking. That is, the plugin with the highest service ranking will be executed first.

The output of the first plugin is sent to the next plugin, until all have executed and the request is sent to the targeted Source.

Implementing Post-Resource Plugins

Develop a custom Post-Resource Plugin.

1. Create a Java class that implements `PostResourcePlugin`.

```
public class SamplePostResourcePlugin implements ddf.catalog.plugin.PostResourcePlugin
```

2. Implement the required method.

- `public ResourceResponse process(ResourceResponse input) throws PluginExecutionException, StopProcessingException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation
```

4. Export the service to the OSGi registry.

Blueprint descriptor example

```
<service ref="[[SamplePostResourcePlugin]]" interface="ddf.catalog.plugin.PostResourcePlugin" />
```

Post-Resource Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` will be thrown. For any other Exceptions, the Catalog will "fail safe" and the Operation will be cancelled. If processing is to be explicitly stopped, a `StopProcessingException` will be thrown.

Invoking Post-Resource Plugins

Post-get resource plugins are invoked serially, prioritized by descending OSGi service ranking. The plugin with the highest service ranking will be executed first.

The output of the first plugin is sent to the next plugin, until all have executed and the response is returned to the requesting endpoint.

Implementing Policy Plugins

Develop a custom Policy Plugin.

1. Create a Java class that implements `PolicyPlugin`.

```
public class SamplePolicyPlugin implements ddf.catalog.plugin.PolicyPlugin
```

2. Implement the required methods.

- `PolicyResponse processPreCreate(Metacard input, Map<String, Serializable> properties) throws StopProcessingException;`
- `PolicyResponse processPreUpdate(Metacard input, Map<String, Serializable> properties) throws StopProcessingException;`
- `PolicyResponse processPreDelete(String attributeName, List<Serializable> attributeValues, Map<String, Serializable> properties) throws StopProcessingException;`
- `PolicyResponse processPreQuery(Query query, Map<String, Serializable> properties) throws StopProcessingException;`
- `PolicyResponse processPostQuery(Result input, Map<String, Serializable> properties) throws StopProcessingException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation
```

4. Export the service to the OSGi registry.

Blueprint descriptor example

```
<service ref="" interface="ddf.catalog.plugin.PolicyPlugin" />
```

Failure Behavior

All failure cases should be handled internally to the plugin with the exception of the `StopProcessingException`. If the exception encountered should stop/block the request then a `StopProcessingException` should be thrown.

Implementing Access Plugins

Develop a custom Access Plugin.

1. Create a Java class that implements `AccessPlugin`.

```
public class SamplePostResourcePlugin implements ddf.catalog.plugin.AccessPlugin
```

2. Implement the required methods.

- `CreateRequest processPreCreate(CreateRequest input) throws StopProcessingException;`
- `UpdateRequest processPreUpdate(UpdateRequest input) throws StopProcessingException;`
- `DeleteRequest processPreDelete(DeleteRequest input) throws StopProcessingException;`
- `QueryRequest processPreQuery(QueryRequest input) throws StopProcessingException;`
- `QueryResponse processPostQuery(QueryResponse input) throws StopProcessingException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).

```
Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation
```

4. Export the service to the OSGi registry.

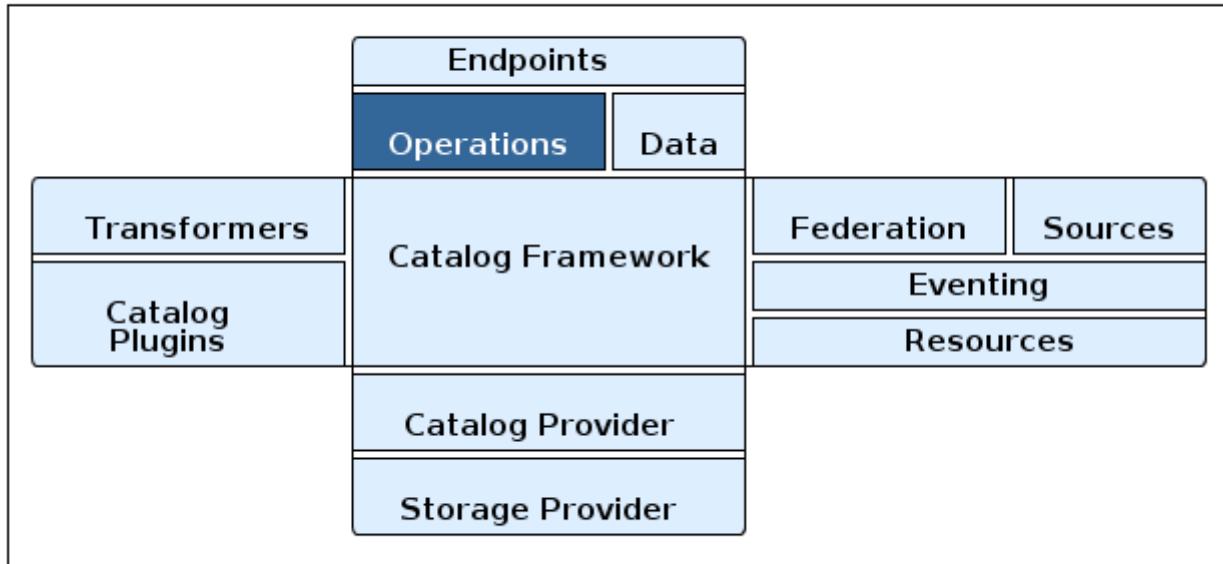
Blueprint descriptor example

```
<service ref="" interface="ddf.catalog.plugin.AccessPlugin" />
```

Access Plugins Failure Behavior

All failure cases should be handled internally to the plugin with the exception of the `StopProcessingException`. If the exception encountered should stop/block the request then a `StopProcessingException` should be thrown.

32. Operations



The Catalog provides the capability to query, create, update, and delete metacards; retrieve resources; and retrieve information about the sources in the enterprise.

Each of these operations follow a request/response paradigm. The request is the input to the operation and contains all of the input parameters needed by the Catalog Framework's operation to communicate with the Sources. The response is the output from the execution of the operation that is returned to the client, which contains all of the data returned by the sources. For each operation there is an associated request/response pair, e.g., the `QueryRequest` and `QueryResponse` pair for the Catalog Framework's query operation.

All of the request and response objects are extensible in that they can contain additional key/value properties on each request/response. This allows additional capability to be added without changing the Catalog API, helping to maintain backwards compatibility.

33. Resources

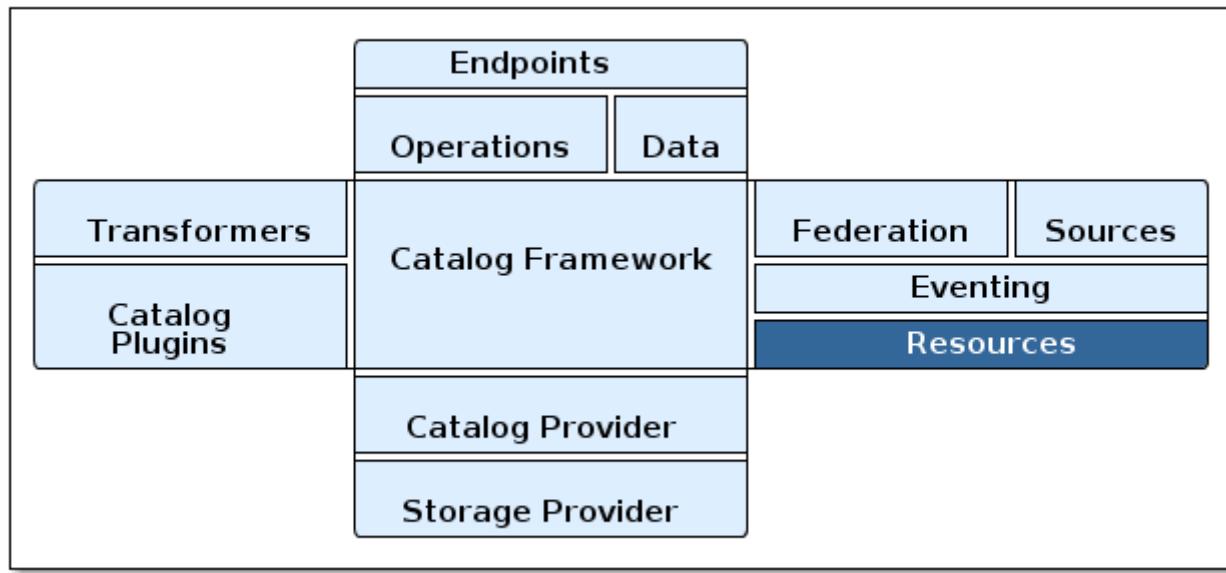


Figure 40. Resources Architecture

Resources are the data that is represented by the cataloged metadata in DDF.

Metacards are used to describe those resources through metadata. This metadata includes the time the resource was created, the location where the resource was created, etc. A DDF Metocard contains the `getResourceUri` method, which is used to locate and retrieve its corresponding resource.

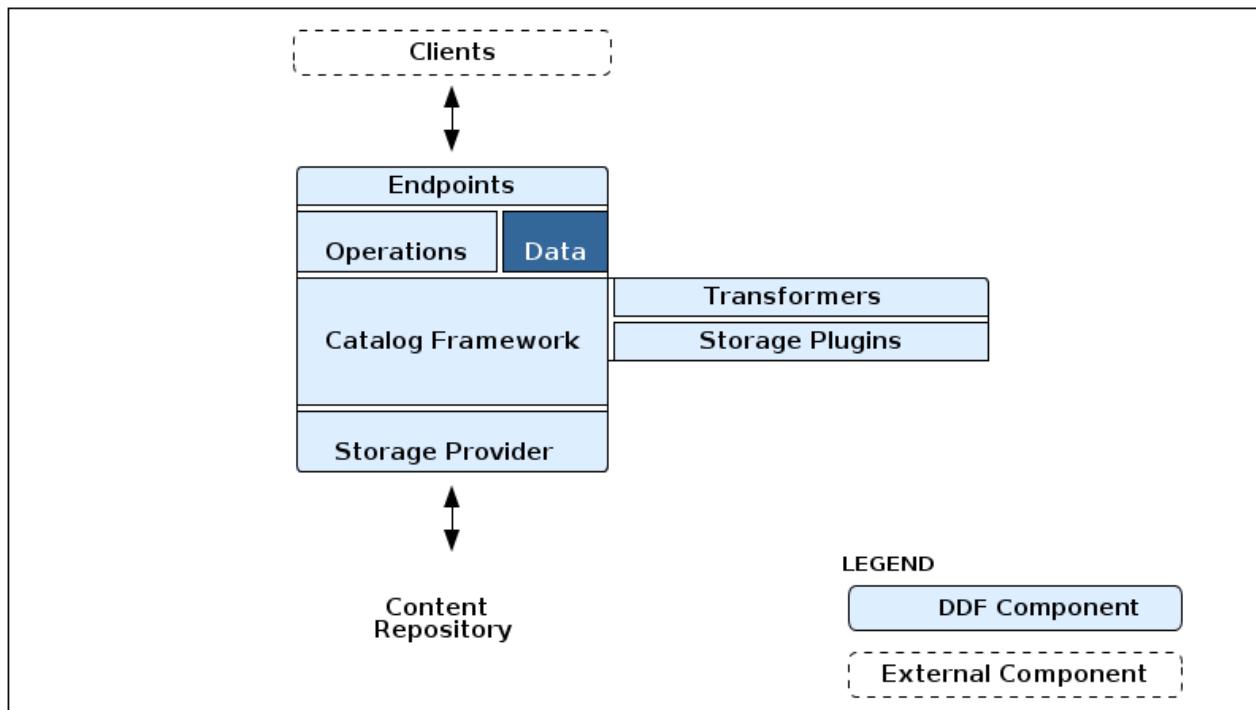


Figure 41. Content Data Component Architecture

33.1. 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 Metocard ID.

33.1.1. Retrieving Resources

When a client attempts to retrieve a resource, it must provide a metocard ID or URI corresponding to a unique resource. As mentioned above, the resource URI is obtained from a Metocard's 'getResourceUri' method. The CatalogFramework has three methods that can be used by clients to obtain a resource: `getEnterpriseResource`, `getResource`, and `getLocalResource`. The `getEnterpriseResource` method invokes the `retrieveResource` method on a local ResourceReader as well as all the Federated and Connected Sources inthe DDF enterprise. The second method, `getResource`, takes in a source ID as a parameter and only invokes `retrieveResource` on the specified Source. The third method invokes `retrieveResource` on a local ResourceReader.

The parameter for each of these methods in the CatalogFramework is a ResourceRequest. DDF includes two implementations of ResourceRequest: `ResourceRequestById` and `ResourceRequestByProductUri`. Since these implementations extend `OperationImpl`, they can pass a Map of generic properties through the CatalogFramework to customize how the resource request is carried out. One example of this is explained in the [Retrieving Resource Options](#) section below. The following is a basic example of how to create a ResourceRequest and invoke the CatalogFramework resource retrieval methods to process the request.

Retrieve Resource Example

```
Map<String, Serializable> properties = new HashMap<String, Serializable>();
properties.put("PropertyKey1", "propertyA"); //properties to customize Resource
retrieval
ResourceRequestById resourceRequest = new ResourceRequestById(
"0123456789abcdef0123456789abcdef", properties); //object containing ID of Resource to
be retrieved
String sourceName = "LOCAL_SOURCE"; //the Source ID or name of the local Catalog or a
Federated Source
ResourceResponse resourceResponse; //object containing the retrieved Resource and the
request that was made to get it.
resourceResponse = catalogFramework.getResource(resourceRequest, sourceName);
//Source-based retrieve Resource request
Resource resource = resourceResponse.getResource(); //actual Resource object
containing InputStream, mime type, and Resource name
```

DDF.`catalog.resource.ResourceReader` instances can be discovered via the OSGi Service Registry. The system can contain multiple `ResourceReaders`. The CatalogFramework determines which one to call based on the scheme of the resource's URI and what schemes the `ResourceReader` supports. The supported schemes are obtained by a `ResourceReader`'s 'getSupportedSchemes' method. As an example, one `ResourceReader` may know how to handle file-based URIs with the scheme `file`,

whereas another `ResourceReader` may support HTTP-based URIs with the scheme `http`.

The `ResourceReader` or `Source` is responsible for locating the resource, reading its bytes, adding the binary data to a `Resource` implementation, then returning that `Resource` in a `ResourceResponse`. The `ResourceReader` or `Source` is also responsible for determining the `Resource's name and mime type, which it sends back in the 'Resource implementation.`

BinaryContent

`BinaryContent` is an object used as a container to store translated or transformed DDF components. `Resource` extends `BinaryContent` and includes a `getName` method. ` `BinaryContent`` has methods to get the `InputStream`, byte array, MIME type, and size of the represented binary data. An implementation of `BinaryContent` (`BinaryContentImpl`) can be found in the Catalog API in the `DDF.catalog.data` package.

33.1.2. Retrieving Resource Options

Options can be specified on a retrieve resource request made through any of the supporting endpoint. To specify an option for a retrieve resource request, the endpoint needs to first instantiate a `ResourceRequestByProductUri` or a `ResourceRequestById`. Both of these `ResourceRequest` implementations allow a `Map` of properties to be specified. Put the specified option into the `Map` under the key `RESOURCE_OPTION`.

Retrieve Resource with Options

```
Map<String, Serializable> properties = new HashMap<String, Serializable>();
properties.put("RESOURCE_OPTION", "OptionA");
ResourceRequestById resourceRequest = new ResourceRequestById(
    "0123456789abcdef0123456789abcdef", properties);
```

Depending on the support that the `ResourceReader` or `Source` provides for options, the `properties`'Map` will be checked for the `RESOURCE_OPTION` entry. If that entry is found, the option will be handled. If the `ResourceReader` or `Source` does not support options, that entry will be ignored.

A new `ResourceReader` or `Source` implementation can be created to support options in a way that is most appropriate. Since the option is passed through the catalog framework as a property, the `ResourceReader` or `Source` will have access to that option as long as the endpoint supports options.

33.1.3. Storing Resources

Resources are saved using a `ResourceWriter`. `DDF.catalog.resource.ResourceWriter` instances can be discovered via the OSGi Service Registry. Once retrieved, the `ResourceWriter` instance provides clients with a way to store resources and get a corresponding URI that can be used to subsequently retrieve the resource via a `ResourceReader`. Simply invoke either of the `storeResource` methods with a resource and any potential arguments. The `ResourceWriter` implementation is responsible for determining where the resource is saved and how it is saved. This allows flexibility for a resource to be saved in any one of a variety of data stores or file systems. The following is an example of how to use a generic implementation of `ResourceWriter`.

Using a ResourceWriter

```
InputStream inputStream = <Video_Input_Stream>; //InputStream of raw Resource data
MimeType mimeType = new MimeType("video/mpeg"); //Mime Type or content type of
Resource
String name = "Facility_Video"; //Descriptive Resource name
Resource resource = new ResourceImpl(inputStream, mimeType, name);
Map<String, Object> optionalArguments = new HashMap<String, Object>();
ResourceWriter writer = new ResourceWriterImpl();
URI resourceUri; //URI that can be used to retrieve Resource
resourceUri = writer.storeResource(resource, optionalArguments); //Null can be passed
in here
```

33.2. Resource Components

Resource components are used when working with resources

A resource is a URI-addressable entity that is represented by a metocard. Resources may also be known as **products** or **data**.

Resources may exist either locally or on a remote data store.

Examples of resources include:

- NITF image
- MPEG video
- Live video stream
- Audio recording
- Document

A resource object in DDF contains an **InputStream** with the binary data of the resource. It describes that resource with a name, which could be a file name, URI, or another identifier. It also contains a mime type or content type that a client can use to interpret the binary data.

33.3. Resource Readers

A resource reader retrieves resources associated with metacards via URIs. Each resource reader must know how to interpret the resource's URI and how to interact with the data store to retrieve the resource.

There can be multiple resource readers in a Catalog instance. The **Catalog Framework** selects the appropriate resource reader based on the scheme of the resource's URI.

In order to make a resource reader available to the Catalog Framework, it must be exported to the OSGi Service Registry as a **DDF.catalog.resource.ResourceReader**.

Included Resource Readers

URL Resource Reader

The URL Resource Reader obtains a resource given an http, https, or file-based URL.

33.3.1. URL Resource Reader

The `URLResourceReader` is an implementation of `ResourceReader` which is included in the DDF Catalog. It obtains a resource given an http, https, or file-based URL. The `URLResourceReader` will connect to the provided Resource URL and read the resource's bytes into an `InputStream`.

WARNING

When a resource linked using a file-based URL is in the product cache, the `URLResourceReader`'s `rootResourceDirectories` is not checked when downloading the product. It is downloaded from the product cache which bypasses the `URLResourceReader`. For example, if path `/my/valid/path` is configured in the `URLResourceReader`'s `rootResourceDirectories` and one downloads the product with resource-uri `file:///my/valid/path/product.txt` and then one removes `/my/valid/path` from the `URLResourceReader`'s `rootResourceDirectories` configuration, the product will still be accessible via the product cache.

Installing the URL Resource Reader

The `URLResourceReader` is installed by default with a standard installation in the Catalog application.

Configuring the URL Resource Reader

Configure the URL Resource Reader from the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **URL Resource Reader**.

Table 129. URL Resource Reader

Name	Id	Type	Description	Default Value	Required
Follow Server Redirects	<code>followRedirects</code>	Boolean	Check the box if you want the Resource Reader to automatically follow server issued redirects (HTTP Response Code 300 series)	false	true
Root Resource Directories	<code>rootResourceDirectories</code>	String	List of root resource directories. A relative path is relative to <code>ddf.home</code> . Specifies the only directories the <code>URLResourceReader</code> has access to when attempting to download resources linked using file-based URLs.	<code>data/products</code>	true

Using the URL Resource Reader

[URLResourceReader](#) will be used by the Catalog Framework to obtain a resource whose metocard is cataloged in the local data store. This particular [ResourceReader](#) will be chosen by the [CatalogFramework](#) if the requested resource's URL has a protocol of [http](#), [https](#), or [file](#).

For example, requesting a resource with the following URL will make the Catalog Framework invoke the [URLResourceReader](#) to retrieve the product.

Example

```
file:///home/users/DDF_user/data/example.txt
```

If a resource was requested with the URL [udp://123.45.67.89:80/SampleResourceStream](#), the [URLResourceReader](#) would *not* be invoked.

Supported Schemes:

- [http](#)
- [https](#)
- [file](#)

NOTE

If a file-based URL is passed to the [URLResourceReader](#), that file path needs to be accessible by the DDF instance.

Usage Limitations of the URL Resource Reader

None.

33.3.2. Developing Resource Readers

A [ResourceReader](#) is a class that retrieves a resource or product from a native/external source and returns it to DDF. A simple example is that of a File [ResourceReader](#). It takes a file from the local file system and passes it back to DDF. New implementations can be created in order to support obtaining Resources from various Resource data stores.

Creating a New [ResourceReader](#)

Complete the following procedure to create a [ResourceReader](#).

1. Create a Java class that implements the [DDF.catalog.resource.ResourceReader](#) interface.
2. Deploy the OSGi bundled packaged service to the DDF run-time.

Implementing the [ResourceReader](#) Interface

```
public class TestResourceReader implements DDF.catalog.resource.ResourceReader
```

[ResourceReader](#) has a couple of key methods where most of the work is performed.

NOTE**URI**

It is recommended to become familiar with the Java API URI class in order to properly build a `ResourceReader`. Furthermore, a URI should be used according to its specification.

retrieveResource

```
public ResourceResponse retrieveResource( URI uri, Map<String, Serializable> arguments )throws IOException, ResourceNotFoundException, ResourceNotSupportedException;
```

This method is the main entry to the `ResourceReader`. It is used to retrieve a `Resource` and send it back to the caller (generally the `CatalogFramework`). Information needed to obtain the entry is contained in the `URI` reference. The URI Scheme will need to match a scheme specified in the `getSupportedSchemes` method. This is how the CatalogFramework determines which `ResourceReader` implementation to use. If there are multiple `ResourceReaders` supporting the same scheme, these `ResourceReaders` will be invoked iteratively. Invocation of the `ResourceReaders` stops once one of them returns a `Resource`.

Arguments are also passed in. These can be used by the `ResourceReader` to perform additional operations on the resource.

The `URLResourceReader` is an example `ResourceReader` that reads a file from a URI.

NOTE

The `Map<String, Serializable> arguments` parameter is passed in to support any options or additional information associated with retrieving the resource.

Implement `retrieveResource()`

1. Define supported schemes (e.g., file, http, etc.).
2. Check if the incoming URI matches a supported scheme. If it does not, throw `ResourceNotSupportedException`.

Example:

```
if ( !uri.getScheme().equals("http") )
{
    throw new ResourceNotSupportedException("Unsupported scheme received, was expecting
http")
}
```

1. Implement the business logic.
2. For example, the `URLResourceReader` will obtain the resource through a connection:

```

URL url = uri.toURL();
URLConnection conn = url.openConnection();
String mimeType = conn.getContentType();
if ( mimeType == null ) {
    mimeType = URLConnection.guessContentTypeFromName( url.getFile() );
}
InputStream is = conn.getInputStream();

```

NOTE The **Resource** needs to be accessible from the DDF installation (see the `rootResourceDirectories` property of the `URLResourceReader`). This includes being able to find a file locally or reach out to a remote URI. This may require Internet access, and DDF may need to be configured to use a proxy (`http.proxyHost` and `http.proxyPort` can be added to the system properties on the command line script).

1. Return **Resource** in `ResourceResponse`.

For example:

```

return ResourceResponseImpl( new ResourceImpl( new BufferedInputStream( is ), new
MimeType( mimeType ), url.getFile() ) );

```

If the **Resource** cannot be found, throw a `ResourceNotFoundException`.

getSupportedSchemes

```

public Set<String> getSupportedSchemes();

```

This method lets the `ResourceReader` inform the CatalogFramework about the type of URI scheme that it accepts and should be passed. For single-use ResourceReaders (like a `URLResourceReader`), there may be only one scheme that it can accept while others may understand more than one. A `ResourceReader` must, at minimum, accept one qualifier. As mentioned before, this method is used by the `CatalogFramework` to determine which `ResourceReader` to invoke.

NOTE

`ResourceReader` extends `Describable`

Additionally, there are other methods that are used to uniquely describe a `ResourceReader`. The `describe` methods are straight-forward and can be implemented with guidance from the Javadoc.

Export to OSGi Service Registry

In order for the `ResourceReader` to be used by the `CatalogFramework`, it should be exported to the OSGi Service Registry as a `DDF.catalog.resource.ResourceReader`.

See the XML below for an example:

Blueprint example

```
<bean id="[[customResourceReaderId]]" class=
"[[example.resource.reader.impl.CustomResourceReader]]" />
<service ref="[[customResourceReaderId]]" interface="
DDF.catalog.source.ResourceReader" />
```

33.4. Resource Writers

A resource writer stores a resource and produces a URI that can be used to retrieve the resource at a later time. The resource URI uniquely locates and identifies the resource. Resource writers can interact with an underlying data store and store the resource in the proper place. Each implementation can do this differently, providing flexibility in the data stores used to persist the resources.

Resource Writers should be used within the Content Framework if and when implementing a custom Storage Provider to store the product. The default Storage Provider that comes with the DDF writes the products to the file system.

Included Resource Writers

The Catalog reference implementation currently does not include any resource writers.

33.4.1. Developing Resource Writers

A `ResourceWriter` is an object used to store or delete a `Resource`. `ResourceWriter` objects should be registered within the OSGi Service Registry, so clients can retrieve an instance when they need to store a `Resource`.

Create a New `ResourceWriter`

Complete the following procedure to create a `ResourceWriter`.

1. Create a Java class that implements the `DDF.catalog.resource.ResourceWriter` interface.

ResourceWriter Implementation Skeleton

```
import java.io.IOException;
import java.net.URI;
import java.util.Map;
import DDF.catalog.resource.Resource;
import DDF.catalog.resource.ResourceNotFoundException;
import DDF.catalog.resource.ResourceNotSupportedException;
import DDF.catalog.resource.ResourceWriter;

public class SampleResourceWriter implements ResourceWriter {

    @Override
    public void deleteResource(URI uri, Map<String, Object> arguments) throws
ResourceNotFoundException, IOException {
        // WRITE IMPLEMENTATION
    }

    @Override
    public URI storeResource(Resource resource, Map<String, Object> arguments) throws
ResourceNotSupportedException, IOException {
        // WRITE IMPLEMENTATION
        return null;
    }

    @Override
    public URI storeResource(Resource resource, String id, Map<String, Object>
arguments) throws ResourceNotSupportedException, IOException {
        // WRITE IMPLEMENTATION
        return null;
    }

}
```

1. Register the implementation as a Service in the OSGi Service Registry.

Blueprint Service Registration Example

```
...
<service ref="[[ResourceWriterReference]]" interface=
"DDF.catalog.resource.ResourceWriter" />
...
```

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

ResourceWriter Javadoc

TIP Refer to the Catalog API Javadoc for more information about the methods required for implementing the interface.

33.5. Registry Clients

Registry Clients create Federated Sources using the OSGi Configuration Admin. Developers should reference an individual [Source's \(Federated, Connected, or Catalog Provider\) documentation for the Configuration properties](#) (such as a Factory PID, addresses, intervals, etc) necessary to establish that 'Source' in the framework.

33.5.1. Developing Registry Clients

Creating a Source Configuration

```
org.osgi.service.cm.ConfigurationAdmin configurationAdmin = getConfigurationAdmin() ;  
org.osgi.service.cm.Configuration currentConfiguration = configurationAdmin  
.createFactoryConfiguration(getFactoryPid(), null);  
Dictionary properties = new Dictionary();  
properties.put(QUERY_ADDRESS_PROPERTY,queryAddress);  
currentConfiguration.update( properties );
```

Note that the `QUERY_ADDRESS_PROPERTY` is specific to this Configuration and might not be required for every [Source](#). The properties necessary for creating a Configuration are different for every [Source](#).

33.6. Directory Monitors

33.6.1. Content Directory Monitor (CDM)

The Content Directory Monitor (CDM) provides the capability to easily add content and metacards into the Catalog by placing a file in a directory.

Installing the Content Directory Monitor

The Content Directory Monitor is installed by default with a standard installation of the Catalog application.

Configuring the Content Directory Monitor

Configure the CDM from the Admin Console:

1. Navigate to the [Admin Console](#).
2. Select the [Catalog](#) application.
3. Select the [Configuration](#) tab.
4. Select [Catalog Content Directory Monitor](#).

Table 130. Catalog Content Directory Monitor

Name	Id	Type	Description	Default Value	Required
Directory Path	<code>monitoredDirectoryPath</code>	String	Specifies the directory to be monitored.	false	true
Maximum Concurrent Files	<code>numThreads</code>	Integer	Specifies the maximum number of concurrent files to be processed within a directory (maximum of 8). If this number exceeds 8, 8 will be used in order to preserve system resources. Make sure that your system has enough memory to support the number of concurrent processing threads across all directory monitors.	1	true
ReadLock Time Interval	<code>readLockIntervalMilliseconds</code>	Integer	Specifies the time to wait (in milliseconds) before acquiring a lock on a file in the monitored directory. This interval is used for sleeping between attempts to acquire the read lock on a file to be ingested. The default value of 100 milliseconds is recommended.	100	true
Processing Mechanism	<code>processingMechanism</code>	String	Choose what happens to the content item after it is ingested. Delete will remove the original file after storing it in the content store. Move will store the item in the content store, and a copy under ./ingested, then remove the original file. (NOTE: this will double the amount of disk space used.) Monitor in place will index the file and serve it from its original location.	in_place	false
Attribute Overrides	<code>attributeOverrides</code>	String	Optional: Metocard attribute overrides (Key-Value pairs) that can be set on the content monitor. If an attribute is specified here, it will overwrite the metocard's attribute that was created from the content directory. The format should be 'key=value'.	null	false

Using the Content Directory Monitor

The CDM processes files in a directory, and all of its sub-directories. The CDM offers three options:

- Delete
- Move

- Monitor in place

Regardless of the option, the DDF takes each file in a monitored directory structure and creates a metocard for it. The metocard is linked to the file. The behavior of each option is given below.

- **Delete**

- Copies the file into the Content Repository.
- Creates a metocard in the Catalog from the file.
- *Erases the original file from the monitored directory.

- **Move**

- **Copies the file into the directory .\ingested (this will double the disk space used)**
- Copies the file into the Content Repository.
- Creates a metocard in the Catalog from the file.
- *Erases the original file from the monitored directory.

- **Monitor in place**

- Creates a metocard in the Catalog from the file.
- Creates a reference from the metocard to the original file in the monitored directory.
- If the original file is deleted, the metocard is removed from the Catalog.
- If the original file is modified, the metocard is updated to reflect the new content.
- If the original file is renamed, the old metocard is deleted and a new metocard is created.

Parallel Processing

The CDM supports parallel processing of files (up to 8 files processed concurrently). This is configured by setting the number of **Maximum Concurrent Files** in the configuration. A maximum of 8 is imposed to protect system resources.

Read Lock

When the CDM is set up, the directory specified is continuously scanned, and files are locked for processing based on the **ReadLock Time Interval**. This does not apply to the **Monitor in place** processing directive. Files will not be ingested without having a ReadLock that has observed no change in the file size. This is done so that files that are in transit will not be ingested prematurely. The interval should be dependent on the speed of the copy to the directory monitor (ex. network drive vs local disk). For local files, the default value of 500 milliseconds is recommended. The recommended interval for network drives is 1000 - 2000 milliseconds. If the value provided is less than 100, 100 milliseconds will be used. It is also recommended that the **ReadLock Time Interval** be set to a lower amount of time when the **Maximum Concurrent Files** is set above 1 so that files are locked in a timely manner and processed as soon as possible. When a higher **ReadLock Time Interval** is set, the time it takes for files to be processed is increased.

Attribute Mapper

The CDM supports setting metocard attributes directly when DDF ingests a file. Custom mappings are entered in the form:

attribute-name=attribute-value

For example, to set the contact email for all metacards, add the custom mapping:

contact.point-of-contact-email=doctor@clinic.com

Each mapping sets the value of a single metocard attribute. To set the value of an additional attribute, select the "plus" icon in the UI. This creates an empty line for the entry.

To set multi-valued attributes, use a separate override for each value. For example, to add the keywords *PPI* and *radiology* to each metocard, add the custom attribute mappings:

topic.keyword=PPI
topic.keyword=radiology

Errors

If the CDM fails to read the file, an error will be logged in the ingest log. If the directory monitor is configured to **Delete** or **Move**, the original file is also moved to the **\.errors** directory.

Other

- Multiple directories can be monitored. Each directory has an independent configuration.
- To support the monitoring in place behavior, DDF indexes the files to track their names and modification timestamps. This enables the Content Directory Monitor to take appropriate action when files are changed or deleted.
- The Content Directory Monitor recursively processes all subdirectories.

34. Queries

Clients use **ddf.catalog.operation.Query** objects to describe which metacards are needed from [Sources](#).

Query objects have two major components:

- [Filters](#)
- [Query Options](#)

A Source uses the Filter criteria constraints to find the requested set of metacards within its domain of metacards. The Query Options are used to further restrict the Filter's set of requested metacards.

34.1. Filters

An OGC Filter is a [Open Geospatial Consortium \(OGC\) standard](#) that describes a query expression in terms of Extensible Markup Language (XML) and key-value pairs (KVP). The OGC Filter is used to represent a query to be sent to sources and the Catalog Provider, as well as to represent a Subscription. The OGC Filter provides support for expression processing, such as adding or dividing expressions in a query, but that is not the intended use for DDF.

The Catalog Framework does not use the XML representation of the OGC Filter

standard. DDF instead uses the Java implementation provided by [GeoTools](#). GeoTools provides Java equivalent classes for OGC Filter XML elements. GeoTools originally provided the standard Java classes for the OGC Filter Encoding 1.0 under the package name `org.opengis.filter`. The same package name is used today and is currently used by DDF. Java developers do not parse or view the XML representation of a Filter in DDF. Instead, developers use only the Java objects to complete query tasks.

Note that the `ddf.catalog.operation.Query` interface extends the `org.opengis.filter.Filter` interface, which means that a Query object is an OGC Java Filter with Query Options.

A Query is an OGC Filter

```
public interface Query extends Filter
```

34.2. FilterBuilder API

To avoid the complexities of working with the Filter interface directly and implementing the DDF Profile of the Filter specification, the Catalog includes an API, primarily in [DDF.filter](#), to build Filters using a fluent API.

To use the `FilterBuilder` API, an instance of `DDF.filter.FilterBuilder` should be used via the OSGI registry. Typically, this will be injected via a dependency injection framework. Once an instance of `FilterBuilder` is available, methods can be called to create and combine Filters.

TIP

The fluent API is best accessed using an IDE that supports code-completion. For additional details, refer to the [Catalog API Javadoc].

34.2.1. Boolean Operators

Filters use a number of boolean operators.

`FilterBuilder.allOf(Filter ...)`

creates a new Filter that requires all provided Filters are satisfied (Boolean AND), either from a List or Array of Filter instances.

`FilterBuilder.anyOf(Filter ...)`

creates a new Filter that requires at least one of the provided Filters are satisfied (Boolean OR), either from a List or Array of Filter instances.

`FilterBuilder.not(Filter filter)`

creates a new Filter that requires the provided Filter must not match (Boolean NOT).

34.2.2. Attribute

Filters can be based on specific attributes.

`FilterBuilder.attribute(String attributeName)`

begins a fluent API for creating an Attribute-based Filter, i.e., a Filter that matches on Metacards with Attributes of a particular value.

34.2.3. XPath

Filters can be based on XML attributes.

`FilterBuilder.xpath(String xpath)`

begins a fluent API for creating an XPath-based Filter, i.e., a Filter that matches on Metacards with Attributes of type XML that match when evaluating a provided XPath selector.

Contextual Operators

```
FilterBuilder.attribute(attributeName).is().like().text(String contextualSearchPhrase);  
FilterBuilder.attribute(attributeName).is().like().caseSensitiveText(String caseSensitiveContextualSearchPhrase);  
FilterBuilder.attribute(attributeName).is().like().fuzzyText(String fuzzySearchPhrase);
```

34.3. FilterDelegates

Filter Delegates help reduce the complexity of parsing OGC Filters. The reference Filter Adapter implementation contains the necessary boilerplate visitor code and input normalization to handle commonly supported OGC Filters.

34.3.1. Creating a New Filter Delegate

A Filter Delegate contains the logic that converts normalized filter input into a form that the target data source can handle. Delegate methods will be called in a depth first order as the Filter Adapter visits filter nodes.

Implementing the Filter Delegate

1. Create a Java class extending `FilterDelegate`.

```
public class ExampleDelegate extends DDF.catalog.filter.FilterDelegate<ExampleReturnObjectType> {
```

2. `FilterDelegate` will throw an appropriate exception for all methods not implemented. Refer to the DDF JavaDoc for more details about what is expected of each `FilterDelegate` method.

NOTE

A code example of a Filter Delegate can be found in `DDF.catalog.filter.proxy.adapter.test` of the `filter-proxy` bundle.

Throwing Exceptions

Filter delegate methods can throw `UnsupportedOperationException` run-time exceptions. The `GeotoolsFilterAdapterImpl` will catch and re-throw these exceptions as `UnsupportedQueryExceptions`.

Using the Filter Adapter

The FilterAdapter can be requested from the OSGi registry.

```
<reference id="filterAdapter" interface="DDF.catalog.filter.FilterAdapter" />
```

The Query in a QueryRequest implements the Filter interface. The Query can be passed to a **FilterAdapter** and **FilterDelegate** to process the Filter.

```
@Override  
public DDF.catalog.operation.QueryResponse query(DDF.catalog.operation.QueryRequest  
queryRequest)  
throws DDF.catalog.source.UnsupportedQueryException {  
  
    DDF.catalog.operation.Query query = queryRequest.getQuery();  
  
    DDF.catalog.filter.FilterDelegate<ExampleReturnObjectType> delegate = new  
    ExampleDelegate();  
  
    // DDF.catalog.filter.FilterAdapter adapter injected via Blueprint  
    ExampleReturnObjectType result = adapter.adapt(query, delegate);  
}
```

Import the Catalog API Filter package and the reference implementation package of the Filter Adapter in the bundle manifest (in addition to any other required packages).

Import-Package: `DDF.catalog, DDF.catalog.filter, DDF.catalog.source`

Filter Support

Not all OGC Filters are exposed at this time. If demand for further OGC Filter functionality is requested, it can be added to the Filter Adapter and Delegate so sources can support more complex filters. The following OGC Filter types are currently available:

Logical

And

Or

Not

Include

Exclude

Property Comparison

`PropertyIsBetween`

`PropertyIsEqualTo`

`PropertyIsGreater Than`

`PropertyIsGreater ThanOrEqual To`

`PropertyIsLessThan`

Property Comparison

[PropertyIsLessThanOrEqualTo](#)

[PropertyIsLike](#)

[PropertyIsNotEqualTo](#)

[PropertyIsNull](#)

Spatial	Definition
Beyond	True if the geometry being tested is beyond the stated distance of the geometry provided.
Contains	True if the second geometry is wholly inside the first geometry.
Crosses	True if: * the intersection of the two geometries results in a value whose dimension is less than the geometries * the maximum dimension of the intersection value includes points interior to both the geometries * the intersection value is not equal to either of the geometries.
Disjoint	True if the two geometries do not touch or intersect.
DWithin	True if the geometry being tested is within the stated distance of the geometry provided.
Intersects	True if the two geometries intersect. This is a convenience method as Not Disjoint(A,B) gets the same result.
Overlaps	True if the intersection of the geometries results in a value of the same dimension as the geometries that is different from both of the geometries.
Touches	True if and only if the only common points of the two geometries are in the union of the boundaries of the geometries.
Within	True if the first geometry is wholly inside the second geometry.

Temporal

[After](#)

[Before](#)

[During](#)

34.4. Developing Filters

34.4.1. Directly Implementing the Filter (Advanced)

The common way to create a [Filter](#) is to use the GeoTools [FilterFactoryImpl](#) object, which provides Java implementations for the various types of filters in the Filter Specification. Examples are the easiest way to understand how to properly create a [Filter](#) and a [Query](#).

NOTE Refer to the [GeoTools javadoc](#) for more information on `FilterFactoryImpl`.

WARNING Implementing the Filter interface directly is only for extremely advanced use cases and is highly discouraged. Instead, use of the DDF-specific `FilterBuilder` API is recommended.

Developers create a `Filter` object in order to filter or constrain the amount of records returned from a `Source`. The OGC Filter Specification has several types of filters that can be combined in a tree-like structure to describe the set of metacards that should be returned.

Categories of Filters

- Comparison Operators
- Logical Operators
- Expressions
- Literals
- Functions
- Spatial Operators
- Temporal Operators

34.4.2. Units of Measure

According to the [OGC Filter Specifications: 09-026r1](#) and [OGC Filter Specifications: 04-095](#), units of measure can be expressed as a URI. To fulfill that requirement, DDF utilizes the GeoTools class `org.geotools.styling.UomOgcMapping` for spatial filters requiring a standard for units of measure for scalar distances. Essentially, the `UomOgcMapping` maps the [OGC Symbology Encoding](#) standard URIs to Java Units. This class provides three options for units of measure:

- FOOT
- METRE
- PIXEL

DDF only supports FOOT and METRE since they are the most applicable to scalar distances.

34.4.3. Filter Examples

The example below illustrates creating a query, and thus an OGC Filter, that does a case-insensitive search for the phrase "mission" in the entire metocard's text. Note that the OGC `PropertyIsLike` Filter is used for this simple contextual query.

Simple Contextual Search

```
org.opengis.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;
boolean isCaseSensitive = false ;

String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard, singleChar,
and the escapeChar itself

String searchPhrase = "mission" ;
org.opengis.filter.Filter propertyIsLikeFilter =
    filterFactory.like(filterFactory.property(Metacard.ANY_TEXT), searchPhrase,
wildcardChar, singleChar, escapeChar, isCaseSensitive);
DDF.catalog.operation.QueryImpl query = new QueryImpl( propertyIsLikeFilter );
```

The example below illustrates creating an absolute temporal query, meaning the query is searching for Metacards whose modified timestamp occurred during a specific time range. Note that this query uses the **During** OGC Filter for an absolute temporal query.

Absolute Temporal Search

```
org.opengis.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;
org.opengis.temporal.Instant startInstant = new org.geotools.temporal.object
.DefaultInstant(new DefaultPosition(start));

org.opengis.temporal.Instant endInstant = new org.geotools.temporal.object
.DefaultInstant(new DefaultPosition(end));

org.opengis.temporal.Period period = new org.geotools.temporal.object.DefaultPeriod
(startInstant, endInstant);

String property = Metacard.MODIFIED ; // modified date of a metacard

org.opengis.filter.Filter filter = filterFactory.during( filterFactory.property
(property), filterFactory.literal(period) ) ;

DDF.catalog.operation.QueryImpl query = new QueryImpl(filter) ;
```

Contextual Searches

Most contextual searches can be expressed using the **PropertyIsLike** filter. The special characters that have meaning in a **PropertyIsLike** filter are the wildcard, single wildcard, and escape characters (see Example Creating-Filters-1).

Table 131. **PropertyIsLike** Special Characters

Character	Description
Wildcard	Matches zero or more characters.

Character	Description
Single Wildcard	Matches exactly one character.
Escape	Escapes the meaning of the Wildcard, Single Wildcard, and the Escape character itself

Characters and words, such as `AND`, `&`, `and`, `OR`, `|`, `or`, `NOT`, `~`, `not`, `{`, and `}`, are treated as literals in a `PropertyIsLike` filter. In order to create equivalent logical queries, a developer must instead use the Logical Operator filters `{AND, OR, NOT}`. The Logical Operator filters can be combined together with `PropertyIsLike` filters to create a tree that represents the search phrase expression.

Creating the search phrase "mission and planning"

```
org.opengis.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;

boolean isCaseSensitive = false ;

String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard, singleChar,
and the escapeChar itself

Filter filter =
    filterFactory.and(
        filterFactory.like(filterFactory.property(Metacard.METADATA), "mission" ,
wildcardChar, singleChar, escapeChar, isCaseSensitive),
        filterFactory.like(filterFactory.property(Metacard.METADATA), "planning" ,
wildcardChar, singleChar, escapeChar, isCaseSensitive)
    );

DDF.catalog.operation.QueryImpl query = new QueryImpl( filter );
```

Tree View of Creating Filters

Filters used in DDF can always be represented in a tree diagram.

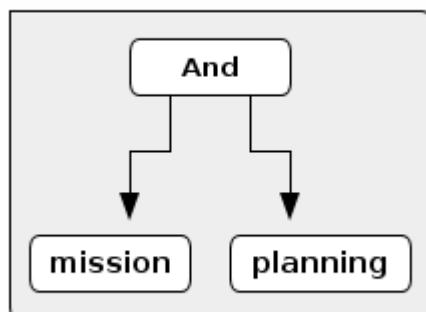


Figure 42. Filter Example Tree Diagram

XML View of Creating Filters

Another way to view this type of Filter is through an XML model, which is shown below.

Pseudo XML of Example Creating-Filters-3

```
<Filter>
  <And>
    <PropertyIsLike wildCard="*" singleChar="?" escapeChar="\">
      <PropertyName>metadata</PropertyName>
      <Literal>mission</Literal>
    </PropertyIsLike>
    <PropertyIsLike wildCard="*" singleChar="?" escapeChar="\">
      <PropertyName>metadata</PropertyName>
      <Literal>planning</Literal>
    </PropertyIsLike>
  <And>
</Filter>
```

Using the Logical Operators and **PropertyIsLike** filters, a developer can create a whole language of search phrase expressions.

Fuzzy Operations

DDF only supports one custom function. The Filter specification does not include a fuzzy operator, so a Filter function was created to represent a fuzzy operation. The function and class is called **FuzzyFunction**, which is used by clients to notify the Sources to perform a fuzzy search. The syntax expected by providers is similar to the Fuzzy Function. Refer to the example below.

```
String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard, singleChar

boolean isCaseSensitive = false ;

Filter fuzzyFilter = filterFactory.like(
  new DDF.catalog.impl.filter.FuzzyFunction(
    Arrays.asList((Expression) (filterFactory.property(Metocard.ANY_TEXT))),
    filterFactory.literal("")),
  searchPhrase,
  wildcardChar,
  singleChar,
  escapeChar,
  isCaseSensitive);

QueryImpl query = new QueryImpl(fuzzyFilter);
```

34.4.4. Parsing Filters

According to the [OGC Filter Specification 04-095](#): a "(filter expression) representation can be ... parsed and then transformed into whatever target language is required to retrieve or modify object instances stored in some persistent object store." Filters can be thought of as the **WHERE** clause for a SQL SELECT statement to "fetch data stored in a SQL-based relational database."

Sources can parse OGC Filters using the **FilterAdapter** and **FilterDelegate**. See Developing a Filter Delegate for more details on implementing a new **FilterDelegate**. This is the preferred way to handle OGC Filters in a consistent manner.

Alternately, `org.opengis.filter.Filter` implementations can be parsed using implementations of the `org.opengis.filter.FilterVisitor` interface. The `FilterVisitor` uses the [http://www.oodesign.com/visitor-pattern.html\[Visitor pattern\]](http://www.oodesign.com/visitor-pattern.html). Essentially, `FilterVisitor` instances "visit" each part of the `Filter` tree allowing developers to implement logic to handle the filter's operations. GeoTools 8 includes implementations of the `FilterVisitor` interface. The `DefaultFilterVisitor`, as an example, provides only business logic to visit every node in the `Filter` tree. The `DefaultFilterVisitor` methods are meant to be overwritten with the correct business logic. The simplest approach when using `FilterVisitor` instances is to build the appropriate query syntax for a target language as each part of the `Filter` is visited. For instance, when given an incoming `Filter` object to be evaluated against a RDBMS, a `CatalogProvider` instance could use a `'FilterVisitor'` to interpret each filter operation on the `Filter` object and translate those operations into SQL. The `FilterVisitor` may be needed to support `Filter` functionality not currently handled by the `FilterAdapter` and `FilterDelegate` reference implementation.

Interpreting a Filter to Create SQL

If the `FilterAdapter` encountered or "visited" a `PropertyIsLike` filter with its property assigned as `title` and its literal expression assigned as `mission`, the `FilterDelegate` could create the proper SQL syntax similar to title `LIKE mission`.

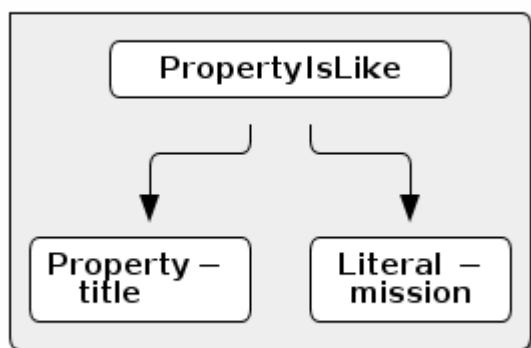


Figure 43. Parsing Filters Tree Diagram

Interpreting a Filter to Create XQuery

If the `FilterAdapter` encountered an `OR` filter, such as in Figure Parsing-Filters2 and the target language was XQuery, the `FilterDelegate` could yield an expression such as

```
ft:query(/inventory:book/@subject,'math') union
ft:query(/inventory:book/@subject,'science').
```

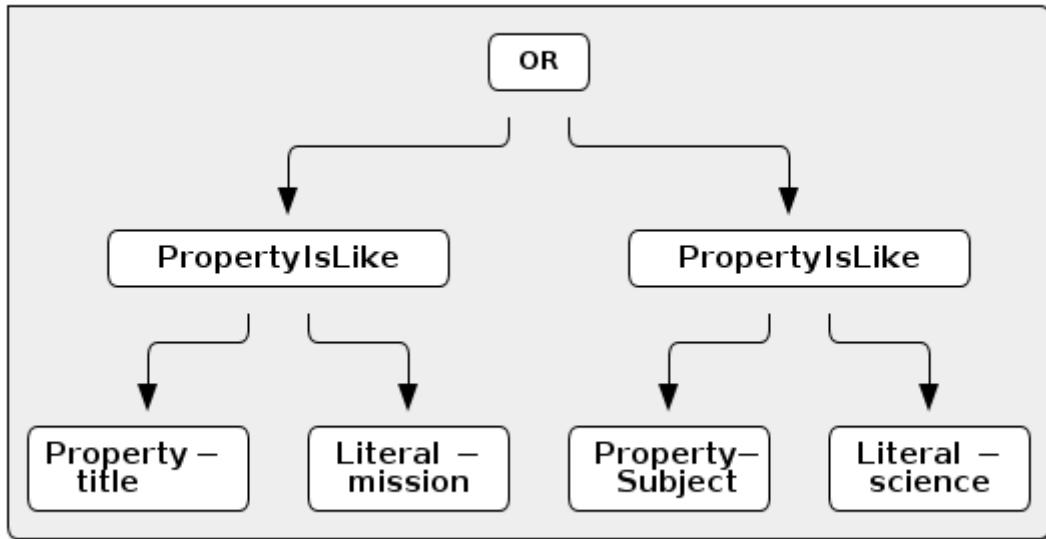


Figure 44. Parsing Filters XQuery

FilterAdapter/Delegate Process for Figure Parsing

1. **FilterAdapter** visits the **OR** filter first.
2. **OR** filter visits its children in a loop.
3. The first child in the loop that is encountered is the LHS **PropertyIsLike**.
4. The **FilterAdapter** will call the **FilterDelegate`PropertyIsLike`** method with the LHS property and literal.
5. The LHS **PropertyIsLike** delegate method builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject`. Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
6. The **FilterAdapter** then moves back to the **OR** filter, which visits its second child.
7. The **FilterAdapter** will call the **FilterDelegate PropertyIsLike** method with the RHS property and literal.
8. The RHS **PropertyIsLike** delegate method builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject`. Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
9. The **FilterAdapter** then moves back to its `OR Filter which is now done with its children.
10. It then collects the output of each child and sends the list of results to the **FilterDelegate**

OR method.

11. The final result object will be returned from the **FilterAdapter adapt** method.

FilterVisitor Process for Figure Parsing

1. FilterVisitor visits the **OR** filter first.
2. **OR** filter visits its children in a loop.
3. The first child in the loop that is encountered is the LHS **PropertyIsLike**.
4. The LHS **PropertyIsLike** builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject`. Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
5. The FilterVisitor then moves back to the **OR** filter, which visits its second child.
6. The RHS **PropertyIsLike** builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject`. Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
7. The FilterVisitor then moves back to its **OR** filter, which is now done with its children. It then collects the output of each child and could potentially execute the following code to produce the above expression.

```
public visit( Or filter, Object data) {  
    ...  
    /* the equivalent statement for the OR filter in this domain (XQuery) */  
    xQuery = childFilter1Output + " union " + childFilter2Output;  
    ...  
}
```

34.4.5. Filter Profile

The filter profile maps filters to metocard types.

Role of the OGC Filter

Both Queries and Subscriptions extend the OGC GeoAPI Filter interface.

The Filter Builder and Adapter do not fully implement the OGC Filter Specification. The filter support profile contains suggested filter to metocard type mappings. For example, even though a Source could support a **PropertyIsGreater Than** filter on `XML_TYPE`, it would not likely be useful.

Catalog Filter Profile

The following table displays the common metocard attributes with their respective types for reference.

Table 132. Metocard Attribute To Type Mapping

Metocard Attribute	Metocard Type
ANY_DATE	DATE_TYPE
ANY_GEO	GEO_TYPE
ANY_TEXT	STRING_TYPE
CONTENT_TYPE	STRING_TYPE
CONTENT_TYPE_VERSION	STRING_TYPE
CREATED	DATE_TYPE
EFFECTIVE	DATE_TYPE
GEOGRAPHY	GEO_TYPE
ID	STRING_TYPE
METADATA	XML_TYPE
MODIFIED	DATE_TYPE
RESOURCE_SIZE	STRING_TYPE
RESOURCE_URI	STRING_TYPE
SOURCE_ID	STRING_TYPE
TARGET_NAMESPACE	STRING_TYPE
THUMBNAIL	BINARY_TYPE
TITLE	STRING_TYPE

Comparison Operators

Comparison operators compare the value associated with a property name with a given Literal value. Endpoints and sources should try to use metocard types other than the object type. The object type only supports backwards compatibility with `java.net.URI`. Endpoints that send other objects will not be supported by standard sources. The following table maps the metocard types to supported comparison operators.

Table 133. Metocard Types to Comparison Operators

PropertyIs	Between	EqualTo	GreaterThan	GreaterThanOrEqual	OrEqualTo	LessThan	LessThanOrEqual	OrEqualTo	Like	NotEqualTo	Null
BINARY_TYPE		X									
BOOLEAN_TYPE		X									

PropertyIs	Between	Equal To	Greater Than	Greater Than	OrEqualTo	LessThan	LessThan	OrEqualTo	Like	NotEqualTo	Null
DATE_TYPE	X	X	X	X	X	X	X	X		X	X
DOUBLE_TYPE	X	X	X	X	X	X	X	X		X	X
FLOAT_TYPE	X	X	X	X	X	X	X	X		X	X
GEO_TYPE											X
INTEGER_TYPE	X	X	X	X	X	X	X	X		X	X
LONG_TYPE	X	X	X	X	X	X	X	X		X	X
OBJECT_TYPE	X	X	X	X	X	X	X	X		X	X
SHORT_TYPE	X	X	X	X	X	X	X	X		X	X
STRING_TYPE	X	X	X	X	X	X	X	X	X	X	X
XML_TYPE		X							X		X

Table 134. Comparison Operators

Operator	Description
PropertyIsBetween	Lower ≤ Property ≤ Upper
PropertyIsEqualTo	Property == Literal
PropertyIsGreaterThan	Property > Literal
PropertyIsGreaterThanOrEqualTo	Property >= Literal
PropertyIsLessThan	Property < Literal
PropertyIsLessThanOrEqualTo	Property ≤ Literal
PropertyIsLike	Property LIKE Literal Equivalent to SQL "like"
PropertyIsNotEqualTo	Property != Literal
PropertyIsNull	Property == null

Logical Operators

Logical operators apply Boolean logic to one or more child filters.

Table 135. Supported Logical Operators

	And	Not	Or
Supported Filters	X	X	X

Temporal Operators

Temporal operators compare a date associated with a property name to a given Literal date or date range.

Table 136. Supported Temporal Operators

	After	AnyInteracts	Before	Begins	BegunBy	During	EndedBy	Meets	MetBy	OverlappedBy	TContains
DATE_TYPE	X		X			X					

Literal values can be either date instants or date periods.

Table 137. Temporal Operator Descriptions

Operator	Description
After	Property > (Literal Literal.end)
Before	Property < (Literal Literal.start)
During	Literal.start < Property < Literal.end

Spatial Operators

Spatial operators compare a geometry associated with a property name to a given Literal geometry.

Table 138. Supported Spatial Operators.

BBox	Beyond	Contains	Crosses	Disjoint	Equals	DWithin	Intersects	Overlaps	Touches	Within
GEO_TYPE		X	X	X	X		X	X	X	

Geometries are usually represented as Well-Known Text (WKT).

Table 139. Spatial Operator Descriptions

Operator	Description
Beyond	Property geometries beyond given distance of Literal geometry

Operator	Description
Contains	Property geometry contains Literal geometry
Crosses	Property geometry crosses Literal geometry
Disjoint	Property geometry direct positions are not interior to Literal geometry
DWithin	Property geometry lies within distance to Literal geometry
Intersects	Property geometry intersects Literal geometry; opposite to the Disjoint operator
Overlaps	Property geometry interior overlaps Literal geometry interior somewhere
Touches	Property geometry touches but does not overlap Literal geometry
Within	Property geometry completely contains Literal geometry

34.5. Query Options

The easiest way to create a Query is to use the `ddf.catalog.operation.QueryImpl` object. It is first necessary to create an OGC Filter object then set the Query Options after `QueryImpl` has been constructed.

`QueryImpl` Example

```
/*
Builds a query that requests a total results count and
that the first record to be returned is the second record found from
the requested set of metacards.
*/
String property = ...;

String value = ...;

org.geotools.filter.FilterFactoryImpl filterFactory = new FilterFactoryImpl() ;

QueryImpl query = new QueryImpl( filterFactory.equals(filterFactory.property(
property),
filterFactory.literal(value))) ;

query.setstartIndex(2) ;

query.setRequestsTotalResultsCount(true);
```

34.5.1. Evaluating a query

Every Source must be able to evaluate a Query object. Nevertheless, each Source could evaluate the Query differently depending on what that Source supports as to properties and query capabilities. For instance, a common property all Sources understand is `id`, but a Source could possibly store frequency values under the property name "frequency." Some Sources may not support frequency property inquiries and will throw an error stating it cannot interpret the property. In addition, some Sources might be able to handle spatial operations, while others might not. A developer should consult a Source's documentation for the limitations, capabilities, and properties that a Source can support.

Table 140. Query Options

Option	Description
<code>StartIndex</code>	1-based index that states which metocard the Source should return first out of the requested metacards.
<code>PageSize</code>	Represents the maximum amount of metacards the Source should return.
<code>SortBy</code>	Determines how the results are sorted and on which property.
<code>RequestsTotalResultsCount</code>	Determines whether the total number of results should be returned.
<code>TimeoutMillis</code>	The amount of time in milliseconds before the query is to be abandoned. If a zero or negative timeout is set, the catalog framework will default to a value configurable via the Admin UI under Catalog → Configuration → Query Operations.

34.5.2. Commons-DDF Utilities

The `commons-DDF` bundle provides utilities and functionality commonly used across other DDF components, such as the endpoints and providers.

FuzzyFunction

`DDF.catalog.impl.filter.FuzzyFunction` class is used to indicate that a `PropertyIsLike` filter should interpret the search as a fuzzy query.

XPathHelper

`DDF.util.XPathHelper` provides convenience methods for executing XPath operations on XML. It also provides convenience methods for converting XML as a `String` from a `org.w3c.dom.Document` object and vice versa.

35. Security Framework

The DDF Security Framework utilizes [Apache Shiro](#) as the underlying security framework. The classes mentioned in this section will have their full package name listed, to make it easy to tell which classes come with the core Shiro framework and which are added by DDF.

35.1. Subject

`ddf.security.Subject <extends> org.apache.shiro.subject.Subject`

The Subject is the key object in the security framework. Most of the workflow and implementations revolve around creating and using a Subject. The Subject object in DDF is a class that encapsulates all information about the user performing the current operation. The Subject can also be used to perform permission checks to see if the calling user has acceptable permission to perform a certain action (e.g., calling a service or returning a metocard). This class was made DDF-specific because the Shiro interface cannot be added to the Query Request property map.

Table 141. Implementations of Subject:

Classname	Description
<code>ddf.security.impl.SubjectImpl</code>	Extends <code>org.apache.shiro.subject.support.DelegatingSubject</code>

35.1.1. Security Manager

`ddf.security.service.SecurityManager`

The Security Manager is a service that handles the creation of Subject objects. A proxy to this service should be obtained by an endpoint to create a Subject and add it to the outgoing `QueryRequest`. The Shiro framework relies on creating the subject by obtaining it from the current thread. Due to the multi-threaded and stateless nature of the DDF framework, utilizing the Security Manager interface makes retrieving Subjects easier and safer.

Table 142. Implementations of Security Managers:

Classname	Description
<code>ddf.security.service.SecurityManagerImpl</code>	This implementation of the Security Manager handles taking in both <code>org.apache.shiro.authc.AuthenticationToken</code> and <code>org.apache.cxf.ws.security.tokenstore.SecurityToken</code> objects.

35.1.2. Authentication Tokens

`org.apache.shiro.authc.AuthenticationToken`

Authentication Tokens are used to verify authentication of a user when creating a subject. A common use-case is when a user is logging directly in to the DDF framework.

Classname	Description
<code>ddf.security.service.impl.cas.CasAuthenticationToken</code>	This Authentication Token is used for authenticating a user that has logged in with CAS. It takes in a proxy ticket which can be validated on the CAS server.

35.1.3. Realms

DDF uses Realms for Authentication and Authorization.

Authenticating Realms

`org.apache.shiro.realm.AuthenticatingRealm`

Authenticating Realms are used to authenticate an incoming authentication token and create a Subject on successful authentication.

Table 143. Implementations of Authenticating Realms in DDF:

Classname	Description
<code>ddf.security.realm.sts.StsRealm</code>	This realm delegates authentication to the Secure Token Service (STS). It creates a <code>RequestSecurityToken</code> message from the incoming Authentication Token and converts a successful STS response into a Subject.

Authorizing Realms

`org.apache.shiro.realm.AuthorizingRealm`

Authorizing Realms are used to perform authorization on the current Subject. These are used when performing both Service AuthZ and Filtering. They are passed in the `AuthorizationInfo` of the Subject along with the Permissions of the object wanting to be accessed. The response from these realms is a true (if the Subject has permission to access) or false (if the Subject does not).

Table 144. Other implementations of the Security API within DDF

Classname	Description
<code>org.codice.ddf.platform.filter.delegate.DelegateServletFilter</code>	The <code>DelegateServletFilter</code> detects any servlet filters that have been exposed as OSGi services and places them in-order in front of any servlet or web application running on the container.
<code>org.codice.ddf.security.filter.websso.WebSSOFilter</code>	This filter serves as the main security filter that works in conjunction with a number of handlers to protect a variety of contexts, each using different authentication schemes and policies.

Classname	Description
<code>org.codice.ddf.security.handler.saml.SAMLAssertionHandler</code>	This handler is executed by the WebSSOFilter for any contexts configured to use it. This handler should always come first when configured in the Web Context Policy Manager, as it provides a caching capability to web contexts that use it. The handler will first check for the existence of an HTTP Authorization header of type SAML, whose value is a Base64 + deflate SAML assertion. If that is not found, then the handler will check for the existence of the deprecated <code>org.codice.websso.saml.token</code> cookie with the same value. Failing that, it will check for a JSESSIONID cookie to use as a reference to a cached assertion. If the JSESSIONID is valid, the <code>SecurityToken</code> will be retrieved from the cache.
<code>org.codice.ddf.security.handler.basic.BasicAuthenticationHandler</code>	Checks for basic authentication credentials in the http request header. If they exist, they are retrieved and passed to the <code>LoginFilter</code> for exchange.
<code>org.codice.ddf.security.handler.pki.PKIHandler</code>	Handler for PKI based authentication. X509 chain will be extracted from the HTTP request and converted to a <code>BinarySecurityToken</code> .
<code>org.codice.ddf.security.handler.guest.GuestHandler</code>	Handler that allows guest user access via a guest user account. The guest account credentials are configured via the <code>org.codice.ddf.security.claims.guest.GuestClaimsHandler</code> . The <code>GuestHandler</code> also checks for the existence of basic auth credentials or PKI credentials that might be able to override the use of the guest user.
<code>org.codice.ddf.security.filter.login.LoginFilter</code>	This filter runs immediately after the WebSSOFilter and exchanges any authentication information found in the request with a Subject via Shiro.
<code>org.codice.ddf.security.filter.authorization.AuthorizationFilter</code>	This filter runs immediately after the <code>LoginFilter</code> and checks any permissions assigned to the web context against the attributes of the user via Shiro.
<code>org.apache.shiro.realm.AuthenticatingRealm</code>	This is an abstract authenticating realm that exchanges an <code>org.apache.shiro.authc.AuthenticationToken</code> for a <code>ddf.security.Subject</code> in the form of an <code>org.apache.shiro.authc.AuthenticationInfo</code>
<code>ddf.security.realm.sts.StsRealm</code>	This realm is an implementation of <code>org.apache.shiro.realm.AuthenticatingRealm</code> and connects to an STS (configurable) to exchange the authentication token for a Subject.
<code>ddf.security.service.AbstractAuthorizingRealm</code>	This is an abstract authorizing realm that takes care of caching and parsing the Subject's <code>AuthorizingInfo</code> and should be extended to allow the implementing realm to focus on making the decision.

Classname	Description
<code>ddf.security.pdp.realm.AuthZRealm</code>	This realm performs the authorization decision and may or may not delegate out to the external XACML processing engine. It uses the incoming permissions to create a decision. However, it is possible to extend this realm using the <code>ddf.security.policy.extension.PolicyExtension</code> interface. This interface allows an integrator to add additional policy information to the PDP that can't be covered via its generic matching policies. This approach is often easier to configure for those that are not familiar with XACML. Note that no <code>PolicyExtension</code> implementations are provided out of the box.
<code>org.codice.ddf.security.validator.*</code>	A number of STS validators are provided for X.509 (BinarySecurityToken), UsernameToken, SAML Assertion, and DDF custom tokens. The DDF custom tokens are all <code>BinarySecurityTokens</code> that may have PKI or username/password information as well as an authentication realm (correlates to JAAS realms installed in the container). The authentication realm allows an administrator to restrict which services they wish to use to authenticate users. For example: installing the <code>security-sts-ldaplogin</code> feature will enable a JAAS realm with the name "ldap". This realm can then be specified on any context using the Web Context Policy Manager. That realm selection is then passed via the token sent to the STS to determine which validator to use.

WARNING

An update was made to the SAML Assertion Handler to pass SAML assertions via headers instead of cookies. Cookies are still accepted and processed to maintain legacy federation compatibility, but only headers are used when federating out. This means that it is still possible to federate and pass a machine's identity, but federation of a user's identity will ONLY work when federating from 2.7.x to 2.8.x+ or between 2.8.x+ and 2.8.x+.

35.2. Security Core

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

Table 145. Security Core Components

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

35.2.1. Security Core API

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

security operations within DDF.

Installing the Security Core API

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.

35.2.2. Configuring the Security Core API

The Security Core API has no configurable properties.

Security Core API Imported Services

None.

Security Core API Exported Services

None.

35.2.3. Security Core Implementation

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

Installing the Security Core Implementation

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 Security Framework.

Configuring the Security Core Implementation

The Security Core Implementation has no configurable properties.

Implementation Details

Table 146. Security Core Imported Services

Registered Interface	Availability	Multiple
<code>org.apache.shiro.realm.Realm</code>	optional	true

Table 147. Security Core Exported Services

Registered Interface	Implementation Class	Properties Set
<code>ddf.security.service.SecurityManager</code>	<code>ddf.security.service.impl.SecurityManagerImpl</code>	None

35.2.4. 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 the Security Core Commons

The Security Core Commons bundle has no configurable properties.

Security Core Commons Imported Services

None.

Security Core Commons Exported Services

None.

35.3. Security Encryption

The 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.

Table 148. Security Encryption 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

35.3.1. 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.

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.

Configuring the Security Encryption API

The Security Encryption API has no configurable properties.

Security Encryption API Imported Services

None.

Security Encryption API Exported Services

None.

35.3.2. 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 Security Encryption Implementation

None.

Security Encryption Implementation Imported Services

None.

Table 149. Security Encryption Implementation Exported Services

Registered Interface	Implementation Class	Properties Set
<code>ddf.security.encryption.EncryptionService</code>	<code>ddf.security.encryption.impl.EncryptionServiceImpl</code>	Key

35.3.3. 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. For more information and sample commands, see [Encryption Service](#).

Installing the 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 the Security Encryption Commands

The Security Encryption Commands have no configurable properties.

Security Encryption Commands Imported Services

None.

Security Encryption Commands Exported Services

None.

35.4. 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.

Table 150. Security LDAP Components

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

35.4.1. Embedded LDAP Server

DDF includes an embedded LDAP server (OpenDJ) for testing and demonstration purposes

Installing the Embedded LDAP Server

The embedded LDAP server is not installed by default with a standard installation.

1. Navigate to the Admin Console.
2. Select **Manage Applications**.
3. Install the **OpenDJ Embedded** application.

Configuring the Embedded LDAP

Configure the Embedded LDAP from the Admin Console:

1. Navigate to the **Admin Console**.
2. Select **Manage Applications**.
3. Start the **OpenDJ Embedded** application.
4. Once started, select the **OpenDJ** application.
5. Select the **Configuration** tab.

Table 151. OpenDJ Embedded Configurable Properties

Configurat ion 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.

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.

WARNING

The embedded LDAP server is intended for testing purposes only and is not recommended for production use.

Connecting 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.

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.

Table 152. Embedded LDAP Default Ports Settings

Protocol	Default Port
LDAP	1389
LDAPS	1636
StartTLS	1389

Table 153. Embedded LDAP Default 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

Username	Password	Groups	Description
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

Table 154. Embedded LDAP Default Admin User Settings

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.

Table 155. Embedded LDAP Default Schemas

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 <code>objectclass</code> 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.

Schema File Name	Schema Description
<code>03-rfc2926.ldif</code>	This file contains schema definitions from RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.
<code>03-rfc3112.ldif</code>	This file contains schema definitions from RFC 3112, which defines the authentication password schema.
<code>03-rfc3712.ldif</code>	This file contains schema definitions from RFC 3712, which defines a mechanism for storing printer information in the directory server.
<code>03-uddiv3.ldif</code>	This file contains schema definitions from RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.
<code>04-rfc2307bis.ldif</code>	This file contains schema definitions from the <code>draft-howard-rfc2307bis</code> specification, used to store naming service information in the directory server.
<code>05-rfc4876.ldif</code>	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.
<code>05-samba.ldif</code>	This file contains schema definitions required when storing Samba user accounts in the directory server.
<code>05-solaris.ldif</code>	This file contains schema definitions required for Solaris and OpenSolaris LDAP naming services.
<code>06-compat.ldif</code>	This file contains the attribute type and <code>objectclass</code> definitions for use with the directory server configuration.

Starting and Stopping the Embedded LDAP

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.

Limitations of the Embedded LDAP

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.

External Links for the Embedded LDAP

Location to the default base LDIF file in the DDF [source code](#).

[OpenDJ documentation](#)

LDAP Administration

OpenDJ provides a number of tools for LDAP administration. Refer to the [OpenDJ Admin Guide](#).

Downloading the Admin Tools

Download [OpenDJ \(Version 2.4.6\)](#) and the included tool suite.

Using 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.

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.

Example Commands for Disabling/Enabling a User's Account

```
D:\OpenDJ-2.4.6\bat>manage-account set-account-is-disabled -h localhost -p 4444 -0
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
```

Notice **Account Is Disabled: true** in the listing:

Verifying an Account is Disabled

```
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:
```

Enabling an 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
```

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

Verifying an Account is Enabled

```
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:
```

35.5. Security PDP

The Security Policy Decision Point (PDP) module contains services that are able to perform authorization decisions based on configurations and policies. In the 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.

Table 156. Security PDP Components

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

35.5.1. Security PDP AuthZ Realm

The Security PDP AuthZ Realm exposes a realm service that makes decisions on authorization requests using the attributes stored within the metocard 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 metocard 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 metocard 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 metocard attribute exists in the corresponding user attribute.

Configuring the Security PDP AuthZ Realm

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

Table 157. Security PDP AuthZ Realm Settings

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

Security PDP AuthZ Realm Imported Services

None.

Table 158. Security PDP AuthZ Realm Exported Services

Registered Interfaces	Implementation Class	Properties Set
<code>org.apache.shiro.realm.Realm</code> <code>org.apache.shiro.authz.Authorizer</code>	<code>ddf.security.pdp.realm.AuthzRealm</code>	None

35.5.2. 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 Security Application.

Configuring Guest Interceptor

Configure the Guest Interceptor from the Admin Console:

1. Navigate to the Admin Console at <https://localhost:8993/admin>
2. Click the 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 additional 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.

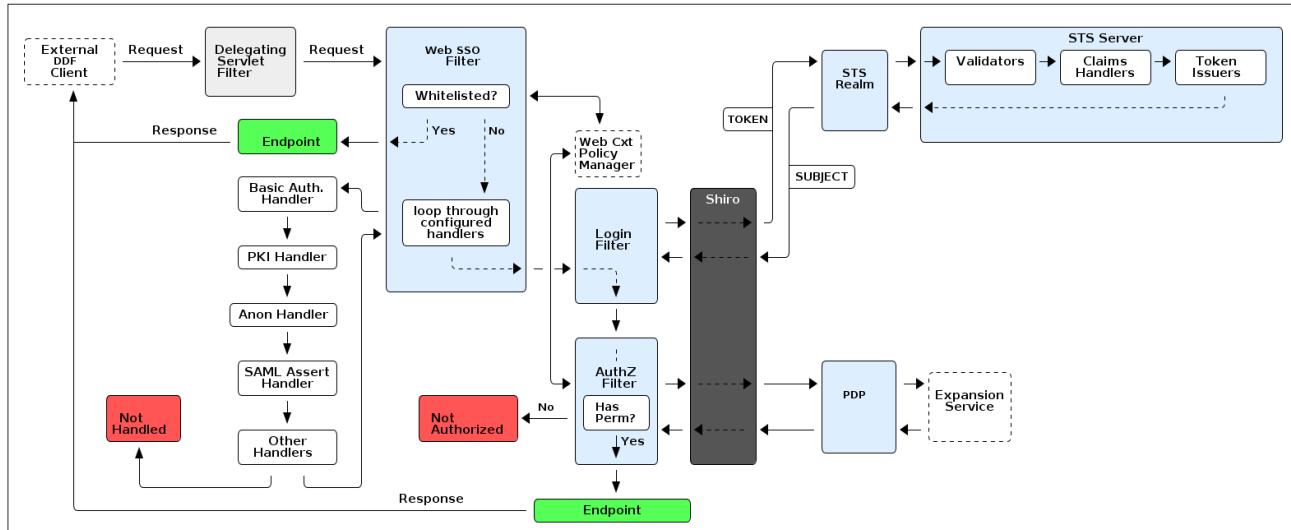
35.6. Web Service Security Architecture

The Web Service Security (WSS) functionality that comes with DDF is integrated throughout the system. This is a central resource describing how all of the pieces work together and where they are

located within the system.

DDF comes with a **Security Framework** and **Security Services**. The Security Framework is the set of APIs that define the integration with the DDF framework and the Security Services are the reference implementations of those APIs built for a realistic end-to-end use case.

35.6.1. Securing REST



The delegating servlet filter is topmost filter for all web contexts. It loads in all security filters. The first filter used is the Web SSO filter. It reads from the web context policy manager and functions as the first decision point. If the request is from a whitelisted context, no further authentication is needed and the request goes directly to the desired endpoint. If the context is not on the whitelist, the filter will attempt to get a handler for the context. The filter loops through all configured context handlers until one signals that it has found authentication information that it can use to build a token. This configuration can be changed by modifying the web context policy manager configuration. If unable to resolve the context, the filter will return an authentication error and the process stops. If a handler is successfully found, an auth token is assigned and the request continues to the login filter. The Login Filter receives a token and return a subject. To retrieve the subject, the token is sent through Shiro to the STS Realm where the token will be exchanged for a SAML assertion through a SOAP call to an STS server. If the Subject is returned, the request moves to the Authorization Filter to check permissions on the user. If the user has the correct permissions to access that web context, the request is allowed to hit the endpoint.

NOTE This diagram does not yet include the SAML 2.0 Web SSO integration.

35.6.2. Encryption Service

The encryption service and encryption command, which are based on [Jasypt](#), provide an easy way for developers to add encryption capabilities to DDF.

Encryption Command

An encrypt security command is provided with DDF that allows plain text to be encrypted. This is

useful when displaying password fields in a GUI.

Below is an example of the security:encrypt command used to encrypt the plain text "myPasswordToEncrypt". The output, `bR9mJpDV08bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=`, is the encrypted value.

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

```
bR9mJpDV08bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

35.6.3. Filtering

Metocard filtering is performed in a Access plugin that occurs after a query has been performed.

How Filtering Works

Each metocard result will contain security attributes that are populated by the CatalogFramework based on the PolicyPlugins (Not provided! You must create your own plugin for your specific metadata!) that populates this attribute. The security attribute is a HashMap containing a set of keys that map to lists of values. The metocard is then processed by a filter plugin that creates a `KeyValueCollectionPermission` from the metocard's security attribute. This permission is then checked against the user subject to determine if the subject has the correct claims to view that metocard. The decision to filter the metocard eventually relies on the PDP (`feature:install security-pdp-authz`). The PDP returns a decision, and the metocard will either be filtered or allowed to pass through.

The security attributes populated on the metocard are completely dependent on the type of the metocard. Each type of metocard must have its own PolicyPlugin that reads the metadata being returned and returns the metocard's security attribute. If the subject permissions are missing during filtering, all resources will be filtered.

Example (represented as simple XML for ease of understanding):

```
<metocard>
  <security>
    <map>
      <entry key="entry1" value="A,B" />
      <entry key="entry2" value="X,Y" />
      <entry key="entry3" value="USA,GBR" />
      <entry key="entry4" value="USA,AUS" />
    </map>
  </security>
</metocard>
```

```

<user>
  <claim name="claim1">
    <value>A</value>
    <value>B</value>
  </claim>
  <claim name="claim2">
    <value>X</value>
    <value>Y</value>
  </claim>
  <claim name="claim3">
    <value>USA</value>
  </claim>
  <claim name="claim4">
    <value>USA</value>
  </claim>
</user>

```

In the above example, the user's claims are represented very simply and are similar to how they would actually appear in a SAML 2 assertion. Each of these user (or subject) claims will be converted to a KeyValuePermission object. These permission objects will be implied against the permission object generated from the metocard record. In this particular case, the metocard might be allowed if the policy is configured appropriately because all of the permissions line up correctly.

35.6.4. Filter a New Type of Metocard

To enable filtering on a new type of record, implement a PolicyPlugin that is able to read the string metadata contained within the metocard record. Note that, in DDF, there is no default plugin that parses a metocard. A plugin must be created to create a policy for the metocard.

35.6.5. Security Token Service

The Security Token Service (STS) is a service running in DDF that generates SAML v2.0 assertions. These assertions are then used to authenticate a client allowing them to issue other requests, such as ingest or queries to DDF services.

The STS is an extension of Apache CXF-STS. It is a SOAP web service that utilizes WS-Trust. The generated SAML assertions contain attributes about a user and is used by the Policy Enforcement Point (PEP) in the secure endpoints. Specific configuration details on the bundles that come with DDF can be found on the Security STS application page. This page details all of the STS components that come out of the box with DDF, along with configuration options, installation help, and which services they import and export.

The STS server contains validators, claim handlers, and token issuers to process incoming requests. When a request is received, the validators first ensure that it is valid. The validators verifies authentication against configured services, such as LDAP, DIAS, PKI. If the request is found to be invalid, the process ends and an error is returned. Next, the claims handlers determine how to handle the request, adding user attributes or properties as configured. The token issuer creates a SAML 2.0 assertion and associates it with the subject. The STS server sends an assertion back to the

requestor, which is used in both SOAP and REST cases.

Using the Security Token Service (STS)

The STS can be used to generate SAML v2.0 assertions via a SOAP web service request. Out of the box, the STS supports authentication from existing SAML tokens, CAS proxy tickets, username/password, and x509 certificates. It also supports retrieving claims using LDAP and properties files.

STS Claims Handlers

Claims handlers are classes that convert the incoming user credentials into a set of attribute claims that will be populated in the SAML assertion. An example in action would be the LDAPClaimsHandler that takes in the user's credentials and retrieves the user's attributes from a backend LDAP server. These attributes are then mapped and added to the SAML assertion being created. Integrators and developers can add more claims handlers that can handle other types of external services that store user attributes.

Add a Custom Claims Handler

Develop a custom claims handler to retrieve attributes from an external attribute store.

A claim is an additional piece of data about a subject that can be included in a token along with basic token data. A claims manager provides hooks for a developer to plug in claims handlers to ensure that the STS includes the specified claims in the issued token.

The following steps define the procedure for adding a custom claims handler to the STS.

1. The new claims handler must implement the `org.apache.cxf.sts.claims.ClaimsHandler` interface.

```

/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership. The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */

package org.apache.cxf.sts.claims;

import java.net.URI;
import java.util.List;

/**
 * This interface provides a pluggable way to handle Claims.
 */
public interface ClaimsHandler {

    List<URI> getSupportedClaimTypes();

    ClaimCollection retrieveClaimValues(RequestClaimCollection claims,
                                        ClaimsParameters parameters);

}

```

2. Expose the new claims handler as an OSGi service under the `org.apache.cxf.sts.claims.ClaimsHandler` interface.

```

<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

    <bean id="CustomClaimsHandler" class=
"security.sts.claimsHandler.CustomClaimsHandler" />

        <service ref="customClaimsHandler" interface=
"org.apache.cxf.sts.claims.ClaimsHandler"/>

</blueprint>

```

3. Deploy the bundle.

If the new claims handler is hitting an external service that is secured with SSL/TLS, a developer may need to add the root CA of the external site to the DDF trustStore and add a valid certificate into the DDF keyStore. For more information on certificates, refer to [\[Configuring a Java Keystore for Secure Communications\]](#).

STS WS-Trust WSDL Document

NOTE

This XML file is found inside of the STS bundle and is named `ws-trust-1.4-service.wsdl`.

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:tns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
xmlns:wstrust="http://docs.oasis-open.org/ws-sx/ws-trust/200512/" xmlns:wsdl=
"http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsap10="http://www.w3.org/2006/05/addressing/wsdl" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp=
"http://www.w3.org/ns/ws-policy" xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsam=
"http://www.w3.org/2007/05/addressing/metadata" targetNamespace="http://docs.oasis-
open.org/ws-sx/ws-trust/200512/">
    <wsdl:types>
        <xsd:schema elementFormDefault="qualified" targetNamespace="http://docs.oasis-
open.org/ws-sx/ws-trust/200512">
            <xsd:element name="RequestSecurityToken" type=
"wst:AbstractRequestSecurityTokenType"/>
            <xsd:element name="RequestSecurityTokenResponse" type=
"wst:AbstractRequestSecurityTokenType"/>
            <xsd:complexType name="AbstractRequestSecurityTokenType">
                <xsd:sequence>
                    <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Context" type="xs:anyURI" use="optional"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>

```

```

<xs:element name="RequestSecurityTokenCollection" type=
"wst:RequestSecurityTokenCollectionType"/>
<xs:complexType name="RequestSecurityTokenCollectionType">
<xs:sequence>
<xs:element name="RequestSecurityToken" type=
"wst:AbstractRequestSecurityTokenType" minOccurs="2" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:element name="RequestSecurityTokenResponseCollection" type=
"wst:RequestSecurityTokenResponseCollectionType"/>
<xs:complexType name="RequestSecurityTokenResponseCollectionType">
<xs:sequence>
<xs:element ref="wst:RequestSecurityTokenResponse" minOccurs="1"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:anyAttribute namespace="#other" processContents="lax"/>
</xs:complexType>
</xs:schema>
</wsdl:types>
<!-- WS-Trust defines the following GEDs -->
<wsdl:message name="RequestSecurityTokenMsg">
<wsdl:part name="request" element="wst:RequestSecurityToken"/>
</wsdl:message>
<wsdl:message name="RequestSecurityTokenResponseMsg">
<wsdl:part name="response" element="wst:RequestSecurityTokenResponse"/>
</wsdl:message>
<wsdl:message name="RequestSecurityTokenCollectionMsg">
<wsdl:part name="requestCollection" element=
"wst:RequestSecurityTokenCollection"/>
</wsdl:message>
<wsdl:message name="RequestSecurityTokenResponseCollectionMsg">
<wsdl:part name="responseCollection" element=
"wst:RequestSecurityTokenResponseCollection"/>
</wsdl:message>
<!-- This portType an example of a Requestor (or other) endpoint that
Accepts SOAP-based challenges from a Security Token Service -->
<wsdl:portType name="WSSecurityRequestor">
<wsdl:operation name="Challenge">
<wsdl:input message="tns:RequestSecurityTokenResponseMsg"/>
<wsdl:output message="tns:RequestSecurityTokenResponseMsg"/>
</wsdl:operation>
</wsdl:portType>
<!-- This portType is an example of an STS supporting full protocol -->
<wsdl:portType name="STS">
<wsdl:operation name="Cancel">
<wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Cancel" message="tns:RequestSecurityTokenMsg"/>
<wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/CancelFinal" message="tns:RequestSecurityTokenResponseMsg"/>
</wsdl:operation>
<wsdl:operation name="Issue">

```

```

<wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue" message="tns:RequestSecurityTokenMsg"/>
    <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTRC/IssueFinal" message="tns:RequestSecurityTokenResponseCollectionMsg
" />
</wsdl:operation>
<wsdl:operation name="Renew">
    <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Renew" message="tns:RequestSecurityTokenMsg"/>
    <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/RenewFinal" message="tns:RequestSecurityTokenResponseMsg"/>
</wsdl:operation>
<wsdl:operation name="Validate">
    <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Validate" message="tns:RequestSecurityTokenMsg"/>
    <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/ValidateFinal" message="tns:RequestSecurityTokenResponseMsg"/>
</wsdl:operation>
<wsdl:operation name="KeyExchangeToken">
    <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/KET" message="tns:RequestSecurityTokenMsg"/>
    <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/KETFinal" message="tns:RequestSecurityTokenResponseMsg"/>
</wsdl:operation>
<wsdl:operation name="RequestCollection">
    <wsdl:input message="tns:RequestSecurityTokenCollectionMsg"/>
    <wsdl:output message="tns:RequestSecurityTokenResponseCollectionMsg"/>
</wsdl:operation>
</wsdl:portType>
<!-- This portType is an example of an endpoint that accepts
    Unsolicited RequestSecurityTokenResponse messages -->
<wsdl:portType name="SecurityTokenResponseService">
    <wsdl:operation name="RequestSecurityTokenResponse">
        <wsdl:input message="tns:RequestSecurityTokenResponseMsg"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="STS_Binding" type="wstrust:STS">
    <wsp:PolicyReference URI="#STS_policy"/>
    <soap:binding style="document" transport="

http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="Issue">
            <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Validate">

```

```

<soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Validate"/>
<wsdl:input>
    <soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
    <soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Cancel">
    <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Cancel"/>
    <wsdl:input>
        <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal"/>
    </wsdl:output>
</wsdl:operation>
<wsdl:operation name="Renew">
    <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Renew"/>
    <wsdl:input>
        <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal"/>
    </wsdl:output>
</wsdl:operation>
<wsdl:operation name="KeyExchangeToken">
    <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/KeyExchangeToken"/>
    <wsdl:input>
        <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal"/>
    </wsdl:output>
</wsdl:operation>
<wsdl:operation name="RequestCollection">
    <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/RequestCollection"/>
    <wsdl:input>
        <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal"/>
    </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsp:Policy wsu:Id="STS_policy">
```

```

<wsp:ExactlyOne>
  <wsp:All>
    <wsap10:UsingAddressing/>
    <wsp:ExactlyOne>
      <sp:TransportBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken>
                <wsp:Policy/>
              </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax/>
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp/>
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:ExactlyOne>
    <sp:Wss11 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
      <wsp:Policy>
        <sp:MustSupportRefKeyIdentifier/>
        <sp:MustSupportRefIssuerSerial/>
        <sp:MustSupportRefThumbprint/>
        <sp:MustSupportRefEncryptedKey/>
      </wsp:Policy>
    </sp:Wss11>
    <sp:Trust13 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
      <wsp:Policy>
        <sp:MustSupportIssuedTokens/>
        <sp:RequireClientEntropy/>
        <sp:RequireServerEntropy/>
      </wsp:Policy>
    </sp:Trust13>
  </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
<wsp:Policy wsu:Id="Input_policy">
  <wsp:ExactlyOne>
    <wsp:All>

```

```

<sp:SignedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
    <sp:Body/>
    <sp:Header Name="To" Namespace=
"http://www.w3.org/2005/08/addressing"/>
    <sp:Header Name="From" Namespace=
"http://www.w3.org/2005/08/addressing"/>
    <sp:Header Name="FaultTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
    <sp:Header Name="ReplyTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
    <sp:Header Name="MessageID" Namespace=
"http://www.w3.org/2005/08/addressing"/>
    <sp:Header Name="RelatesTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
    <sp:Header Name="Action" Namespace=
"http://www.w3.org/2005/08/addressing"/>
</sp:SignedParts>
<sp:EncryptedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
    <sp:Body/>
    </sp:EncryptedParts>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
<wsp:Policy wsu:Id="Output_policy">
    <wsp:ExactlyOne>
        <wsp:All>
            <sp:SignedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                <sp:Body/>
                <sp:Header Name="To" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                <sp:Header Name="From" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                <sp:Header Name="FaultTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                <sp:Header Name="ReplyTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                <sp:Header Name="MessageID" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                <sp:Header Name="RelatesTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                <sp:Header Name="Action" Namespace=
"http://www.w3.org/2005/08/addressing"/>
            </sp:SignedParts>
            <sp:EncryptedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                <sp:Body/>
                </sp:EncryptedParts>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>

```

```

    </wsp:ExactlyOne>
</wsp:Policy>
<wsdl:service name="SecurityTokenService">
    <wsdl:port name="STS_Port" binding="tns:STS_Binding">
        <soap:address location=
"http://localhost:8181/services/SecurityTokenService"/>
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

35.6.6. Example Request and Responses for a SAML Assertion

A client performs a RequestSecurityToken operation against the STS to receive a SAML assertion. The DDF STS offers several different ways to request a SAML assertion. For help in understanding the various request and response formats, samples have been provided. The samples are divided out into different request token types.

Most endpoints that have been used in DDF require the X.509 PublicKey SAML assertion.

35.6.7. BinarySecurityToken (CAS) SAML Security Token Request/Response

BinarySecurityToken (CAS) Sample Request/Response

Request

Sample Request

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RST/Issue</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:60652909-
faca-4e4a-a4a7-8a5ce243a7cb</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>https://server:8993/services/SecurityTokenService</To>
        <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
            <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
        </ReplyTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand
="1">
            <wsu:Timestamp wsu:Id="TS-1">
                <wsu:Created>2013-04-29T18:35:10.688Z</wsu:Created>
                <wsu:Expires>2013-04-29T18:40:10.688Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-

```

```

trust/200512">
    <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
    <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
            <wsa:Address>
https://server:8993/services/SecurityTokenService</wsa:Address>
            </wsa:EndpointReference>
        </wsp:AppliesTo>
        <wst:Claims xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512" Dialect=
"http://schemas.xmlsoap.org/ws/2005/05/identity">
            <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity" Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
            <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity" Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
            <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity" Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
            <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity" Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
            <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity" Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
        </wst:Claims>
        <wst:OnBehalfOf>
            <BinarySecurityToken ValueType="#CAS" EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ns1:Id=
"CAS" xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
U1QtMTQtYUtmcDYxcFRtS0FxZG1pVDMzOWMtY2FzfGh0dHBzOi8vdG9rZW5pc3N1ZXI6Dk5My9zZXJ2aWNlc
y9TZWNIcm10eVRva2VuU2VydmljZQ==</BinarySecurityToken>
            </wst:OnBehalfOf>
            <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0</wst:TokenType>
            <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/PublicKey</wst:KeyType>
            <wst:UseKey>
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data>
                        <ds:X509Certificate>
MIIC5DCCAk2gAwIBAgIJAKj7ROPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUzEQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZHl1YXIxDGAWBgNVBAoMD0xvY2toZWVkJIE1h
cnRpbjENMASGA1UECwwESTDRTEPMA0GA1UEAwwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGNl
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXTDiyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTA1VT
MRAwDgYDVQQIDAdbcm16b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2h1ZWQg

```

```

TWFydgLuMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbGllbnQxHDAaBgkqhkiG9w0BCQEWDWk0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVcnEmHndkf9bWU1AOfNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnh1NW9+lxBDAfBgNVHSMEGDAwgbQx11VLtYXL
vFGpFdHnh1NW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4GBAHYs20I0K6yVXzyS
sKcv2fmfw6XCICGTnyA7B0dAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcDTR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/05tywXRT1+freI3bwAN0
L6tQ
</ds:X509Certificate>
    </ds:X509Data>
        </ds:KeyInfo>
            </wst:UseKey>
            <wst:Renewing/>
        </wst:RequestSecurityToken>
    </soap:Body>
</soap:Envelope>

```

Response

Sample Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:7a6fde04-
9013-41ef-b08b-0689ffa9c93e</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>http://www.w3.org/2005/08/addressing/anonymous</To>
        <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:60652909-
faca-4e4a-a4a7-8a5ce243a7cb</RelatesTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand
="1">
            <wsu:Timestamp wsu:Id="TS-2">
                <wsu:Created>2013-04-29T18:35:11.459Z</wsu:Created>
                <wsu:Expires>2013-04-29T18:40:11.459Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/ws-
sx/ws-trust/200512" xmlns:ns2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" xmlns:ns3="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns4=
"http://www.w3.org/2005/08/addressing" xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-
trust/200802">
            <RequestSecurityTokenResponse>
                <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-

```

```

profile-1.1#SAMLV2.0</TokenType>
    <RequestedSecurityToken>
        <saml2 Assertion xmlns:saml2=
"urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ID=
"_BDC44EB8593F47D1B213672605113671" IssueInstant="2013-04-29T18:35:11.370Z" Version=
"2.0" xsi:type="saml2:AssertionType">
            <saml2:Issuer>tokenissuer</saml2:Issuer>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#" />
                    <ds:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#_BDC44EB8593F47D1B213672605113671
">
                        <ds:Transforms>
                            <ds:Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                            <ds:Transform Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#" />
                            <ec:InclusiveNamespaces xmlns:ec=
"http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs"/>
                            </ds:Transform>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>
                            6wnWbft6Pz5XOF5Q9AG59gcGwLY=</ds:DigestValue>
                            </ds:Reference>
                        </ds:SignedInfo>

<ds:SignatureValue>h+NvkgXGdQtca3/eKebhAKgG38tHp3i2n5uLLy8xXXIg02qyKgEP0FCowp2LiYlsQU9
YjKfSwCUbH3WR6jhAv9zj29CE+ePfEny7MeXvgNl3wId+vcHqtI/DGGhhgtO2MbX/tyX1BhHQUwKR1cHajxHe
ecwmvV7D85NmDv48tI=</ds:SignatureValue>
            <ds:KeyInfo>
                <ds:X509Data>
```

<ds:X509Certificate>MIIDmjCCAoGAwIBAgIBBDANBgkqhkiG9w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA
4GA1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZHl1YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxFVoXDTIz
MDQwNzE4MzcxFMvowgaYxCzAJBgNVBAYTA1VTMRAwDgYDVQQIEwdBcm16b25hMREwDwYDVQQHEwhH
b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgxtN9
uB44diiIqq3J0zDGfDhGLu6mjpuH01hrKITv42hB0hhmH71S9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1LfI7IvsVVvNsrtAEQlj0Mmw9sxS3SUoSQRQX+bD8jq7Uj1hpoF7DdqpV8Kb0COOGwIDAQAB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBhvCAQ0EHxYdT3B1b1NTTCBHZW51cmF0ZWQgQ2Vy
dGlmaWNhdGUwHQYDVR0OBByEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgNVHSMEgZ8wgZyAFBcn
en6/j05DzaVw0RwrtteKc7TZoOxmkdzb1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER

```

MA8GA1UEBxMIR29vZH1YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk70cw07gwwDQYJKoZIhvcNAQEEBQADgYE
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZl06VrAxRQ+eFeN6NyuyhgYy5K6l/sIWdaGou5iJOQx
2pQYWx1v8Klyl0W22IfEAXYv/epi089hpdACryuDJpioXI/X8TAwvRwLKL21Dk3k2b+eyCgA00++
HM0dPfiQLQ99ElWkv/0=</ds:X509Certificate>
    </ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
<saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" NameQualifier="http://cxf.apache.org/sts">srogers</saml2:NameID>
        <saml2:SubjectConfirmation Method=
"urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
            <saml2:SubjectConfirmationData xsi:type=
"saml2:KeyInfoConfirmationDataType">
                <ds:KeyInfo xmlns:ds=
"http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data>

<ds:X509Certificate>MIIC5DCCAk2gAwIBAgIJAKj7ROPHjo1yMA0GCSqGSIB3DQEBCwUAMIGKMQswCQYDVQ
QGEwJVUzEQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZH1YXIxDGAWBgNVBAoMD0xvY2toZWVkIE1h
cnRpbyENMAsGA1UECwwESTRDRTEPMA0GA1UEAwGY2xpZW50MRwwGyJJKoZIhvcNAQkBFg1pNGNl
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXTDiMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTA1VT
MRAwDgYDVQQIDA0Bcm16b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2h1ZWQg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbG1lbnQxHDAaBgvkhkiG9w0BCQEWDWk0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAiPhxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVnEmHndkf9bWU1A0fNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnh1NW9+lxBDAfBgNVHSMEGDAwgbBQx11VLtYXL
vFGpFdHnh1NW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIB3DQEBCwUAA4GBAHYs20I0K6yVXzyS
sKcv2fmfw6XCICGTnyA7B0dAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcDTR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/05tywXRT1+freI3bwAN0
L6tQ</ds:X509Certificate>
    </ds:X509Data>
</ds:KeyInfo>
</saml2:SubjectConfirmationData>
</saml2:SubjectConfirmation>
</saml2:Subject>
<saml2:Conditions NotBefore="2013-04-29T18:35:11.407Z"
NotOnOrAfter="2013-04-29T19:05:11.407Z">
    <saml2:AudienceRestriction>

<saml2:Audience>https://server:8993/services/SecurityTokenService</saml2:Audience>
    </saml2:AudienceRestriction>
</saml2:Conditions>
<saml2:AuthnStatement AuthnInstant="2013-04-29T18:35:11.392Z">
    <saml2:AuthnContext>

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml2:
AuthnContextClassRef>

```

```

        </saml2:AuthnContext>
    </saml2:AuthnStatement>
    <saml2:AttributeStatement>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
>srogers@example.com</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
        </saml2:Attribute>
    </saml2:AttributeStatement>
</saml2:Assertion>
</RequestedSecurityToken>
<RequestedAttachedReference>
    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID"
>_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
        </ns3:SecurityTokenReference>
    </RequestedAttachedReference>

```

```

<RequestedUnattachedReference>
    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID"
>_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
    </ns3:SecurityTokenReference>
</RequestedUnattachedReference>
<wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wst="http://docs.oasis-
open.org/ws-sx/ws-trust/200512">
    <wsa:EndpointReference xmlns:wsa=
"http://www.w3.org/2005/08/addressing">
        <wsa:Address>
https://server:8993/services/SecurityTokenService</wsa:Address>
        </wsa:EndpointReference>
    </wsp:AppliesTo>
    <Lifetime>
        <ns2:Created>2013-04-29T18:35:11.444Z</ns2:Created>
        <ns2:Expires>2013-04-29T19:05:11.444Z</ns2:Expires>
    </Lifetime>
</RequestSecurityTokenResponse>
</RequestSecurityTokenResponseCollection>
</soap:Body>
</soap:Envelope>

```

UsernameToken Bearer SAML Security Token Request/Response

To obtain a SAML assertion to use in secure communication to DDF, a RequestSecurityToken (RST) request has to be made to the STS.

A Bearer SAML assertion is automatically trusted by the endpoint. The client doesn't have to prove it can own that SAML assertion. It is the simplest way to request a SAML assertion, but many endpoints won't accept a KeyType of Bearer.

Request

Explanation

- WS-Addressing header with Action, To, and Message ID
- Valid, non-expired timestamp
- Username Token containing a username and password that the STS will authenticate
- Issued over HTTPS
- KeyType of <http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer>
- Claims (optional): Some endpoints may require that the SAML assertion include attributes of the user, such as an authenticated user's role, name identifier, email address, etc. If the SAML assertion needs those attributes, the **RequestSecurityToken** must specify which ones to include.

Sample Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="TS-1">
        <wsu:Created>2013-04-29T17:47:37.817Z</wsu:Created>
        <wsu:Expires>2013-04-29T17:57:37.817Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="UsernameToken-1">
        <wsse:Username>srogers</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">password1</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID>uuid:a1bba87b-0f00-46cc-975f-001391658cbe</wsa:MessageID>
    <wsa:To>https://server:8993/services/SecurityTokenService</wsa:To>
  </soap:Header>
  <soap:Body>
    <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
      <wst:SecondaryParameters>
        <t:TokenType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</t:TokenType>
        <t:KeyType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer</t:KeyType>
        <t:Claims xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity" xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512" Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
          <!--Add any additional claims you want to grab for the service-->
          <ic:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/uid"/>
          <ic:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
          <ic:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
          <ic:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
          <ic:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
          <ic:ClaimType Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
        </t:Claims>
      </wst:SecondaryParameters>
      <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
```

```

trust/200512/Issue</wst:RequestType>
    <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
            <wsa:Address>
https://server:8993/services/QueryService</wsa:Address>
            </wsa:EndpointReference>
        </wsp:AppliesTo>
        <wst:Renewing/>
    </wst:RequestSecurityToken>
</soap:Body>
</soap:Envelope>

```

Response

This is the response from the STS containing the SAML assertion to be used in subsequent requests to QCRUD endpoints:

The **saml2:Assertion** block contains the entire SAML assertion.

The **Signature** block contains a signature from the STS's private key. The endpoint receiving the SAML assertion will verify that it trusts the signer and ensure that the message wasn't tampered with.

The **AttributeStatement** block contains all the Claims requested.

The **Lifetime** block indicates the valid time interval in which the SAML assertion can be used.

Sample Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RSTR/IssueFinal</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:eee4c6ef-
ac10-4cbc-a53c-13d960e3b6e8</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing">
http://www.w3.org/2005/08/addressing/anonymous</To>
        <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:a1bba87b-0f00-
46cc-975f-001391658cbe</RelatesTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand
="1">
            <wsu:Timestamp wsu:Id="TS-2">
                <wsu:Created>2013-04-29T17:49:12.624Z</wsu:Created>
                <wsu:Expires>2013-04-29T17:54:12.624Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>

```

```

<RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/wss-ws-trust/200512" xmlns:ns2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns4="http://www.w3.org/2005/08/addressing" xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200802">
    <RequestSecurityTokenResponse>
        <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</TokenType>
        <RequestedSecurityToken>
            <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ID="_7437C1A55F19AFF22113672577526132" IssueInstant="2013-04-29T17:49:12.613Z" Version="2.0" xsi:type="saml2:AssertionType">
                <saml2:Issuer>tokenissuer</saml2:Issuer>
                <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:SignedInfo>
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                        <ds:Reference URI="#_7437C1A55F19AFF22113672577526132" />
                    <ds:Transforms>
                        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                    <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs" />
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                <ds:DigestValue>
                    Re0qEbGZlyplW5kqiynX0jPnVEA=</ds:DigestValue>
                </ds:Reference>
            </ds:SignedInfo>

            <ds:SignatureValue>X5Kzd54PrKI1GVV2XxzCmWFRzHRoybF7hU6zxbEhSLMR0AWS9R7Me3epq91Xqe0wvIDDbwmE/oJNC7vI0fIw/rqXkx4aZsY5a5nbAs7f+aXF9T6dk82x2eNhNGYpViq0YZJfsJ5WSyMtG8w5nRekmDMy9oTLsHG+Y/OhJDEwq58=</ds:SignatureValue>
                <ds:KeyInfo>
                    <ds:X509Data>

<ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTERMA8GA1UEBxMIR29vZH1lYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxFVoXTDIzMDQwNzE4MzcxFVoWgaYxCzAJBgNVBAYTA1VTMRAwDgYDVQQIEwdBcm16b25hMREwDwYDVQQHEwhH
```

```

b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtxtN9
uB44diiIqq3J0zDGfDhGLu6mjpuH01hrKITv42hB0hhmH7LS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lf17IvsVVsvNrtAEQ1j0Mmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqpV8Kb0C00GwIDAQAB
o4IBBjCCAQIwCQYDVR0TBAlwADAsBglghkgBvhCAQ0EHxYdT3B1b1NTTCBHZW51cmF0ZWQgQ2Vy
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBqNVHSMEgZ8wgZyAFBcn
en6/j05DzaVw0RwrteKc7TZoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZHl1YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk70cw07gwwDQYJKoZIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZl06VrAxRQ+eFeN6NyuyhgYy5K61/sIWdaGou5iJOQx
2pQYWx1v8Klyl0W22IfEAXYv/epi089hpDAcryuDJpioXI/X8TAwvRwLKL21Dk3k2b+eyCgA00++
HM0dPfiQLQ99ElWkv/0=</ds:X509Certificate>
                                </ds:X509Data>
                                </ds:KeyInfo>
                            </ds:Signature>
                            <saml2:Subject>
                                <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" NameQualifier="http://cxf.apache.org/sts">srogers</saml2:NameID>
                                <saml2:SubjectConfirmation Method=
"urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
                                </saml2:Subject>
                                <saml2:Conditions NotBefore="2013-04-29T17:49:12.614Z"
NotOnOrAfter="2013-04-29T18:19:12.614Z">
                                    <saml2:AudienceRestriction>

<saml2:Audience>https://server:8993/services/QueryService</saml2:Audience>
                                    </saml2:AudienceRestriction>
                                </saml2:Conditions>
                                <saml2:AuthnStatement AuthnInstant="2013-04-29T17:49:12.613Z">
                                    <saml2:AuthnContext>

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml2:
AuthnContextClassRef>
                                    </saml2:AuthnContext>
                                </saml2:AuthnStatement>
                                <saml2:AttributeStatement>
                                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                        <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
                                    </saml2:Attribute>
                                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                        <saml2:AttributeValue xsi:type="xs:string">
>srogers@example.com</saml2:AttributeValue>
                                    </saml2:Attribute>
                                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=

```

```

"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
    </saml2:Attribute>
    </saml2:AttributeStatement>
    </saml2:Assertion>
</RequestedSecurityToken>
<RequestedAttachedReference>
    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID"
>_7437C1A55F19AFF22113672577526132</ns3:KeyIdentifier>
        </ns3:SecurityTokenReference>
    </RequestedAttachedReference>
    <RequestedUnattachedReference>
        <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
            <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID"
>_7437C1A55F19AFF22113672577526132</ns3:KeyIdentifier>
            </ns3:SecurityTokenReference>
        </RequestedUnattachedReference>
        <wsp:AppliesTo xmlns:wsp="

http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wst="http://docs.oasis-
open.org/ws-sx/ws-trust/200512">
            <wsa:EndpointReference xmlns:wsa=
"http://www.w3.org/2005/08/addressing">
                <wsa:Address>
https://server:8993/services/QueryService</wsa:Address>
            </wsa:EndpointReference>

```

```

</wsp:AppliesTo>
<Lifetime>
    <ns2:Created>2013-04-29T17:49:12.620Z</ns2:Created>
    <ns2:Expires>2013-04-29T18:19:12.620Z</ns2:Expires>
</Lifetime>
</RequestSecurityTokenResponse>
</RequestSecurityTokenResponseCollection>
</soap:Body>
</soap:Envelope>

```

X.509 PublicKey SAML Security Token Request/Response

In order to obtain a SAML assertion to use in secure communication to DDF, a **RequestSecurityToken** (RST) request has to be made to the STS.

An endpoint's policy will specify the type of security token needed. Most of the endpoints that have been used with DDF require a SAML v2.0 assertion with a required KeyType of <http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey>. This means that the SAML assertion provided by the client to a DDF endpoint must contain a SubjectConfirmation block with a type of "holder-of-key" containing the client's public key. This is used to prove that the client can possess the SAML assertion returned by the STS.

Request

Explanation

The STS that comes with DDF requires the following to be in the RequestSecurityToken request in order to issue a valid SAML assertion. See the request block below for an example of how these components should be populated.

- WS-Addressing header containing Action, To, and MessageID blocks
- Valid, non-expired timestamp
- Issued over HTTPS
- TokenType of <http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0>
- KeyType of <http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey>
- X509 Certificate as the Proof of Possession or POP. This needs to be the certificate of the client that will be both requesting the SAML assertion and using the SAML assertion to issue a query
- Claims (optional): Some endpoints may require that the SAML assertion include attributes of the user, such as an authenticated user's role, name identifier, email address, etc. If the SAML assertion needs those attributes, the RequestSecurityToken must specify which ones to include.
 - UsernameToken: If Claims are required, the RequestSecurityToken security header must contain a UsernameToken element with a username and password.

Sample Request

```

<soapenv:Envelope xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

```

```

<soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID>uuid:527243af-94bd-4b5c-a1d8-024fd7e694c5</wsa:MessageID>
    <wsa:To>https://server:8993/services/SecurityTokenService</wsa:To>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsu:Timestamp wsu:Id="TS-17">
            <wsu:Created>2014-02-19T17:30:40.771Z</wsu:Created>
            <wsu:Expires>2014-02-19T19:10:40.771Z</wsu:Expires>
        </wsu:Timestamp>

        <!-- OPTIONAL: Only required if the endpoint that the SAML assertion will be
        sent to requires claims. -->
        <wsse:UsernameToken wsu:Id="UsernameToken-16">
            <wsse:Username>pparker</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">password1</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary">
LCTD+5Y7hlWIP6SpsEg9XA==</wsse:Nonce>
            <wsu:Created>2014-02-19T17:30:37.355Z</wsu:Created>
        </wsse:UsernameToken>
        </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
        <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512">
            <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</wst:TokenType>
            <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/PublicKey</wst:KeyType>

            <!-- OPTIONAL: Only required if the endpoint that the SAML assertion will be
            sent to requires claims. -->
            <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity"
            xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity">
                <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
                <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
                <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
                <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
                <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
            </wst:Claims>
            <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>

```

```

<wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Address>https://server:8993/services/QueryService</wsa:Address>
  </wsa:EndpointReference>
</wsp:AppliesTo>
<wst:UseKey>
  <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:X509Data>

<ds:X509Certificate>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGD
AWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjU00VoXDTE2MDUwNzAwMjU00VowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCKNPTlRSQUNUT1IxZxANBgNVBAMTBmNsawWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBA0q6L1/jjZ5cyjhHEbOHr5WQpb0KACYbrsn8lg85LGNoAfcwImr9KBm0xGb
ZCxHYIkW7pJ+kpppH8DbbbDMviIvvdkvrAIU0180BRn2wReCBGQ01Imdc3+WzFF2svW75d6wi2ZVd
eMvU015p/pAD/sdIfXmAfyu8+tqi08KVZGkTnlg3AMzfeSrkcisUHMVWj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpdzN2pC85tj9xIoWGH5dFWG1fPcYRgzGPxsybiG0ylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAhWRGyP1Y08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAAoOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNxy43NLBBlnDjp1NzJoMB0GA1UdDgQWBGRPGiX6zzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2Nybc5nZHMu
bm10LmRp2EubWlsL2Nybc9ET0RKSVDQ0FfMjcuY3JsMIGUoIGRoIGOhGLbGRhcDovL2Nybc5n
ZHMubm10LmRp2EubWlsL2NuJTNkRE9EJTiwSk1UQyUyMENBLTI3JTjb3U1M2RQS0k1MmNvdSUz
ZERvRCUyY281M2RVL1MuJTIwR292ZXJubWVudCUyY2M1M2RVUz9jZXJ0aWZpY2F0ZXJ1dm9jYXRp
b25saXN002JpbmFyeTAjBgNVHSAEHDAaMASGCWCGSAF1AgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzAChjFodHRw0i8vY3Jslmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVDQ0FfMjcuY2Vyc4GCCsGAQUFBzABhiJodHRw0i8vb2NzcC5uc24wLnJjdnuMubm10LmRp
c2EubWlsMA0GCSqGSIB3DQEBCQUAA4IBAQCGUJPGe4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSgxnT4Q9X8WalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIhr
hwtE3xo6qQRRL03/c1C4RnTev1crFVJQVF3yfpRu8udJ2S0GdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCsg1LjC4dmbjKVr9s4QK+/JLNjxD8IkJiZE</ds:X509Certificate>
  </ds:X509Data>
</ds:KeyInfo>
</wst:UseKey>
</wst:RequestSecurityToken>
</soapenv:Body>
</soapenv:Envelope>

```

Response

Explanation

This is the response from the STS containing the SAML assertion to be used in subsequent requests to QCRUD endpoints.

The **saml2:Assertion** block contains the entire SAML assertion.

The **Signature** block contains a signature from the STS's private key. The endpoint receiving the

SAML assertion will verify that it trusts the signer and ensure that the message wasn't tampered with.

The **SubjectConfirmation** block contains the client's public key, so the server can verify that the client has permission to hold this SAML assertion. The **AttributeStatement** block contains all of the claims requested.

Sample Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:b46c35ad-3120-
4233-ae07-b9e10c7911f3</MessageID>
    <To xmlns="http://www.w3.org/2005/08/addressing"
>http://www.w3.org/2005/08/addressing/anonymous</To>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:527243af-94bd-4b5c-
a1d8-024fd7e694c5</RelatesTo>
    <wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <wsu:Timestamp wsu:Id="TS-90DBA0754E55B4FE7013928310431357">
        <wsu:Created>2014-02-19T17:30:43.135Z</wsu:Created>
        <wsu:Expires>2014-02-19T17:35:43.135Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <ns2:RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-
open.org/ws-sx/ws-trust/200802" xmlns:ns2="http://docs.oasis-open.org/ws-sx/ws-
trust/200512" xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://www.w3.org/2005/08/addressing
">
      <ns2:RequestSecurityTokenResponse>
        <ns2:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0</ns2:TokenType>
        <ns2:RequestedSecurityToken>
          <saml2:Assertion ID="_90DBA0754E55B4FE7013928310431176" IssueInstant=
"2014-02-19T17:30:43.117Z" Version="2.0" xsi:type="saml2:AssertionType" xmlns:saml2=
"urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <saml2:Issuer>tokenissuer</saml2:Issuer>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
              <ds:SignedInfo>
                <ds:CanonicalizationMethod Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#" />
                <ds:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                <ds:Reference URI="#_90DBA0754E55B4FE7013928310431176">
```

```

<ds:Transforms>
    <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces PrefixList="xs" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
        </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>
/bEGqsRGHVJbx298WPmGd8I53zs=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
mYR7w1/dnuh8Z7t9xjCb4XkYQLshj+UuYlGOuTwDYsUPcS2qI0nAgMD1VsDP7y1fDJxeqsq7HYhFKsnqRfebMM
4WLH1D/lJ4rD4U0+i9l3tuihm17SN24WM1/b0qfDUCoDqmwG8afUJ3r4vmTNPxwf0ss8BZ/80DgZzm08ndlKx
DfvcN70rExbV/3/45JwF/MMPZoqvi2MJGfX56E9fErJNuzezpWnRqP0lWPxyffKMA1VaB9zF6gvVnUqcW2k/Z8
X91N705jouBI281ZnIfsIPuBJERFtYNVDHsIXM1pJnrY6FlKIaOsisi55LQu3Ruir/n82pU7BT5aWtxwrn7akBg=
=        </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>

<ds:X509Certificate>MIIFHTCCBAWgAwIBAgICJe8wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGD
AWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjYzN1oXDTE2MDUwNzAwMjYzN1owbjELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAstCkNPTlRSQUNUT1IxFDASBgNVBAMTC3Rva2VuaXNzdWVMIIBiIANBqkqhkiG9w0B
AQEFAAACQ8AMIIBCgKCAQEAx01/U4M1wG+wL1JxX2RL1g1j101FkJXMK3Kft3zD//N8x/Dcwwvs
ngCQjXrV6YhbB2V7scHwnThPv3RSwYYi062z+g6ptfbkGGBLSz0zLe3fyJR4RxblFKsELFgPHfx
vgUHS/keG5uSRk9S/0kps/yxKB7+Z1xeFxIsZ5QywXvBpMiXtc2zF+M7BsBStDsx5LcPcDFBwjF
c66rE3/y/25VMht9EZX1QoKr7f8rWD4xgd5J6DYMFWEcniCz4BDJH9sfTw+n1P+CYgrhws1WGqxt
cDME9t6SWR3GLT4Sdtr8ziIM5uUteEhPIV3rVC3/u23JbYEeS8mpnp0bxt5eHQIDAQABo4IB1TCC
AdEwHwYDVR0jBBgwFoAUIxQ0IE1fLjc1ksEGwCOMOmU1kmgwHQYDVR0OBBYEFGBjdkdey+bMHMhC
Z7gwiQ/mJf5VMA4GA1UdDwEB/wQEawIFoDCB2gYDVR0fBIHSMIHPMDagNKAyhjBodHRwOi8vY3Js
Lmdkcy5uaXQuZGlzYS5taWwvY3JsL0RPREpJVENDQV8yNy5jcmwwgZSggZGggY6GgYtsZGFwOi8v
Y3JsLmdkcy5uaXQuZGlzYS5taWwvY241M2RET0Q1MjBKSVDJTlWQ0EtMjclMmNvdSUzZFBLSUy
Y291JTNkRG9EJTJbyUzZFuuUy41MjBHb3Z1cm5tZW50JTjYyUzZFVTP2N1cnRpZmljYXRlcwV2
b2NhG1vbmxpc3Q7YmluYXJ5MCMGA1UdIAQcMBowCwYJYIZIAWUCAQsFMAsgCWCgsAF1AgELEjB9
BggRBgEFBQcBAQRxMG8wPQYIKwYBBQUHMAKGmWh0dHA6Ly9jcmwuZ2RzLm5pdC5kaXNhLm1pbC9z
aWduL0RPREpJVENDQV8yNy5jZXiwlgyIKwYBBQUHMAKGmWh0dHA6Ly9vY3NwLm5zbjAucmN2cy5u
aXQuZGlzYS5taWwvDQYJKoZIhvcNAQEFBQADggEBAlHZQTINU3bMpJ/PkwTYLWPmwCqAYgEUzSYx
bNcVY5MWD8b4XCdw5nM36nFl0qr4IrHeyyOzsEbIebTe3bv011pHx0Uyj059nAhx/AP8DjVtuRU1
/Mp4b6uJ/4yaomjIGceqBzHqhHIJinG0Y2azua7eM9hVbWZsa912ihbiupCq22mYuHFP7NUNzBvV
j03YUcsy/sES5sRx9Rops/CBN+LUUY0dJ0xYWxo8oAbtF8ABE5ATLAwqz4tsToKPUYh1sxdx5Ef
APeZ+wYDmMu40fLckwnCKZgkEtJ0xXpdIJHY+VmyZtQSB0LkR5toeH/ANV4259Ia5ZT8h2/vIJBg
6B4=</ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>

```

```

</ds:Signature>
<saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" NameQualifier="http://cxf.apache.org/sts">pparker</saml2:NameID>
        <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
            <saml2:SubjectConfirmationData xsi:type="saml2:KeyInfoConfirmationDataType">
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data>

<ds:X509Certificate>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGD
AWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIEUBLT13MB4XDTEzMDUwNzAwMjU0VOVoXDTE2MDUwNzAwMjU0VowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPT1RSQUNUT1IxDzANBgNVBAMTBmNsawWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBA0q6L1/jjZ5cyjhjHEbOHr5WQpb0KACYbrsn8lg85LGNoAfcwImr9KBm0xGb
ZCxHYIkW7pJ+kpppH8bbbviIvvdkvrAIU0180BRn2wReCBGQ01Imdc3+WzFF2svW75d6wi2ZVd
eMvU015p/pAD/sdIfXmAfyu8+tqi08KVZGkTnlg3AMzfeSrkci5UHMVwj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpdzN2pC85tj9xIoWgh5dFWG1fPcYRgzGPxsyb1G0ylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAh iWRGyP1Y08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNxY43NZLBBlnDjDp1NZJoMB0GA1UdDgQWBGRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2Nybc5nZHMu
bm10LmRp2EubWlsL2Nybc9ET0RKSVDQ0FfMjcuY3JsMIGUoIGRoIG0hoGLbGRhcDovL2Nybc5n
ZHMubm10LmRp2EubWlsL2NuJTNkRE9EJTIwSk1UQyUyMENBLTI3JTjb3U1M2RQS0k1MmNvdSUz
ZERvRCUyY281M2RVL1MuJTIwR292ZXJubWVudCUyY2M1M2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN0O2JpbmFyeTAjBgnVHSAEHDaaMASGCWCGSAf1AgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzACHjFodHRw0i8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVDQ0FfMjcuY2Vyc4GCCsGAQUFBzABhiJodHRw0i8vb2NzcC5uc24wLnJjdnuMubm10LmRp
c2EubWlsMA0GCSqGSib3DQEBBQUAA4IBAQCGUJPGr4iGCr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4Ypwhuffkx9QqkwSkXXIsim
EhmzSgznT4Q9X8WwalqVYofNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcp0r+iaJBhyDRinIHr
hwtE3xo6qQRRWl03/c1C4RnTev1crFVJQVF3yfpRu8udJ2S0GdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCgx1ljC4dmbjKVR9s4QK+/JLNjxD8IkJiZE</ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
    </saml2:SubjectConfirmationData>
</saml2:SubjectConfirmation>
</saml2:Subject>
<saml2:Conditions NotBefore="2014-02-19T17:30:43.119Z" NotOnOrAfter="2014-02-19T18:00:43.119Z"/>
    <saml2:AuthnStatement AuthnInstant="2014-02-19T17:30:43.117Z">
        <saml2:AuthnContext>

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml2:AuthnContextClassRef>
        </saml2:AuthnContext>
    </saml2:AuthnStatement>

```

```

<!-- This block will only be included if Claims were requested in
the RST. -->
    <saml2:AttributeStatement>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
pparker</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
pparker@example.com</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
pparker</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">Peter
Parker</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
users</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
users</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
        </saml2:Attribute>

```

```

        </saml2:AttributeStatement>
    </saml2:Assertion>
</ns2:RequestedSecurityToken>
<ns2:RequestedAttachedReference>
    <ns4:SecurityTokenReference wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0" xmlns:wsse11=
"http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
        <ns4:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-
wss-saml-token-profile-1.1#SAMLID">
_90DBA0754E55B4FE7013928310431176</ns4:KeyIdentifier>
        </ns4:SecurityTokenReference>
    </ns2:RequestedAttachedReference>
    <ns2:RequestedUnattachedReference>
        <ns4:SecurityTokenReference wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0" xmlns:wsse11=
"http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
        <ns4:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-
wss-saml-token-profile-1.1#SAMLID">
_90DBA0754E55B4FE7013928310431176</ns4:KeyIdentifier>
        </ns4:SecurityTokenReference>
    </ns2:RequestedUnattachedReference>
    <ns2:Lifetime>
        <ns3:Created>2014-02-19T17:30:43.119Z</ns3:Created>
        <ns3:Expires>2014-02-19T18:00:43.119Z</ns3:Expires>
    </ns2:Lifetime>
</ns2:RequestSecurityTokenResponse>
</ns2:RequestSecurityTokenResponseCollection>
</soap:Body>
</soap:Envelope>

```

35.7. Expansion Service

The Expansion Service and its corresponding expansion-related commands provide an easy way for developers to add expansion capabilities to DDF during user attribute and metadata card processing. In addition to these two defined uses of the expansion service, developers are free to utilize the service in their own implementations.

Each instance of the expansion service consists of a collection of rule sets. Each rule set consists of a key value and its associated set of rules. Callers of the expansion service provide a key and an original value to be expanded. The expansion service then looks up the set of rules for the specified key. The expansion service then cumulatively applies each of the rules in the set starting with the original value, with the resulting set of values being returned to the caller.

Key (Attribute)	Rules (original → new)	
key1	value1	replacement1
	value2	replacement2
	value3	replacement3

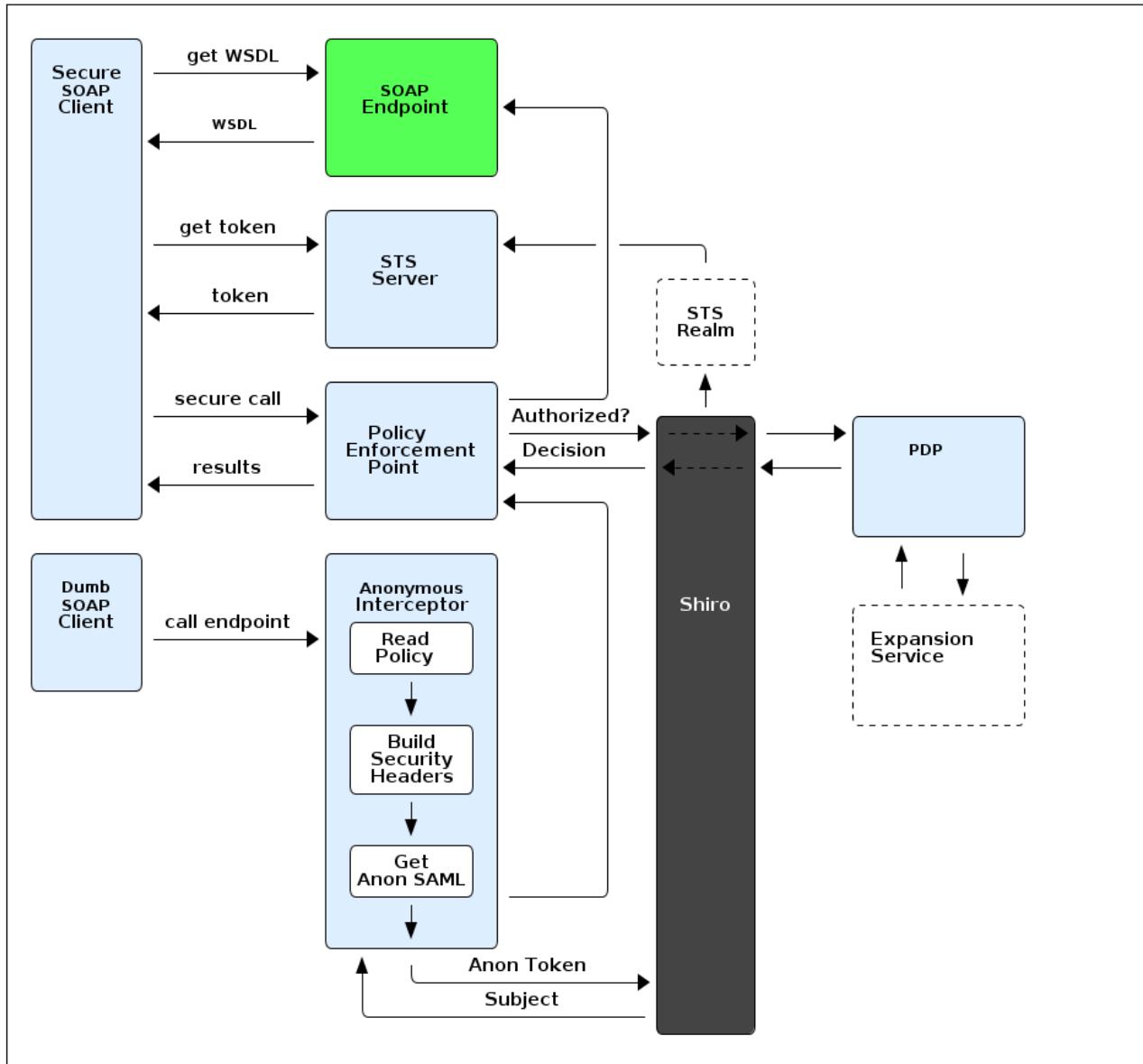
Key (Attribute)	Rules (original → new)	
key2	value1	replacement1
	value2	replacement2

The examples below use the following collection of rule sets:

Key (Attribute)	Rules (original → new)	
Location	Goodyear	Goodyear AZ
	AZ	AZ USA
	CA	CA USA
Title	VP-Sales	VP-Sales VP Sales
	VP-Engineering	VP-Engineering VP Engineering

Note that the rules listed for each key are processed in order, so they may build upon each other, i.e., a new value from the new replacement string may be expanded by a subsequent rule.

35.8. Securing SOAP



35.8.1. SOAP Secure Client

When calling to an endpoint from a SOAP secure client, it first requests the WSDL from the endpoint and the SOAP endpoint returns the WSDL. The client then calls to STS for authentication token to proceed. If the client receives the token, it makes a secure call to the endpoint and receives results.

35.8.2. Dumb SOAP Client

If calling an endpoint from a non-secure client, at the point of the initial call, the Guest Interceptor catches the request and prepares it to be accepted by the endpoint.

First, the interceptor reads the configured policy, builds a security header, and gets an anonymous SAML assertion. Using this, it makes a `getSubject` call which is sent through Shiro to the STS realm. Upon success, the STS realm returns the subject and the call is made to the endpoint.

35.8.3. Developing Token Validators

Token validators are used by the Security Token Service (STS) to validate incoming token requests. The `TokenValidator` CXF interface must be implemented by any custom token validator class. The `canHandleToken` and `validateToken` methods must be overridden. The `canHandleToken` method should return true or false based on the `ValueType` value of the token that the validator is associated with. The validator may be able to handle any number of different tokens that you specify. The `validateToken` method returns a `TokenValidatorResponse` object that contains the `Principal` of the identity being validated and also validates the `ReceivedToken` object that was collected from the RST (`RequestSecurityToken`) message.

35.9. Security PEP

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

Table 159. Security PEP Components

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

35.9.1. 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 the 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 the Security PEP Interceptor

The Security PEP Interceptor has no configurable properties.

Security PEP Interceptor Imported Services

None.

Security PEP Interceptor Exported Services

None.

36. Metrics

DDF includes a system of data-collection to enable monitoring system health, user interactions, and overall system performance: **Metrics Collection**.

36.1. Metrics Services

The Metrics Collection Application 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_HOME>/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.

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

WARNING

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

Types of Metrics Collected

Catalog Metrics

Metrics collected about the catalog status.

Source Metrics

Metrics collected per source.

36.1.1. Metrics Collection Application

The Metrics Collection Application is responsible for collecting both Catalog and Source metrics.

Use Metrics Collection to collect historical metrics data, such as catalog query metrics, message latency, or individual sources' metrics type of data.

Installing Metrics Collection

The Metrics Collection application is installed by default with a standard installation.

The catalog-level metrics are packaged as the `catalog-core-metricsplugin` feature, and the source-level metrics are packaged as the `catalog-core-sourcemetricsplugin` feature.

Configuring Metrics Collection

No configuration is made for the Metrics Collection application. All metrics collected are either pre-configured in DDF or dynamically created as sources are created or deleted.

36.1.2. Metrics Reporting Application

The DDF Metrics Reporting Application provides access to historical data in several formats: 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.

To use the Metrics Reporting Application:

1. Navigate to the **Admin Console**.
2. Select the **Platform** Application.
3. Select the **Metrics** tab.

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_HOME>/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.

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

WARNING

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

Hyperlinks are provided for each metric and each format in which 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 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.

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
```

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.
- 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.

These example reports represent custom aggregate reports. NOTE: all example URLs begin with "https://localhost:8993", which is omitted in the table for brevity.

Table 160. Example Aggregate Reports

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
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

Table 161. Catalog Metrics Collected

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Exceptions	ddf.metrics.catalog:name=Exceptions	Count	The number of exceptions, of all types, thrown across all catalog queries executed.
Catalog Exceptions Federation	ddf.metrics.catalog:name=Exceptions.Federation	Count	The total number of Federation exceptions thrown across all catalog queries executed.
Catalog Exceptions Source Unavailable	ddf.metrics.catalog:name=Exceptions.SourceUnavailable	Count	The total number of <code>SourceUnavailable</code> 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	Total number of <code>UnsupportedQuery</code> 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	The number of catalog entries created in the Metadata Catalog.
Catalog Ingest Deleted	ddf.metrics.catalog:name=Ingest.Deleted	The number of catalog entries deleted from the Metadata Catalog.	Count
Catalog Ingest Updated	ddf.metrics.catalog:name=Ingest.Updated	Count	The number of catalog entries updated in the Metadata Catalog.
Catalog Queries	ddf.metrics.catalog:name=Queries	Count	The number of queries attempted.
Catalog Queries Comparison	ddf.metrics.catalog:name=Queries.Comparison	Count	The number of queries attempted that included a string comparison criteria as part of the search criteria, e.g., <code>PropertyIsLike</code> , <code>PropertyIsEqualTo</code> , etc.
Catalog Queries Federated	ddf.metrics.catalog:name=Queries.Federated	Count	The number of federated queries attempted.
Catalog Queries Fuzzy	ddf.metrics.catalog:name=Queries.Fuzzy	Count	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	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	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	The average of the total number of results returned from executed queries. This total results data is averaged over the metric's sample rate.

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Queries Xpath	ddf.metrics.catalog:name=Queries.X path	Count	The number of queries attempted that included a Xpath criteria as part of the search criteria.
Catalog Resource Retrieval	ddf.metrics.catalog:name=Resource	Count	The number of resources 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.

36.1.3. 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. Metrics are collected for each request (whether enterprise query or a source-specific query). 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>`).

Table 162. Source Metrics Collected

Metric	JMX MBean Name	MBean Attribute Name	Description
Source <code><sourceId></code> Exceptions	<code>ddf.metrics.catalog.source:name=<sourceId>.Exceptions</code>	Count	A count of the total number of exceptions, of all types, thrown from catalog queries executed on this source.
Source <code><sourceId></code> Queries	<code>ddf.metrics.catalog.source:name=<sourceId>.Queries</code>	Count	A count of the number of queries attempted on this source.
Source <code><sourceId></code> Queries Total Results	<code>ddf.metrics.catalog.source:name=<sourceId>.Queries.TotalResults</code>	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_HOME>/data/metrics` directory. Also note that the source metrics' names are modified to remove all non-alphanumeric characters and renamed in camelCase.

36.1.4. Viewing Metrics in the Admin Console

The Metrics viewer has reports in various formats.

1. Navigate to the **Admin Console**.
2. Select the **Platform** application.
3. Select the **Metrics** tab.

Reports are organized by timeframe and output format.

Standard time increments: * `15m`: 15 Minutes * `1h`: 1 Hour * `1d`: 1 Day * `1w`: 1 Week * `1M`: 1 Month * `3M`: 3 Month * `6M`: 6 Month * `1y`: 1 Year

Custom timeframes are also available via the selectors at the bottom of the page.

Output formats: * PNG * CSV (Comma-separated values) * XLS

NOTE

Based on the browser's configuration, either the `.xls` file will be downloaded or automatically displayed in Excel.

37. Action Framework

The Action Framework was designed as a way to limit dependencies between applications (apps) in a system. For instance, a feature in an app, such as an Atom feed generator, might want to include an external link as part of its feed's entries. That feature does not have to be coupled to a REST endpoint to work, nor does it have to depend on a specific implementation to get a link. In reality, the feature does not identify how the link is generated, but it does identify whether the link works or does not work when retrieving the intended entry's metadata. Instead of creating its own mechanism or adding an unrelated feature, it could use the Action Framework to query the OSGi container for any service that can provide a link. This does two things: it allows the feature to be

independent of implementations, and it encourages reuse of common services.

The Action Framework consists of two major Java interfaces in its API:

1. `ddf.action.Action`
2. `ddf.action.ActionProvider`

37.1. Actions

Actions are specific tasks that can be performed as services.

37.2. Action Providers

Action Providers are lists of related actions that a service is capable of performing.

Included Action Providers

Download Resource ActionProvider

Downloads a resource to the local product cache.

IdP Logout Action Provider

Identity Provider Logout.

Karaf Logout Action

Local Logout.

LDAP Logout Action

Ldap Logout.

Overlay ActionProvider

Provides a metocard URL that transforms the metocard into a geographically aligned image (suitable for overlaying on a map).

View Metocard ActionProvider

Provides a URL to a metocard.

Metocard Transformer ActionProvider

Provides a URL to a metocard that has been transformed into a specified format.

37.2.1. Developing Action Components

To provide a service, such as a link to a metocard, the `ActionProvider` interface should be implemented. An `ActionProvider` essentially provides a List of `Actions` when given input that it can recognize and handle. For instance, if a REST endpoint ActionProvider was given a metocard, it could provide a link based on the metocard's ID. An Action Provider performs an action when given a subject that it understands. If it does not understand the subject or does not know how to handle the given input, it will return `Collections.emptyList()`. An Action Provider is required to have an `ActionProvider` id. The Action Provider must register itself in the OSGi Service Registry with the `ddf.action.ActionProvider` interface and must also have a service property value for `id`.

An action is a URL that, when invoked, provides a resource or executes intended business logic.

Action Component Naming Convention

For each Action, a title and description should be provided to describe what the action does. The recommended naming convention is to use the verb 'Get' when retrieving a portion of a metocard, such as the metadata or thumbnail, or when downloading a product. The verb 'Export' or the expression 'Export as' is recommended when the metocard is being exported in a different format or presented after going some transformation.

Action Component Taxonomy

An Action Provider registers an **id** as a service property in the OGSi Service Registry based on the type of service or action that is provided. Regardless of implementation, if more than one Action Provider provides the same service, such as providing a URL to a thumbnail for a given metocard, they must both register under the same **id**. Therefore, Action Provider implementers must follow an Action Taxonomy.

The following is a sample taxonomy:

1. **catalog.data.metocard** shall be the grouping that represents Actions on a Catalog metocard.
 - a. **catalog.data.metocard.view**
 - b. **catalog.data.metocard.thumbnail**
 - c. **catalog.data.metocard.html**
 - d. **catalog.data.metocard.resource**
 - e. **catalog.data.metocard.metadata**

Table 163. Action ID Service Descriptions

ID	Required Action	Naming Convention
catalog.data.metocard.view	Provides a valid URL to view a metocard. Format of data is not specified; i.e. the representation can be in XML, JSON, or other.	Export as ...
catalog.data.metocard.thumbnail	Provides a valid URL to the bytes of a thumbnail (Metocard.THUMBNAIL) with MIME type image/jpeg.	Export as Thumbnail
catalog.data.metocard.map.overlay.thumbnail	Provides a metocard URL that translates the metocard into a geographically aligned image (suitable for overlaying on a map).	Export as Thumbnail Overlay
catalog.data.metocard.html	Provides a valid URL that, when invoked, provides an HTML representation of the metocard.	Export as HTML
catalog.data.metocard.xml	Provides a valid URL that, when invoked, provides an XML representation of the metocard.	Export as XML
catalog.data.metocard.geojson	Provides a valid URL that, when invoked, provides an XML representation of the metocard.	Export as GeoJSON

ID	Required Action	Naming Convention
<code>catalog.data.metocard.resource</code>	Provides a valid URL that, when invoked, provides the underlying resource of the metocard.	Export as Resource
<code>catalog.data.metocard.metadata</code>	Provides a valid URL to the XML metadata in the metocard (Metocard.METADATA).	Export as Metadata

38. Migration API

38.1. Migration API

NOTE

This code is experimental. While this interface is functional and tested, it may change or be removed in a future version of the library.

DDF currently has an experimental API for making bundles migratable. Interfaces in [platform/migration/platform-migratable-api](#) are used by the system to identify bundles that provide implementations for export and import(*coming soon*) operations.

The [Migratable](#) API provides a mechanism for bundles to handle exporting data required to clone a DDF system. The migration process is meant to be flexible, so an implementation of `org.codice.ddf.migration.Migratable` can handle exporting data for a single bundle or groups of bundles such as applications. For example, the `org.codice.ddf.platform.migratable.impl.PlatformMigratable` handles exporting core system files for the Platform Application. Exporting configurations stored in `org.osgi.service.cm.ConfigurationAdmin` does not need to be handled by implementations of `org.codice.ddf.migration.Migratable` as all `ConfigurationAdmin` configurations are exported by `org.codice.ddf.configuration.admin.ConfigurationAdminMigration`. An export operation can be performed through the Command Console or through the Admin Console. When an export operation is processed, the migration API will do a look-up for all registered OSGi services that are implementing `Migratable` and call their `export()` method.

The services that implement one of the migratable interfaces will be called one at a time, in any order, and do not need to be thread safe. A bundle or a feature can have as many services implementing the interfaces as needed.

38.1.1. The Migratable API Interfaces

1. `org.codice.ddf.migration.Migratable`
2. `org.codice.ddf.migration.AbstractMigratable`
3. `org.codice.ddf.migration.MigrationException`
4. `org.codice.ddf.migration.MigrationMetadata`
5. `org.codice.ddf.migration.MigrationWarning`

Migratable

The contract for a `migratable` is stored here. It is used by both sub-interfaces `ConfigurationMigratable` and `DataMigratable`.

WARNING

Do not implement `Migratable` directly; it is intended for use only as a common base interface. Instead, the appropriate sub-interface should be used.

The `org.codice.ddf.migration.Migratable` interface defines these methods:

```
boolean isOptional()
```

The `exportPath` in `export(Path exportPath)` is the path where all of the exportable data is copied. It is provided via an argument to the `migration:export` console command or via the Export Dialog in the Admin Console. The default value is `<DDF_HOME>/etc/exported`. It is the responsibility of a `Migratable` to prevent naming collisions upon export. For example, if a `Migratable` writes files for its export, it must namespace the files. The `getDescription()` operation returns a short description of the type of data exported by the `Migratable`. The `isOptional()` operation returns whether the exported data for the `Migratable` is optional or required. The description and optional flag are for display purposes in the Admin Console.

In order to create a `Migratable` for a module of the system, the `org.codice.ddf.migration.Migratable` interface must be implemented and the implementation must be registered under the `org.codice.ddf.migration.Migratable` interface as an OSGI service in the OSGI service registry.

Creating an OSGI service allows for the `org.codice.ddf.configuration.migration.ConfigurationMigrationManager` to lookup all implementations of `org.codice.ddf.migration.Migratable` and command them to export.

ConfigurationMigratable

This is the base interface that must be implemented by all bundles or features that need to export and/or import system related settings (e.g., bundle specific Java properties, XML or JSON configuration files) during system migration. The information exported must allow the new system to have the same configuration as the original system. Only bundle- or feature-specific settings need be handled. All configurations stored in OSGi's `ConfigurationAdmin` will automatically be migrated and do not need to be managed by implementors of this class.

Also, any other data not related to the system's configuration and settings (for example, data stored in Solr) should be handled by a different service that implements the `DataMigratable` interface, not by implementors of this class.

DataMigratable

This is the interface that must be implemented by all bundles or features that need to export and/or import data during system migration. The data is not mandatory for the system to come up (e.g., data stored in Solr) and doesn't affect the system's configuration, i.e., without this data, the new system will still have the same configuration as the original one.

Any system related configuration and settings should be handled by a different service that implements the `ConfigurationMigratable` interface, not by implementors of this class.

`AbstractMigratable`

The abstract base class `org.codice.ddf.migration.AbstractMigratable` in the `platform-migratable-api` implements common boilerplate code required when implementing `org.codice.ddf.migration.Migratable` and should be extended when creating an `org.codice.ddf.migration.Migratable`.

`MigrationException`

An `org.codice.ddf.migration.MigrationException` should be thrown when an unrecoverable exception occurs that prevents required data from exporting. The exception message is displayed to the administrator.

`MigrationMetadata`

The `org.codice.ddf.migration.MigrationMetadata` provides metadata with details of the exported `org.codice.ddf.migration.Migratable`.

`MigrationWarning`

An `org.codice.ddf.migration.MigrationWarning` should be used when a `Migratable` wants to warn an admin that certain aspects of the export may cause problems upon import. For example, if an absolute path is encountered, that path may not exist on the target system and cause the installation to fail. All migration warnings are displayed to the administrator.

Do Not Use FileBackedOutputStream

It is recommended to avoid using `com.google.common.io.FileBackedOutputStream` (FBOS). FBOS can create temporary files that are not automatically removed when FBOS is closed. Use `org.codice.ddf.platform.util.TemporaryFileBackedOutputStream` (TFBOS). TFBOS provides the same method calls as FBOS, but will remove temporary files when it is closed.

WARNING

39. Application Service API

39.1. Application Service API

The Application service has multiple interfaces which are exposed on top of the OSGi runtime for other applications to use. For more information on these interfaces, see [Application Service Interfaces](#).

39.1.1. 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 applications that are specified by their URL.

39.1.2. Initial Application Installation

In DDF, an **application** is a collection of one or more **features**.

WARNING

This application list configuration file is only read the first time the server is started after being unzipped.

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 Admin 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>=<location>` 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.

39.1.3. Application Service Interfaces

The interfaces used by that application service subsystem are **ApplicationService**, **Application**, **ApplicationStatus**, and **ApplicationNode**. The purpose of these interfaces and their public methods

are explained below.

ApplicationService Interface

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

`getApplications()`

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.

`getApplication(String applicationName)`

Returns the application that has the given name.

`startApplication(Application application)`

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

`stopApplication(Application application)`

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

`addApplication(URI applicationURL)`

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

`removeApplication(URI applicationURL)`

Removes an application that has the given URI.

`isApplicationStarted(Application application)`

Returns a boolean value relating whether a given application is started or not. This method is generally called after retrieving a list of applications in the first method.

`getApplicationStatus(Application application)`

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(Feature feature)`

Determine which application contains a certain feature.

Application Interface

`getName()`

Returns the name of the application, which should be unique among applications.

`getFeatures()`

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

`getBundles()`

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

ApplicationStatus Interface

getApplication

Returns 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 for a node reference.

getStatus()

Returns the status for the referenced application.

getParent()

Returns the parent of the application.

getChildren()

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

Table 164. Application Service Imported Services

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

Table 165. Application Service Exported Services

Registered Interface	Implementation Class	Notes
<code>org.codice.ddf.admin.application.service.ApplicationService</code>	<code>org.codice.ddf.admin.application.service.impl.ApplicationServiceImpl</code>	

40. General Development Guidelines

The Distributed Data Framework is free and open-source software offered under the GNU Lesser General Public License. The DDF is managed under the guidance of the [Codice Foundation](#).

Contributions are welcomed and encouraged. Please visit the [Codice DDF Contributor Guidelines](#) and the [DDF source code repository](#) for more information.

40.1. Developing for DDF

DDF includes several extension points where external developers can add functionality to support individual use cases.

DDF is written in Java and uses many open source libraries. DDF uses OSGi to provide modularity, lifecycle management, and dynamic services. OSGi services can be installed and uninstalled while DDF is running. DDF development typically means developing new OSGi bundles and deploying them to the running DDF. A complete description of OSGi is outside the scope of this documentation. For more information about OSGi, see the [OSGi Alliance website](#).

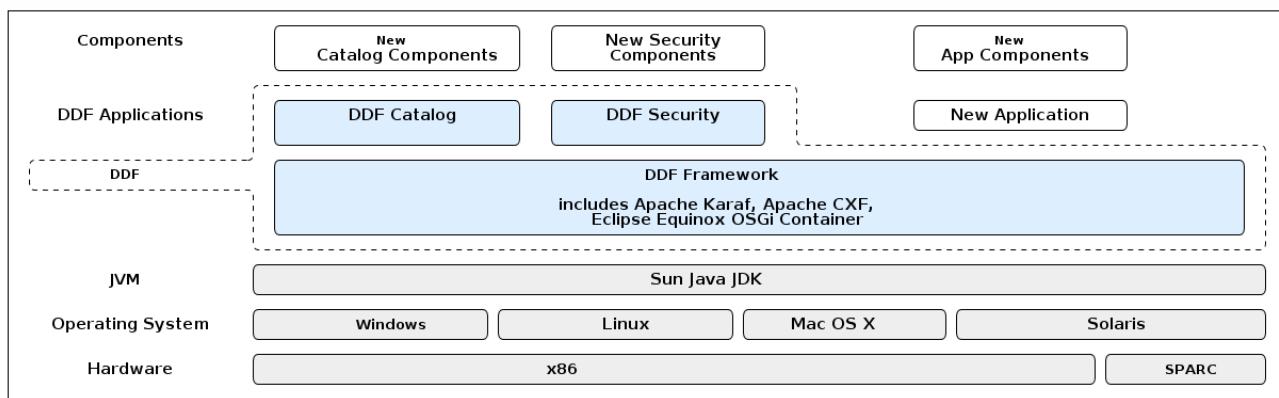


Figure 45. Architecture Diagram

40.2. OSGi Basics

DDF runs on top of an OSGi framework, a Java virtual machine (JVM), several choices of operating systems, and the physical hardware infrastructure. The items within the dotted line represent the standard DDF components.

DDF is a customized and branded distribution of [Apache Karaf](#). DDF could also be considered to be a more lightweight OSGi distribution, as compared to Apache ServiceMix, FUSE ESB, or Talend ESB, all of which are also built upon Apache Karaf. Similar to its peers, DDF incorporates ([additional upstream dependencies](#)).

The DDF framework hosts DDF applications, which are extensible by adding components via OSGi. The best example of this is the DDF Catalog (API), which offers extensibility via several types of Catalog Components. The DDF Catalog API serves as the foundation for several applications and resides in the applications tier.

The Catalog Components consist of [Endpoints](#), [Plugins](#), [Catalog Frameworks](#), [Sources](#), and [Catalog Providers](#). Customized components can be added to DDF.

Capability

A general term used to refer to an ability of the system.

Component

Represents a portion of an Application that can be extended.

Bundle

Java Archives (JARs) with special OSGi manifest entries.

Feature

One or more bundles that form an installable unit; defined by Apache Karaf but portable to other OSGi containers.

Application

One or more features that together form a cohesive collection of capabilities.

40.2.1. Packaging Capabilities as Bundles

Services and code are physically deployed to DDF using bundles. The bundles within DDF are created using the maven bundle plug-in. Bundles are Java JAR files that have additional metadata in the [MANIFEST.MF](#) that is relevant to an OSGi container.

The best resource for learning about the structure and headers in the manifest definition is in section 3.6 of the [OSGi Core Specification](#). The bundles within DDF are created using the [maven bundle plug-in](#), which uses the [BND tool](#).

Alternative Bundle Creation Methods

TIP

Using Maven is not necessary to create bundles. Many alternative tools exist, and OSGi manifest files can also be created by hand, although hand-editing should be avoided by most developers.

Creating a Bundle

Bundle Development Recommendations

Avoid creating bundles by hand or editing a manifest file

Many tools exist for creating bundles, notably the Maven Bundle plugin, which handle the details of OSGi configuration and automate the bundling process including generation of the manifest file.

Always make a distinction on which imported packages are optional or required

Requiring every package when not necessary can cause an unnecessary dependency ripple effect among bundles.

Embedding is an implementation detail

Using the [Embed-Dependency](#) instruction provided by the [maven-bundle-plugin](#) will insert the specified jar(s) into the target archive and add them to the [Bundle-ClassPath](#). These jars and their contained packages/classes are not for public consumption; they are for the internal implementation of this service implementation only.

Bundles should never be embedded

Bundles expose service implementations; they do not provide arbitrary classes to be used by other bundles.

Bundles should expose service implementations

This is the corollary to the previous rule. Bundles should not be created when arbitrary concrete classes are being extracted to a library. In that case, a library/jar is the appropriate module packaging type.

Bundles should generally only export service packages

If there are packages internal to a bundle that comprise its implementation but not its public manifestation of the API, they should be excluded from export and kept as private packages.

Concrete objects that are not loaded by the root classloader should not be passed in or out of a bundle

This is a general rule with some exceptions (JAXB generated classes being the most prominent example). Where complex objects need to be passed in or out of a service method, an interface should be defined in the API bundle.

Bundles separate contract from implementation and allow for modularized development and deployment of functionality. For that to be effective, they must be defined and used correctly so inadvertent coupling does not occur. Good bundle definition and usage leads to a more flexible environment.

Maven Bundle Plugin

Below is a code snippet from a Maven `pom.xml` for creating an OSGi Bundle using the Maven Bundle plugin.

Maven pom.xml

```
...
<packaging>bundle</packaging>
...
<build>
...
<plugin>
    <groupId>org.apache.felix</groupId>
    <artifactId>maven-bundle-plugin</artifactId>
    <configuration>
        <instructions>
            <Bundle-Name>${project.name}</Bundle-Name>
            <Export-Package />
            <Bundle-SymbolicName>${project.groupId}.${project.artifactId}</Bundle-
SymbolicName>
            <Import-Package>
                ddf.catalog,
                ddf.catalog.*
            </Import-Package>
        </instructions>
    </configuration>
</plugin>
...
</build>
...
```

Third Party and Utility Bundles

It is recommended to avoid building directly on included third party and utility bundles. These components do provide utility and reuse potential; however, they may be upgraded or even replaced at anytime as bug fixes and new capabilities dictate. For example, web services may be built using CXF. However, the distributions frequently upgrade CXF between releases to take advantage of new features. If building on these components, be aware of the version upgrades with each distribution release.

Instead, component developers should package and deliver their own dependencies to ensure future compatibility. For example, if re-using a bundle, the specific bundle version that you are depending on should be included in your packaged release, and the proper versions should be referenced in your bundle(s).

Deploying a Bundle

A bundle is typically installed in one of two ways:

1. Installed as a feature
2. Hot deployed in the `/deploy` directory

The fastest way to deploy a created bundle during development is to copy it to the `/deploy` directory of a running DDF. This directory checks for new bundles and deploys them immediately. According

to Karaf documentation, "Karaf supports hot deployment of OSGi bundles by monitoring JAR files inside the [home]/deploy directory. Each time a JAR is copied in this folder, it will be installed inside the runtime. It can be updated or deleted and changes will be handled automatically. In addition, Karaf also supports exploded bundles and custom deployers (Blueprint and Spring DM are included by default)." Once deployed, the bundle should come up in the Active state, if all of the dependencies were properly met. When this occurs, the service is available to be used.

Verifying Bundle State

To verify if a bundle is deployed and running, go to the running command console and view the status.

- Execute the `list` command.
- If the name of the bundle is known, the `list` command can be piped to the `grep` command to quickly find the bundle.

The example below shows how to verify if a Client is deployed and running.

Verifying with grep

```
ddf@local>list | grep -i example
[ 162] [Active      ] [          ] [  ] [ 80] DDF :: Registry :: example Client (2.0.0)
```

The state is `Active`, indicating that the bundle is ready for program execution.

40.2.2. Working with Features Files

Features XML files group other features and/or bundle(s) for ease of installation/uninstallation.

In order to ensure that the installer can correctly interpret and display application details, there are several guidelines that should be followed when creating the features file for the application.

- Ensure only one feature in the `features.xml` has the same name as the project itself. (e.g. If the project name is `spatial-app`, there should be one feature named `spatial-app`) This is the feature that the installer displays to the user (name, description, version, etc.) and provides the description of what the application contains. This feature should only declare the `prerequisite` features of the app and any bundles or features that are required by all of the features within the app.
- Be sure all features that should be installed by default include the `auto-install` flag.

```
install='auto'
```

- Each feature should declare its `prerequisite` features. These are the features that are required to be installed prior to this feature starting up. At a minimum this should include the application feature, so that the application prerequisites and required dependencies are satisfied before this feature starts.

```
<feature prerequisite="true">feature1</feature>
```

Auto-starting an Application Feature

Within the `features.xml` file for an application, the feature that is named the same as the project will have the `install` attribute set to `auto`. Within this feature, refer to any prerequisites of the application as well as any features or bundles within the application that are required by all features. Features that should start automatically should have `install` set to `auto`. Other features should have `install` set to `manual`.

The following example demonstrates configuring features to be auto-started. The naming convention for this feature is typically “application name” + “`-app`,” as shown.

Auto-start Features

```
<feature name="example-app" install="auto">
    <feature prerequisite="true">prereq-feature</feature>
    <bundle>mvn:com.example/example.dependency/1.3</bundle>
</feature>
<feature name="auto-start-feature" install="auto">
    <feature prerequisite="true">example-app</feature>
    <bundle>mvn:com.example/example.auto/1.3</bundle>
</feature>
<feature name="manual-start-feature" install="manual">
    <feature prerequisite="true">example-app</feature>
    <bundle>mvn:com.example/example.manual/1.3</bundle>
</feature>
```

40.3. Developing DDF Applications

The DDF applications are comprised of components, packaged as Karaf features, which are collections of OSGi bundles. These features can be installed/uninstalled using the Admin Console or Command Console. DDF applications also consist of one or more OSGi bundles and, possibly, supplemental external files. These applications are packaged as Karaf KAR files for easy download and installation. These applications can be stored on a file system or a Maven repository.

A KAR file is a Karaf-specific archive format (*K*araf *AR*chive). It is a jar file that contains a feature descriptor file and one or more OSGi bundle jar files. The feature descriptor file identifies the application’s name, the set of bundles that need to be installed, and any dependencies on other features that may need to be installed.

40.3.1. Describing Application Services

Given the modular nature of OSGi, some applications perform operations on the services themselves. In order to present, identify, and manipulate the services, they need descriptive identifying information. Any service that implements the `Describable` interface in `org.codice.ddf.platform.services.common` will have an obligation to provide this information. The

relevant fields are as follows:

ID

a unique identifier for the service

Title

the informal name for the service

Description

a short, human-consumable description of the service

Organization

the name of the organization that wrote the service

Version

the current version of the service (example: 1.0)

The only field with stringent requirements is the ID field. The format is **[product].[component]** such as **ddf.metacards** or **ddf.platform**; while the **[component]** within a **[product]** may simply be a module or bundle name, the **[product]** itself should be the unique name of the plug-in or integration that belongs to the organization provided. Note that **ddf** as a **[product]** is reserved for core features only and is not meant to be used during extension or integration.

40.3.2. Creating a KAR File

The recommended method for creating a KAR file is to use the **features-maven-plugin**, which has a **create-kar** goal. This goal reads all of the features specified in the feature's descriptor file. For each feature in this file, it resolves the bundles defined in the feature. All bundles are then packaged into the KAR archive.

create-kar Goal Example

```
<plugin>
<groupId>org.apache.karaf.tooling</groupId>
<artifactId>features-maven-plugin</artifactId>
<version>2.2.5</version>
<executions>
    <execution>
        <id>create-kar</id>
        <goals>
            <goal>create-kar</goal>
        </goals>
        <configuration>
            <descriptors>
                <!-- Add any other <descriptor> that the features file may
reference here -->
                </descriptors>
                <!--
                    Workaround to prevent the target/classes/features.xml file from being
included in the
                    kar file since features.xml already included in kar's repository
directory tree.
                    Otherwise, features.xml would appear twice in the kar file, hence
installing the
                    same feature twice.
                    Refer to Karaf forum posting at
http://karaf.922171.n3.nabble.com/Duplicate-feature-repository-entry-using-archive-kar-to-build-deployable-applications-td3650850.html
-->
                <resourcesDir>${project.build.directory}/doesNotExist</resourcesDir>

                <!--
                    Location of the features.xml file. If it references properties that
need to be filtered, e.g., ${project.version}, it will need to be
                    filtered by the maven-resources-plugin.
-->
                <featuresFile>${basedir}/target/classes/features.xml</featuresFile>

                <!-- Name of the kar file (.kar extension added by default). If not
specified, defaults to ${project.build.finalName} -->
                <finalName>ddf-ifis-${project.version}</finalName>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Examples of how KAR files are created for DDF components can be found in the DDF source code under the [ddf/distribution/ddf-kars directory](#).

The `.kar` file generated should be deployed to the application author's maven repository. The URL

to the application's KAR file in this Maven repository should be the installation URL that is used.

40.3.3. Including Data Files in a KAR File

The developer may need to include data or configuration file(s) in a KAR file. An example of this is a properties file for the JDBC connection properties of a catalog provider.

It is recommended that:

- Any `data/configuration` files be placed under the `src/main/resources` directory of the maven project. Sub-directories under `src/main/resources` can be used, e.g., `etc/security`.
- The Maven project's pom file should be updated to attach each `data/configuration` file as an artifact (using the `build-helper-maven-plugin`).
- Add each `data/configuration` file to the KAR file using the `<configfile>` tag in the KAR's `features.xml` file.

40.3.4. Installing a KAR File

When the user downloads an application by clicking on the **Installation** link, the application's KAR file is downloaded. To install manually, the KAR file can be placed in the `<DDF_HOME>/deploy` directory of the running DDF instance. DDF then detects that a file with a `.kar` file extension has been placed in this monitored directory, unzips the KAR file into the `<DDF_HOME>/system` directory, and installs the bundle(s) listed in the KAR file's feature descriptor file. To install via the Admin Console: . Navigate to <https://localhost:8993/admin> . Click the **Manage** button in the upper right . Click the **Add an Application** tile . Upload the KAR file via the popup window . Click **Save Changes** to activate The new application can be viewed via the Admin Console's Active Applications list.

Developing Application Configuration Modules

An application within DDF is a collection of bundles contained in a KAR file that may or may not have configurations associated with it. Plugins are used to advertise applications. These configuration module plugins are often used to add user interface elements to make the use of the DDF simpler and/or more intuitive.

Creating an Application Configuration Module

This example demonstrates a plugin that allows the DDF to use the Admin UI.

1. Create an application plugin to advertise your configuration by extending `AbstractApplicationPlugin`.

```

import org.codice.ddf.admin.application.plugin.AbstractApplicationPlugin;

public class SourcesPlugin extends AbstractApplicationPlugin {
    /**
     * Constructor.
     */

    public SourcesPlugin() {
        this.displayName = "Sources";
        this.iframeLocation = URI.create("/admin/sources/index.html");
        List<String> apps = new ArrayList<String>();
        apps.add("catalog-app");
        this.setAssociations(apps);
    }
}

```

2. Configure as shown with a name, URI, and any dependency applications.
3. Register the application with Blueprint through a `blueprint.xml` file.

`blueprint.xml`

```

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.osgi.org/xmlns/blueprint/v1.0.0
        http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <bean id="appModule" class="org.codice.ui.admin.applications.ApplicationModule"
    "></bean>

    <service interface="org.codice.ddf.ui.admin.api.module.AdminModule" ref=
    "appModule" />

</blueprint>

```

4. Create application to use this configuration.

Including KAR Files

Sometimes a developer may need to include data or configuration file(s) in a KAR file. An example of this would be a properties file for the JDBC connection properties of a catalog provider.

It is recommended that:

- Any data/configuration files be placed under the `src/main/resources` directory of the maven project. (Sub-directories under `src/main/resources` can also be used, e.g., `etc/security`)
- The maven project's pom file should be updated to attach each data/configuration file as an artifact (using the `build-helper-maven-plugin`)

- Add each data/configuration file to the KAR file by using the `<configfile>` tag in the KAR's `features.xml` file

40.3.5. Assuring Authenticity of Bundles and Applications

DDF Artifacts in the JAR file format (such as bundles or DDF applications packaged as KAR files) can be signed and verified using the tools included as part of the Java Runtime Environment.

Prerequisites

To work with Java signatures, a keystore/truststore is required. For testing or trial purposes DDF can sign and validate using a self-signed certificate, generated with the keytool utility. In an actual installation, a certificate issued from a trusted Certificate Authority will be used.

Additional documentation on keytool can be found at [Keytool home](#).

Using keytool to generate a self-signed certificate keystore

```
~ $ keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass
password -validity 360 -keysize 2048
What is your first and last name?
[Unknown]: Nick Fury
What is the name of your organizational unit?
[Unknown]: Marvel
What is the name of your organization?
[Unknown]: SHIELD
What is the name of your City or Locality?
[Unknown]: New York
What is the name of your State or Province?
[Unknown]: NY
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=Nick Fury, OU=SHIELD, O=Marvel, L="New York", ST=NY, C=US correct?
[no]: yes
Enter key password for <selfsigned>
    (RETURN if same as keystore password):
Re-enter new password:
```

Siging a JAR/KAR

Once a keystore is available, the JAR can be signed using the `jarsigner` tool.

Additional documentation on jarsigner can be found at [Jarsigner](#).

Using jarsigner to sign a KAR

```
~ $ jarsigner -keystore keystore.jks -keypass shield -storepass password catalog-app-
2.5.1.kar selfsigned
```

Verifying a JAR/KAR

The jarsigner utility is also used to verify a signature in a JAR-formatted file.

Using jarsigner to verify a file

```
~ $ jarsigner -verify -verbose -keystore keystore.jks catalog-app-2.5.1.kar
    9447 Mon Oct  6 17:05:46 MST 2014 META-INF/MANIFEST.MF
    9503 Mon Oct  6 17:05:46 MST 2014 META-INF/SELFSIGN.SF

[... section abbreviated for space]

smk      6768 Wed Sep 17 17:13:58 MST 2014 repository/ddf/catalog/security/catalog-
security-logging/2.5.1/catalog-security-logging-2.5.1.jar
s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore
i = at least one certificate was found in identity scope
jar verified.
```

Note the last line: *jar verified*. This indicates that the signatures used to sign the JAR (or in this case, KAR) were valid according to the trust relationships specified by the keystore.

Appendix A: Application Reference

Installation and configuration details by application.

A.1. Catalog UI Reference

The Catalog UI application provides a metadata search and discovery, resource retrieval and workspace management with a 3d or optional 2d map visualization.

A.1.1. Catalog UI Prerequisites

To use the Catalog UI application, the following applications/features must be installed:

- Platform
- Catalog

A.1.2. Installing Catalog UI

Catalog UI is included with a standard installation.

A.1.3. Configuring Catalog UI

From the Admin Console, the following configurations are available from a standard installation.

Table 166. Catalog UI Configurations

Configuration	Configuration ID	Description
Catalog UI Search	org.codice.ddf.catalog.ui.config	Catalog UI Search
Email Notifier	org.codice.ddf.catalog.ui.query.monitor.email.EmailNotifier	Email Notifier
Workspace Query Monitor	org.codice.ddf.catalog.ui.query.monitor.impl.WorkspaceQueryService	Workspace Query Monitor
Workspace Security	org.codice.ddf.catalog.ui.security	Workspace security

Table 167. Catalog UI Search

Name	ID	Type	Description	Default Value	Required
Result Count	resultCount	Integer	Specifies the number of results to request from each source	250	true
Imagery Providers	imageryProviders	String	List of imagery providers to use. Valid types are: OSM (OpenStreetMap), AGM (ArcGisMap), BM (BingMap), WMS (WebMapService), WMT (WebMapTile), TMS (TileMapService), GE (GoogleEarth). Example: {"type": "WMS", "url": "http://example.com", "layers": ["layer1", "layer2"], "parameters": {"FORMAT": "image/png", "VERSION": "1.1.1"}, "alpha": 0.5}		false
Terrain Provider	terrainProvider	String	Terrain provider to use for height data. Valid types are: CT (CesiumTerrain), AGS (ArcGisImageServer), VRW (VRTheWorld). Example: {"type": "CT", "url": "http://example.com"}	{ "type": "CT", "url": "http://assets.agi.com/stk-terrain/tilesets/world/tiles" }	false
Map Projection	projection	String	Projection of imagery providers	EPSG:4326	false
Bing Maps Key	bingKey	String	Bing Maps API key. This should only be set if you are using Bing Maps Imagery or Terrain Providers.		false

Name	Id	Type	Description	Default Value	Required
Connection Timeout	<code>timeout</code>	Integer	Specifies the client-side connection timeout in milliseconds.	300000	false
Source Poll Interval	<code>sourcePollInterval</code>	Integer	Specifies the interval to poll for sources in milliseconds.	60000	true
Show Sign In	<code>signIn</code>	Boolean	Allow Sign In to Search UI and welcome notice. Enable this if the Search UI is protected.	true	false
Show Tasks	<code>task</code>	Boolean	Show task menu area for long running actions.	false	false
Show Gazetteer	<code>gazetteer</code>	Boolean	Show gazetteer for searching place names.	true	false
Show Uploader	<code>ingest</code>	Boolean	Show upload menu for adding new record.	true	false
Use External Authentication	<code>externalAuthentication</code>	Use an external authentication point, such as IdP	Boolean	false	false
Disable Cache	<code>cacheDisabled</code>	Locally cached results will not be returned in search results.	Boolean	false	false
Type Name Mapping	<code>typeNameMapping</code>	String	Use an external authentication point, such as IdP.		false
Read Only Metocard Attributes	<code>readOnly</code>	String	List of metocard attributes that are read-only. NOTE: the provided values will be evaluated as JavaScript regular expressions when matched against metocard attributes.	<code>checksum\$, checksum-algorithm\$ id\$, metadata\$, metocard-type\$, source-id\$, metocard\\., version\\., validation\\.</code>	false
Summary Metocard Attributes	<code>summaryShow</code>	String	List of metocard attributes to display in the summary view.		false

Name	Id	Type	Description	Default Value	Required
Result Preview Metocard Attributes	<code>resultShow</code>	String	List of metocard attributes to display in the result preview.		false
Attribute Aliases	<code>attributeAliases</code>	String	List of attribute aliases. Example 'title=Title'		false
Hidden Attributes	<code>hiddenAttributes</code>	String	List of attributes to be hidden. NOTE: the provided values will be evaluated as JavaScript regular expressions when matched against metocard attributes.	<code>metocard\\.modifie d\$, metocard\\ .created\$ metocard- tags\$, metacard d\\\.owner\$ validation- errors\$, valida tion- warnings\$, ^sortOrder \$, sortField\$, cql\$ enterprise\$, po licing\$, securi ty\\\.acces s- individual \$, failed- validators- warnings\$, fail ed- validators- - errors\$, sourc es\$, federati on\$, metocard\\ .sharing\$, cache d\$</code>	false
Query Schedule Frequencies	<code>scheduleFrequencyList</code>	Long	Custom list of schedule frequencies in seconds. This will override the frequency list in the query schedule tab. Leave this empty to use the frequency list on the Catalog UI.	1800,3600, 7200,14400 ,28800,576 00,86400	true

Table 168. Email Notifier

Name	Id	Type	Description	Default Value	Required
Subject	<code>subjectTemplate</code>	String	Set the subject line template.	Workspace '%[attribute=title]' notification	true

Name	Id	Type	Description	Default Value	Required
Body	<code>bodyTemplate</code>	String	Set the body template.	The workspace '%[attribute=title]' contains up to %[hitCount] results. Log in to see results https://localhost:8993/search/catalog/#workspaces/%[attribute=id].	true
Mail Server	<code>mailHost</code>	String	Set the hostname of the mail server.	localhost	true
From Address	<code>fromEmail</code>	String	Set the 'from' email address.	<code>donotreply@test.com</code>	true

Table 169. Workspace Query Monitor

Name	Id	Type	Description	Default Value	Required
Query Timeout	<code>queryTimeoutMinutes</code>	Long	Set the number of minutes to wait for query to complete.	5	true
Notification Time Interval	<code>queryTimeInterval</code>	Integer	Set the Relative Time Search (past X minutes up to 24 hours). Note: This will query for results from the interval to the time the query is sent out.	1440	true
Email Subscription Interval	<code>cronString</code>	String	Email Subscription Interval (Cron Expression)	0 0 0 * * ?	true

Table 170. Workspace Security

Name	Id	Type	Description	Default Value	Required
System User Attribute	systemUserAttribute	String	The name of the attribute to determine the system user.	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role	true
System User Attribute Value	systemUserAttributeValue	String	The value of the attribute to determine the system user.	admin	true

A.2. Admin Reference

The Admin Application contains components that are integral for the installation and configuration DDF applications.

It contains various services and interfaces that allow administrators more control over their systems and enhances administrative capabilities when installing and managing DDF.

The Admin application contains an application service that handles all operations that are performed on applications. This includes adding, removing, starting, stopping, and showing status.

A.2.1. Admin Application Prerequisites

None.

A.2.2. Installing the Admin Application

The Admin application is installed by default with a standard installation.

A.2.3. Configuring the Admin Application

1. Navigate to the Admin Console.
2. Select the **Admin** application.
3. Select the **Configuration** tab.

Table 171. Admin Configurations

Name	Property	Description
Admin Configuration Policy	org.codice.ddf.admin.config.policy.AdminConfigPolicy	Admin Configuration Policy
Admin UI Configuration	org.codice.admin.ui.configuration	null

Table 172. Admin Configuration Policy

Name	Id	Type	Description	Default Value	Required
Feature and App Permissions	<code>featurePolicies</code>	String	When enabled, the desired features or apps will only be modifiable and viewable to users with the set attributes. The entry should be the format of: <code>feature name/app name = "user attribute name=user attribute value"</code>		false
Configuration Permissions	<code>servicePolicies</code>	String	When enabled, the desired service will only be modifiable and viewable to users with the set attributes. The entry should be the format of: <code>configuration ID = "user attribute name=user attribute value"</code>	null	false

Table 173. Admin UI Configuration

Name	Id	Type	Description	Default Value	Required
Enable System Usage message	<code>systemUsageEnabled</code>	Boolean	Turns on a system usage message, which is shown when the Admin Application is opened.	false	true
System Usage Message Title	<code>systemUsageTitle</code>	String	A title for the system usage message when the application is opened.		true
System Usage Message	<code>systemUsageMessage</code>	String	A system usage message to be displayed to the user each time the user opens the application.		true
Show System Usage Message once per session	<code>systemUsageOncePerSession</code>	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 admin page is opened or refreshed.	true	true
Header	<code>header</code>	String	Specifies the header text to be rendered on the Administrator Console.		true
Footer	<code>footer</code>	String	Specifies the footer text to be rendered on the Administrator Console.		true
Style	<code>style</code>	String	Specifies the style of the Header and Footer.	green	true

Name	Id	Type	Description	Default Value	Required
Text Color	<code>textColor</code>	String	Specifies the text color of the Header and Footer.	banner-text-white	true
Ignored Installer Applications	<code>disabledInStallerApps</code>	String	Comma delimited list (appName, appName2, ...appNameN) of applications that will be disabled in the installer.	admin-app,platform-app	null

A.3. Catalog Reference

The Catalog provides a framework for storing, searching, processing, and transforming information.

Clients typically perform create, read, update, and delete (CRUD) operations against the Catalog.

At the core of the Catalog functionality is the Catalog Framework, which routes all requests and responses through the system, invoking additional processing per the system configuration.

A.3.1. Catalog Application Prerequisites

To use the Catalog Application, the following applications/features must be installed:

- Platform

A.3.2. Installing the Catalog Application

The Catalog is pre-installed with a standard installation.

A.3.3. Configuring the Catalog Application

These configurations are available through the Catalog application.

Table 174. Catalog Available Configurations

Configuration	ID	Description
Backup Post-Ingest Plugin	<code>plugin.backup</code>	Backup Post-Ingest Plugin Configuration
Catalog OpenSearch Federated Source	<code>OpenSearchSource</code>	DDF OpenSearch Federated Source
Catalog Policy Plugin	<code>org.codice.ddf.catalog.security.CatalogPolicy</code>	Catalog Policy Plugin
Catalog Standard Framework	<code>ddf.catalog.CatalogFrameworkImpl</code>	DDF Local site
Expiration Date Pre-Ingest Plugin	<code>org.codice.ddf.catalog.plugin.expiration.ExpirationDatePlugin</code>	Catalog pre-ingest plugin to set an expiration date on metacards
Metocard Attribute Security Policy Plugin	<code>org.codice.ddf.catalog.security.policy.metocard.MetocardAttributeSecurityPolicyPlugin</code>	Metocard Attribute Security Policy Plugin
Schematron Validation Services	<code>ddf.services.schematron.SchematronValidationService</code>	Schematron Validators
XML Attribute Security Policy Plugin	<code>org.codice.ddf.catalog.security.policy.xml.XmlAttributeSecurityPolicyPlugin</code>	XML Attribute Security Policy Plugin

Configuration	ID	Description
Xml Query Transformer	<code>ddf.catalog.transformer.xml.XmlResponseQueueTransformer</code>	XML Response Queue Transformer

A.4. GeoWebCache Reference

GeoWebCache enables a server providing a map tile cache and tile service aggregation.

WARNING

The GeoWebCache application is currently in an EXPERIMENTAL status and should not be installed on a security-hardened installation.

GeoWebCache enables a server providing a tile cache and tile service aggregation. See ([GeoWebCache](#)) for more information. This application also provides an administrative plugin for the management of GeoWebCached layers. GeoWebCache also provides a user interface for previewing, truncating, or seeding layers at <https://localhost:8993/geowebcache/>.

A.4.1. GeoWebCache Application Prerequisites

None.

A.4.2. Installing GeoWebCache

The GeoWebCache application is **not** installed by default.

To install:

1. Navigate to the Admin Console.
2. Select **Manage**.
3. Select the GeoWebCache application **Start** button. The application will move to **Active Applications** on startup.
4. Select **Back** in Admin Console.

A.4.3. Configuring GeoWebCache

GeoWebCache can be configured to cache layers locally, using the following procedures.

Adding GeoWebCache Layers

Add layers to the local cache:

1. Navigate to the Admin Console.
2. Select the GeoWebCache Application.
3. Select the **GeoWebCache Layers** tab.
4. Click the **Add** button.
5. Enter the data in the fields provided.
6. If necessary, click the **Add** button to add additional MIME types.
7. If necessary, click the **Add** button to add additional WMS Layer Names.

Table 175. Add Layer

Name	Property	Type	Description	Default Value	Required
Name		String			no
Mime Formats		String			yes
URL		URI			yes
WMS Layer Name		String			yes

Editing GeoWebCache Layers

1. Navigate to the Admin Console.
2. Select the GeoWebCache application.
3. Navigate to the **GeoWebCache Layers** tab.
4. Click the **Name** field of the layer to edit.

Removing GeoWebCache Layers

1. Click the **Delete** icon at the end of the row of the layer to be deleted.

Configuring GWC Disk Quota

Storage usage for a GeoWebCache server is managed by a `diskquota.xml` file with configuration details to prevent image-intensive data from filling the available storage.

To view the disk quota XML representative: <https://localhost:8993/geowebcache/rest/diskquota.xml>

To update the disk quota, a client can post a new XML configuration: `curl -v -k -XPUT -H "Content-type: text/xml" -d @diskquota.xml "https://localhost:8993/geowebcache/rest/diskquota.xml"`

Example `diskquota.xml`

```
<gwcQuotaConfiguration>
  <enabled>true</enabled>
  <diskBlockSize>2048</diskBlockSize>
  <cacheCleanUpFrequency>5</cacheCleanUpFrequency>
  <cacheCleanUpUnits>SECONDS</cacheCleanUpUnits>
  <maxConcurrentCleanUps>5</maxConcurrentCleanUps>
  <globalExpirationPolicyName>LFU</globalExpirationPolicyName>
  <globalQuota>
    <value>100</value>
    <units>GiB</units>
  </globalQuota>
  <layerQuotas/>
</gwcQuotaConfiguration>
```

See [Disk Quotas](#) for more information on configuration options for disk quota.

Configuring the Standard Search UI for GeoWebCache

Add a new Imagery Provider in the Admin Console:

1. Navigate to <https://<localhost>:8993/admin>.
2. Select **Configuration** tab.
3. Select **Standard Search UI** configuration.
4. Click the **Add** button next to **Imagery Providers**
5. Enter configuration for Imagery Provider in new textbox:
`{"type": "WMS", "url": "https://<HOSTNAME>:<PORT>/geowebcache/service/wms", "layers": ["states"], "parameters": {"FORMAT": "image/png"}, "alpha": 0.5}`
7. Set the Map Projection to [EPSG:900913](#) or [EPSG:4326](#). (GeoWebCache supports either of these projections.)

NOTE

Currently, GeoWebCache only supports WMS 1.1.1 and below. If the version number is not specified in the imagery provider, DDF will default to version [1.3.0](#), and OpenLayers will not project the image tiles properly. Thus, the version [1.1.1](#) must be specified when using [EPSG:4326](#) projections.

```
{"type": "WMS", "url": "https://<HOSTNAME>:<PORT>/geowebcache/service/wms", "layers": ["states"], "parameters": {"FORMAT": "image/png", "VERSION": "1.1.1"}, "alpha": 0.5}
```

A.5. Platform Reference

The Platform application is considered to be a core application of the distribution. The Platform application provides the fundamental building blocks that the distribution needs to run. These building blocks include subsets of:

- [Karaf](#)
- [CXF](#)
- [Camel](#)

A [Command Scheduler](#) is also included as part of the Platform application to allow users to schedule Command Line Shell Commands.

A.5.1. Platform Application Prerequisites

None.

A.5.2. Installing Platform

The Platform application is installed by default with a standard installation.

A.5.3. Configuring Platform

Table 176. Platform Available Configurations

Configuration	ID	Description
Landing Page	org.codice.ddf.distribution.landing-page.properties	Starting page for users to interact with DDF.
Logging Service	org.codice.ddf.platform.logging.LoggingService	Service that logs system activities.
MIME Custom Types	DDF_Custom_Mime_Type_Resolver	DDF Custom Mime Types
Metrics Reporting	MetricsReporting	Metrics Reporting
Platform Command Scheduler	ddf.platform.scheduler.Command	
Platform UI Configuration	ddf.platform.ui.config	Global User Interface configurations used across the applications. Contains configuration for banners and other generic ui components.
HTTP Response Security	org.codice.ddf.security.response.filter.ResponseHeaderConfig	Instructions for the client browser detailing which location and/or which type of resources may be loaded.

A.6. Registry Reference

Registry contains the base registry components, plugins, sources and interfaces needed for DDF to function as a registry connecting multiple nodes.

A.6.1. Registry Prerequisites

To use the Registry, the following apps/features must be installed:

- Catalog
- Admin
- Spatial
- Platform
- Security

A.6.2. Installing Registry

Registry is not installed by default with a standard installation.

To install:

1. Navigate to the **Admin Console**.
2. Click the **Manage** applications button.
3. Click the **Install** icon.
4. Registry will move to **Active Applications** upon startup.

A.6.3. Configuring Registry

These configurations are available from the Registry application in the Admin Console.

Table 177. Registry Configurations

Name	Configuration ID	Description
CSW Registry Store	Csw_Registry_Store	Registry CSW Store
Registry Federation Admin Service	Registry_Federation_Admin_Service	Registry Federation Admin Service
Registry Policy Plugin	org.codice.ddf.registry.policy.RegistryPolicyPlugin	Registry Policy Plugin
Registry Source Configuration Handler	Registry_Configuration_Event_Handler	Configurable values for the registry source configuration handler

A.6.4. Customizing Registry Fields

All the fields that appear in a registry node are customizable. This is done through a JSON

configuration file located at `<DDF_HOME>/etc/registry/registry-custom-slots.json` that defines the registry fields. In this file there are JSON objects that relate to each part of the edit registry modal. These objects are

- General
- Service
 - ServiceBinding
- Organization
- Person (Contact)
- Content (Content Collection)

Each of the objects listed above is a JSON array of field objects that can be modified. There are some other objects in the JSON file like **PersonName**, **Address**, **TelephoneNumber**, and **EmailAddress** that should not be modified.

Table 178. Field Properties

Property Key	Required	Property Value
key	yes	The string value that will be used to identify this field. Must be unique within field grouping array. This value is what will show up in the generated EBRIM xml.
displayName	yes	The string name that will be displayed in the edit node dialog for this field
description	yes	A brief description of what the field represents or is used for. Shown when user hovers or click the question mark icon for the field.
value	no	The initial or default value of the field. For most cases this should be left as an empty array or string.
type	yes	Identifies what type of field this is. Value must be one of string , date , number , boolean , point , or bounds
required	no	Indicates if this field must be filled out. Default is false . If true an asterisk will be displayed next to the field name.

Property Key	Required	Property Value
possibleValues	no	An array of values that could be used for this field. If multiValued=true this list will be used for suggestions for autocomplete. If multiValued=false this list will be used to populate a dropdown.
multiValued	no	Flag indicating if this field accepts multiple values or not. Default is false.
isSlot	no	Indicates that this field represents a slot value in the EBRIM document. If this is false the key must match a valid EBRIM attribute for the parent object. Default is true.
advanced	no	A flag indicating if this field should be placed under the Advanced section of the edit modal ui. Default is false.
regex	no	A regular expression for validating users input.
regexMessage	no	A message to show the user if the regular expression test fails.
isGroup, constructTitle	N/A	These fields are used for nesting objects and should not be modified

A.6.5. Using Registry

The **Node Information** and **Remote Registries** tabs appear in both the Registry application and the Catalog application.

Configuring Identity Node

The "Identity Node" is the local DDF instance. Configure the information to share with other registries/nodes.

1. Navigate to **Registry** (or **Catalog**) application.
2. Navigate to **Node Information** tab.
3. Click the name of the identity node.
4. Complete all *required* and any desired *optional* fields.
 - a. Add any desired **service bindings** under the **Services** tab.

5. Click **Save**.

Table 179. General Information Tab

Field	Description	Type	Required
Node Name	This node's name as it should appear to external systems	string	yes
Node Description	Short description for this node	string	yes
Node Version	This node's Version	string	yes
Security Attributes	Security attributes associated with this node.	String	
Last Updated	Date this entry's data was last updated	Date	
Live Date	Date indicating when this node went live or operational	Date	
Custom Fields	click Add button to add custom fields	Configurable	no
Associations	click Add button to add associations	Configurable	no

Table 180. Services

Field	Description	Type	Required
Service Name	This service name	string	
Service Description	Short description for this service	string	
Service Version	This service version	string	
Service Type	Identifies the type of service this is by a URN.	string	
Bindings (Click Add to add a service binding)			
Binding Name	This binding name	String	yes
Binding Description	Short description for this binding	String	
Binding Version	This binding version		
Access URL	The url used to access this binding		
Service Binding Type	The binding type for the service		
URL Property Key	Property that the accessURI value should be put into for source creation		
Custom Fields	click Add button to add custom fields	Configurable	no
Associations	click Add button to add associations	Configurable	no

Table 181. Organizations Tab (click **Add** to add an organization)

Field	Description	Type	Required
Organization Name	This organization's name	string	yes
Address	This organization's primary address	Expand to enter address information	yes
TelephoneNumber	Primary contact number for this organization		no
Email	Primary contact email for this organization		no
Custom Fields	click Add button to add custom fields	Configurable	no
Associations	click Add button to add associations	Configurable	no

Table 182. Contacts (click **Add** button to add contact info)

Field	Description	Type	Required
Contact Title	Contact Title	String	yes
Contact First Name	Contact First Name	String	yes
Contact Last Name	Contact Last Name	String	yes
Address	Address for listed contact	String	minimum one
Phone number	Contact phone number		minimum one
Email	Contact email	String	minimum one
Custom Fields	click Add button to add custom fields	Configurable	no
Associations	click Add button to add associations	Configurable	no

Table 183. Collections (Click **Add** to add Content Collection(s))

Field	Description	Type	Required
Content Name	Name for this metadata content	string	yes
Content Description	Short description for this metadata content	string	no
Content Object Type	The kind of content object this will be. Default value should be used in most cases.	string	yes
Custom Fields	click Add button to add custom fields	Configurable	no
Associations	click Add button to add associations	Configurable	no

Adding a Service Binding to a Node

Advertise the methods other nodes use to connect to the local DDF instance.

1. Navigate to Admin Console.
2. Select Registry or Catalog.
 - a. (**Node Information** tab is editable from either application.)
3. Click the name of the desired local node.
4. Click the **Services** tab.
5. Click **Add** to add a service.
6. Expand new Service.
7. Enter Service name and details.
8. Click **Add** to add binding.
9. Select Service Binding type.
 - a. Select one of the defaults or *empty* for a custom service binding.
 - b. If selecting *empty*, fill in all required fields.
10. Click Save.

Publishing to Others

Send details about the local DDF instance to other nodes.

1. Navigate to the **Remote Registries** tab in either Registry or Catalog application.
2. Click **Add** to add a remote registry.
3. Enter Registry Service (CSW) Url.
4. Confirm **Allow Push** is checked.
5. Click **Add** to save the changes.
6. Navigate to the **Sources** Tab in Catalog App
7. Click desired node to be published.
8. Under **Operations**, click the *Publish to ... * link that corresponds to the desired registry.

Subscribing to Another Node

Receive details about another node.

1. Navigate to the **Remote Registries** tab in either Registry or Catalog application.
2. Click **Add** to add a remote registry.
3. Add the URL to access node.
4. Enter any needed credentials in the Username/password fields.
5. Click **Save/Add**.

Editing a Subscription

Update the configuration of an existing subscription.

1. Navigate to the **Remote Registries** tab in either Registry or Catalog application.
2. Click the name of the desired subscription.
3. Make changes.
4. Click **Save**.

Deleting a Subscription

Remove a subscription.

1. Click the **Delete** icon at the top of the **Remote Registries** tab.
2. Check the boxes of the Registry Nodes to be deleted.
3. Select the **Delete** button.

A.7. Security Reference

The Security application provides authentication, authorization, and auditing services for the DDF. These services comprise both a framework that developers and integrators can extend as well as a reference implementation that meets security requirements.

This section documents the installation, maintenance, and support of this application.

Applications Included in Security

- Security CAS
- Security Core
- Security Encryption
- Security IdP
- Security PEP
- Security PDP
- Security STS

A.7.1. Security Prerequisites

To use the Security application, the following applications/features must be installed:

- Platform

A.7.2. Installing Security

Security is included with a standard installation.

A.7.3. Configuring Security

From the Admin Console, the following configurations are available from a standard installation.

Table 184. Security Configurations

Configuration	Configuration ID	Description
Login Page	<code>org.codice.ddf.security.handler.guest.configuration</code>	Login Page
SAML NameID Policy	<code>ddf.security.service.SecurityManager</code>	SAML NameID Policy
STS Server Token Endpoint	<code>ddf.security.sts.StsStaticService</code>	STS Server Token Endpoint
Security SOAP Guest Interceptor	<code>org.codice.ddf.security.interceptor.GuestInterceptor</code>	Security SOAP Guest Interceptor

Configuration	Configuration ID	Description
Security STS Address Provider	<code>ddf.security.sts.address.provider</code>	STS Address Provider
Security STS Client	<code>ddf.security.sts.client.configuration</code>	STS Client Configuration Settings
Security STS Guest Claims Handler	<code>ddf.security.sts.guestclaims</code>	Guest Claims Handler Configuration
Security STS Guest Validator	<code>ddf.security.sts.guestvalidator</code>	Guest Validator Configuration
Security STS PKI Token Validator	<code>org.codice.ddf.security.validator.pki</code>	STS PKI Token Validator Configuration.
File Based Claims Handler	<code>org.codice.ddf.security.sts.claims.property.PropertyFileClaimsHandler</code>	File Based Claims Handler.
Security STS Server	<code>ddf.security.sts</code>	STS Configuration
Security STS WSS	<code>ddf.security.sts.wss.configuration</code>	STS WSS Configuration Settings
Security AuthZ Realm	<code>ddf.security.pdp.realm.AuthzRealm</code>	AuthZ Security Settings
Session	<code>org.codice.ddf.security.filter.login.Session</code>	Session
Web Context Policy Manager	<code>org.codice.ddf.security.policy.context.impl.PolicyManager</code>	Web Context Security Policies

Table 185. Web Context Policy Manager

Name	Id	Type	Description	Default Value	Required
Context Realms	<code>realms</code>	String	List of realms supporting each context. Karaf is provided by default. Example: <code>:/karaf</code> "	<code>/karaf</code>	true

Name	Id	Type	Description	Default Value	Required
Authentication Types	authenticationTypes	String	List of authentication types required for each context. List of default valid authentication types are: SAML, BASIC, PKI, GUEST. Example: /context=AUTH1 AUTH2 AUTH3"	/=SAML GUEST,/admin=SAML basic,/system=basic,/solr=SAML PKI basic,/sources=SAM basic,/security-config=SAM basic	true
Required Attributes	requiredAttributes	String	List of attributes required for each Web Context. Example: /context={role=role1;type=type1}	/=/admin={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin},/solr={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin},/system={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin},/security-config={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin}	true

Name	Id	Type	Description	Default Value	Required
White Listed Contexts	<code>whiteListContexts</code>	String	List of contexts that will not use security. Note that sub-contexts to ones listed here will also skip security, unless authentication types are provided for it. For example: if /foo is listed here, then /foo/bar will also not require any sort of authentication. However, if /foo is listed and /foo/bar has authentication types provided in the 'Authentication Types' field, then that more specific policy will be used.	<code> \${org.codice.ddf.system.rootContext}/SecurityTokenService,\${org.codice.ddf.system.rootContext}/internal/metrics,\${org.codice.ddf.system.rootContext}/saml,\${org.codice.ddf.system.rootContext}/saml,\${org.codice.ddf.system.rootContext}/idp,\${org.codice.ddf.system.rootContext}/platfrom/config/ui/login</code>	true

Table 186. Session

Name	Id	Type	Description	Default Value	Required
Session Timeout (in minutes)	<code>expirationTime</code>	Integer	The number of minutes after a session has been inactive that it should be invalidated.	31	true

A.8. Solr Catalog Reference

DDF uses Solr for data storage, by default.

A.8.1. Solr Catalog Prerequisites

To use the Solr Catalog Application, the following apps/features must be installed:

- Platform
- Catalog

A.8.2. Installing Solr Catalog

The Solr Catalog Application is installed by default with a standard installation.

A.8.3. Configuring Solr Catalog

The Solr Catalog Application includes the Solr Catalog Provider feature.

Solr Catalog Provider Default Configurations

These configurations are available by default on a standard installation of Solr Catalog Provider.

Table 187. Catalog Solr Provider Default Configurations

Name	Id	Type	Description	Default Value	Required
Force Auto Commit	<code>forceAutoCommit</code>	Boolean	WARNING: Performance Impact. Only in special cases should auto-commit be forced. Forcing auto-commit makes the search results visible immediately.	false	true
Disable Text Path indexing	<code>disableTextPath</code>	Boolean	Disables the ability to make Text Path queries by disabling the Text Path index. Disabling Text Path indexing typically increases ingest performance.	false	true

A.9. Spatial Reference

The 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.

A.9.1. Spatial Prerequisites

To use the Spatial Application, the following apps/features must be installed:

- Platform
- Catalog

A.9.2. Installing Spatial

The Spatial application is installed by default with a standard installation.

A.9.3. Configuring Spatial

These configurations are available from the Spatial application in the Admin Console.

Spatial Default Configurations

These configurations are available by default on a standard installation of Spatial.

Table 188. Spatial Default Configurations

Name	Configuration ID	Description
CSW Federation Profile Source	Csw_Federation_Profile_Source	DDF's full fidelity CSW Federation Profile. Use this when federating to a DDF based system.
CSW Specification Profile Federated Source	Csw_Federated_Source	CSW Specification Profile Federated Source should be used when federating to an external CSW service.
GMD CSW ISO Federated Source	Gmd_Csw_Federated_Source	CSW Federated Source using the Geographic MetaData (GMD) format (ISO 19115:2003)
Metocard to WFS Feature Map	org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetocardMapper	Metocard to WFS Feature Map
Spatial KML Endpoint	org.codice.ddf.spatial.kml.endpoint.KmlEndpoint	Spatial KML Endpoint

Name	Configuration ID	Description
Spatial KML Style Map Entry	<code>org.codice.ddf.spatial.kml.style</code>	Spatial KML Style Map Entry
WFS 2.0.0 Federated Source	<code>Wfs_v2_0_0_Federated_Source</code>	WFS 2.0.0 Federated Source
WFS v1.0.0 Federated Source	<code>Wfs_v1_0_0_Federated_Source</code>	WFS v1.0.0 Federated Source

Offline Gazetteer Service

In the Spatial Application, you have the option to install a feature called `offline-gazetteer`. This feature enables you to use an offline index of GeoNames data (as an alternative to the GeoNames Web service enabled by the `webservice-gazetteer` feature) to perform searches via the gazetteer search box in the Search UI.

To use the offline gazetteer, you will need to create an index using the `geonames:update` command.

Spatial GeoNames Console Commands

The `geonames` commands provide the ability to interact with a local GeoNames index. The local GeoNames index is used by the `offline-gazetteer` feature, which is an optional feature available in this application and is explained above. Note that these commands are only available if the `offline-gazetteer` feature is installed.

Table 189. GeoNames Command Descriptions

Command	Description	Usage
update	<p>Adds new entries to an existing local GeoNames index. Entries can be manually downloaded from http://download.geonames.org/export/dump, where the absolute path of the file would be passed as an argument to the command (ex. /Users/johndoe/Downloads/AU.zip). Currently .txt and .zip files are supported for manual entries. Entries can also be automatically downloaded from http://download.geonames.org/export/dump by passing the country code as an argument to the command (ex. AU) which will add the country to the local GeoNames index. The full list of country codes available can be found in http://download.geonames.org/export/dump/countryInfo.txt. Using the argument "all" will download all of the current country codes (this process may take some time). In addition to country codes, GeoNames also provides entries for cities based on their population sizes. The arguments "cities1000", "cities5000", and "cities15000" will add cities to the index that have at least 1000, 5000, or 15000 people respectively. The index location can be configured via the Admin Console. By default, the index location is <code>data/geonames-index</code>. If you specify a relative path, it is relative to the location of the unzipped DDF distribution. You may specify an absolute path if you want the index to be located somewhere else. The <code>-c</code> or <code>--create</code> flag can be added to create a new GeoNames index. This will overwrite any existing index at the location specified in the Admin Console. The new index will be filled with the entries in the file you pass to the command. You must create an index before you can add additional entries to it (i.e. running the command without the <code>-c</code> or <code>--create</code> flag).</p>	geonames:update

A.10. Search UI Reference

The Standard Search UI is a user interface that enables users to search a catalog and associated sites for content and metadata.

A.10.1. Search UI Prerequisites

To use the Search UI application, the following applications/features must be installed:

- Platform
- Catalog

A.10.2. Installing Search UI

The Search UI application is installed by default with a standard installation.

A.10.3. Configuring Search UI

These configurations are available from the Search UI application in the Admin Console.

Table 190. Search UI Configurations

Configuration	ID	Description
Search UI Endpoint	<code>org.codice.ddf.ui.search.standard.endpoint</code>	Search UI Endpoint
Search UI Redirect	<code>org.codice.ddf.ui.searchui.filter.RedirectServlet</code>	Search UI Redirect
Simple Search UI	<code>org.codice.ddf.ui.search.simple.properties</code>	Simple Search UI
Standard Search UI	<code>org.codice.ddf.ui.search.standard.properties</code>	Standard Search UI

A.11. Resource Management Reference

The Resource Management Application provides administrative functionality to monitor and manage data usage on the system. This application allows an administrator to:

- View data usage.
- Set limits on users.
- View and terminate searches that are in progress.

A.11.1. Resource Management Prerequisites

To use the Resource Management Application, the following apps/features must be installed:

- Platform
- Security
- Admin
- Catalog

A.11.2. Installing Resource Management

The Resource Management application is not installed by default with a standard installation.

To Install:

1. Navigate to Admin Console.
2. Select **Manage** applications.
3. Select **Resource Management** application.
4. Select **Start**.
5. Resource Management application moves to **Active Applications** upon start.

A.11.3. Configuring Resource Management

These configurations are available from the Resource Management application in the Admin Console.

Table 191. Resource Management Configurations

Name	Configuration ID	Description
None		

A.11.4. Resource Management Data Usage Tab

View data usage and configure users' data limits and reset times for those limits.

A.11.5. Resource Management Queries Tab

View and cancel actively running queries.

Appendix B: Application Whitelists

- camel
- cxf
- Karaf Console Commands
- Jetty

B.1. Admin Whitelist

The following packages have been exported by the Admin application and are approved for use by third parties:

- `org.codice.ddf.ui.admin.api`
- `org.codice.ddf.ui.admin.api.module`
- `org.codice.ddf.ui.admin.api.plugin`
- `org.codice.ddf.admin.configuration.plugin`
- `org.codice.ddf.admin.application.service`
- `org.codice.ddf.admin.application.plugin`

B.2. Catalog Whitelist

The following packages have been exported by the Catalog application and are approved for use by third parties:

- `ddf.camel.component.catalog`
- `ddf.catalog`
- `ddf.catalog.cache`
- `ddf.catalog.data`
- `ddf.catalog.data.metocardtype`
- `ddf.catalog.event`
- `ddf.catalog.federation`
- `ddf.catalog.federation.impl`
- `ddf.catalog.filter`
- `ddf.catalog.filter.delegate`
- `ddf.catalog.impl.filter`
- `ddf.catalog.operation`
- `ddf.catalog.plugin`
- `ddf.catalog.plugin.groomer`
- `ddf.catalog.pubsub`
- `ddf.catalog.pubsub.tracker`
- `ddf.catalog.resource`

- `ddf.catalog.resource.data`
- `ddf.catalog.resource.impl`
- `ddf.catalog.resourceretriever`
- `ddf.catalog.service`
- `ddf.catalog.source`
- `ddf.catalog.transform`
- `ddf.catalog.transformer.api`
- `ddf.catalog.transformer.metacard.geojson`
- `ddf.catalog.util`
- `ddf.catalog.validation`
- `ddf.common`
- `ddf.geo.formatter`
- `ddf.util`
- `org.codice.ddf.endpoints`
- `org.codice.ddf.endpoints.rest`
- `org.codice.ddf.endpoints.rest.action`
- `org.codice.ddf.opensearch.query`
- `org.codice.ddf.opensearch.query.filter`

B.3. Platform Whitelist

The following packages have been exported by the Platform application and are approved for use by third parties:

- `ddf.action`
- `ddf.security`
- `ddf.security.assertion`
- `ddf.security.common.audit`
- `ddf.security.http`
- `ddf.security.permission`
- `ddf.security.policy.extension`
- `ddf.security.principal`
- `ddf.security.samlp`
- `ddf.security.service`
- `ddf.security.settings`
- `ddf.security.sts.client.configuration`
- `ddf.security.ws.policy`
- `ddf.security.ws.proxy`
- `org.codice.ddf.branding`
- `org.codice.ddf.configuration`
- `org.codice.ddf.configuration.admin`
- `org.codice.ddf.configuration.migration`
- `org.codice.ddf.configuration.persistence`

- `org.codice.ddf.configuration.persistence.felix`
- `org.codice.ddf.configuration.status`
- `org.codice.ddf.notifications.store`
- `org.codice.ddf.parser`
- `org.codice.ddf.parser.xml`
- `org.codice.ddf.platform.error.handler`
- `org.codice.ddf.platform.util`

B.4. Registry Whitelist

The following packages have been exported by the Registry Application and are approved for use by third parties:

None.

B.5. Security Whitelist

The following packages have been exported by the Security application and are approved for use by third parties:

- `ddf.security.assertion.impl`
- `ddf.security.common.util`
- `ddf.security.encryption`
- `ddf.security.expansion`
- `ddf.security.http.impl`
- `ddf.security.impl`
- `ddf.security.pdp.realm`
- `ddf.security.realm.sts`
- `ddf.security.samlp.impl`
- `ddf.security.service.impl`
- `ddf.security.soap.impl`
- `ddf.security.sts`
- `ddf.security.ws.policy.impl`
- `org.apache.cxf.sts.cache`
- `org.apache.cxf.sts.claims`
- `org.apache.cxf.sts.event.map`
- `org.apache.cxf.sts.event`
- `org.apache.cxf.sts.interceptor`
- `org.apache.cxf.sts.operation`
- `org.apache.cxf.sts.provider`
- `org.apache.cxf.sts.request`
- `org.apache.cxf.sts.service`
- `org.apache.cxf.sts.token.canceller`
- `org.apache.cxf.sts.token.delegation`

- org.apache.cxf.sts.token.provider
- org.apache.cxf.sts.token.realm
- org.apache.cxf.sts.token.renewer
- org.apache.cxf.sts.token.validator
- org.apache.cxf.sts
- org.codice.ddf.security.certificate.generator
- org.codice.ddf.security.certificate.keystore.editor
- org.codice.ddf.security.common
- org.codice.ddf.security.filter.authorization
- org.codice.ddf.security.filter.login
- org.codice.ddf.security.filter.websso
- org.codice.ddf.security.handler.api
- org.codice.ddf.security.handler.basic
- org.codice.ddf.security.handler.guest.configuration
- org.codice.ddf.security.handler.guest
- org.codice.ddf.security.handler.pki
- org.codice.ddf.security.handler.saml
- org.codice.ddf.security.interceptor
- org.codice.ddf.security.interceptor
- org.codice.ddf.security.policy.context.attributes
- org.codice.ddf.security.policy.context.impl
- org.codice.ddf.security.policy.context
- org.codice.ddf.security.servlet.logout
- org.codice.ddf.security.validator.username

B.6. Solr Catalog Whitelist

The following packages have been exported by the Solr Catalog application and are approved for use by third parties:

None.

B.7. Spatial Whitelist

The following packages have been exported by the Spatial application and are approved for use by third parties:

- net.opengis.cat.csw.v_2_0_2.dc.elements
- net.opengis.cat.csw.v_2_0_2.dc.terms
- net.opengis.cat.csw.v_2_0_2
- net.opengis.filter.v_1_1_0
- net.opengis.gml.v_3_1_1
- net.opengis.ows.v_1_0_0
- org.codice.ddf.spatial.geocoder.geonames

- org.codice.ddf.spatial.geocoder
- org.codice.ddf.spatial.geocoding.context
- org.codice.ddf.spatial.geocoding
- org.codice.ddf.spatial.kml.endpoint
- org.codice.ddf.spatial.kml.transformer
- org.codice.ddf.spatial.ogc.catalog.resource.impl
- org.codice.ddf.spatial.ogc.catalog
- org.codice.ddf.spatial.ogc.wfs.catalog.converter
- org.codice.ddf.spatial.ogc.wfs.catalog.mapper
- org.codice.ddf.spatial.ogc.wfs.v1_0_0.catalog.converter
- org.codice.ddf.spatial.ogc.wfs.v2_0_0.catalog.converter

B.8. Search UI Whitelist

The following packages have been exported by the Search UI application and are approved for use by third parties:

None.

Appendix C: All File Formats Supported

Using the various input transformers, DDF supports ingest of the following MIME types.

Table 192. Supported File Types

application		
activemessage	andrew-inset	applefile
applixware	atom+xml	atomcat+xml
atomicmail	atomsvc+xml	auth-policy+xml
batch-smtp	beep+xml	bizagi-modeler
cals-1840	cbor	ccxml+xml
cea-2018+xml	cellml+xml	cnrp+xml
commonground	conference-info+xml	cpl+xml
csta+xml	cstadata+xml	cu-seeme
cybercash	davmount+xml	dca-rft
dec-dx	dialog-info+xml	dicom
dif+xml	dita+xml	dita+xml
dita+xml	dita+xml	dita+xml
dita+xml	dns	dvcs
ecmascript	edi-consent	edi-x12
edifact	emma+xml	epp+xml

epub+zip	eshop	example
fastinfoset	fastsoap	fits
font-tdpfr	gzip	h224
http	hyperstudio	ibe-key-request+xml
ibe-pkg-reply+xml	ibe-pp-data	iges
illustrator	im-iscomposing+xml	index
index.cmd	index.obj	index.response
index/vnd	inf	iotp
ipp	isup	java-archive
java-serialized-object	java-vm	javascript
json	kate	kpml-request+xml
kpml-response+xml	lost+xml	mac-binhex40
mac-compactpro	macwriteii	marc
mathematica	mathml+xml	mbms-associated-procedure-description+xml
mbms-deregister+xml	mbms-envelope+xml	mbms-msk+xml
mbms-msk-response+xml	mbms-protection-description+xml	mbms-reception-report+xml
mbms-register+xml	mbms-register-response+xml	mbms-user-service-description+xml
mbox	media_control+xml	mediaservercontrol+xml
mikey	moss-keys	moss-signature
mosskey-data	mosskey-request	mp4
mpeg4-generic	mpeg4-iod	mpeg4-iod-xmt
msword	msword2	msword5
mxf	nasdata	news-checkgroups
news-groupinfo	news-transmission	nss
ocsp-request	ocsp-response	octet-stream
oda	oebps-package+xml	ogg
onenote	parityfec	patch-ops-error+xml
pdf	pgp-encrypted	pgp-keys
pgp-signature	pics-rules	pidf+xml
pidf-diff+xml	pkcs10	pkcs7-mime
pkcs7-signature	pkix-cert	pkix-crl
pkix-pkipath	pkixcmp	pls+xml
poc-settings+xml	postscript	prs.alvestrand.titrax-sheet
prs.cww	prs.nprend	prs.plucker
qsig	quicktime	rdf+xml

reginfo+xml	relax-ng-compact-syntax	remote-printing
resource-lists+xml	resource-lists-diff+xml	riscos
rlmi+xml	rls-services+xml	rsd+xml
rss+xml	rtf	rtx
samlassertion+xml	samlmetadata+xml	sbml+xml
scvp-cv-request	scvp-cv-response	scvp-vp-request
scvp-vp-response	sdp	sereal
sereal	sereal	sereal
set-payment	set-payment-initiation	set-registration
set-registration-initiation	sgml	sgml-open-catalog
shf+xml	sieve	simple-filter+xml
simple-message-summary	simplesymbolcontainer	slate
sldworks	smil+xml	soap+fastinfoset
soap+xml	sparql-query	sparql-results+xml
spirits-event+xml	srgs	srgs+xml
ssml+xml	timestamp-query	timestamp-reply
tve-trigger	ulpfec	vemmi
vividence.scriptfile	vnd.3gpp.bsf+xml	vnd.3gpp.pic-bw-large
vnd.3gpp.pic-bw-small	vnd.3gpp.pic-bw-var	vnd.3gpp.sms
vnd.3gpp2.bcmcsinfo+xml	vnd.3gpp2.sms	vnd.3gpp2.tcap
vnd.3m.post-it-notes	vnd.accpac.simply.aso	vnd.accpac.simply.imp
vnd.acucobol	vnd.acucorp	vnd.adobe.aftereffects.project
vnd.adobe.aftereffects.template	vnd.adobe.air-application-installer-package+zip	vnd.adobe.xdp+xml
vnd.adobe.xfdf	vnd.aether.imp	vnd.airzip.filessecure.azf
vnd.airzip.filessecure.azs	vnd.amazon.ebook	vnd.americandynamics.acc
vnd.amiga.ami	vnd.android.package-archive	vnd.anser-web-certificate-issue-initiation
vnd.anser-web-funds-transfer-initiation	vnd.antix.game-component	vnd.apple.installer+xml
vnd.apple.iwork	vnd.apple.keynote	vnd.apple.numbers
vnd.apple.pages	vnd.arastraswi	vnd.audiograph
vnd.autopackage	vnd.avistar+xml	vnd.blueice.multipass
vnd.bluetooth.ep.oob	vnd.bmi	vnd.businessobjects
vnd.cab-jscript	vnd.canon-cpd1	vnd.canon-lips
vnd.cendio.thinlinc.clientconf	vnd.chemdraw+xml	vnd.chipnuts.karaoke-mmd
vnd.cinderella	vnd.cirpack.isdn-ext	vnd.claymore
vnd.clonk.c4group	vnd.commerce-battelle	vnd.commonspace

vnd.contact.cmsg	vnd.cosmocaller	vnd.crick.clicker
vnd.crick.clicker.keyboard	vnd.crick.clicker.palette	vnd.crick.clicker.template
vnd.crick.clicker.wordbank	vnd.criticaltools.wbs+xml	vnd.ctc-posml
vnd.ctct.ws+xml	vnd.cups-pdf	vnd.cups-postscript
vnd.cups-ppd	vnd.cups-raster	vnd.cups-raw
vnd.curl.car	vnd.curl.pcurl	vnd.cybank
vnd.data-vision.rdz	vnd.denovo.fcselayout-link	vnd.dir-bi.plate-dl-nosuffix
vnd.dna	vnd.dolby.mlp	vnd.dolby.mobile.1
vnd.dolby.mobile.2	vnd.dpgraph	vnd.dreamfactory
vnd.dvb.esgcontainer	vnd.dvb.ipcdftnotifaccess	vnd.dvb.ipdcesgaccess
vnd.dvb.ipdcroaming	vnd.dvb.iptv.alfec-base	vnd.dvb.iptv.alfec-enhancement
vnd.dvb.notif-aggregate-root+xml	vnd.dvb.notif-container+xml	vnd.dvb.notif-generic+xml
vnd.dvb.notif-ia-msglist+xml	vnd.dvb.notif-ia-registration-request+xml	vnd.dvb.notif-ia-registration-response+xml
vnd.dvb.notif-init+xml	vnd.dxr	vnd.dynageo
vnd.ecdis-update	vnd.ecowin.chart	vnd.ecowin.filerequest
vnd.ecowin.fileupdate	vnd.ecowin.series	vnd.ecowin.seriesrequest
vnd.ecowin.seriesupdate	vnd.emclient.accessrequest+xml	vnd.enliven
vnd.epson.esf	vnd.epson.msf	vnd.epson.quickanime
vnd.epson.salt	vnd.epson.ssf	vnd.ericsson.quickcall
vnd.eszigno3+xml	vnd.etsi.aoc+xml	vnd.etsi.asic-e+zip
vnd.etsi.asic-s+zip	vnd.etsi.cug+xml	vnd.etsi.ipvcommand+xml
vnd.etsi.iptvdiscovery+xml	vnd.etsi.ipvprofile+xml	vnd.etsi.iptvsad-bc+xml
vnd.etsi.ptsad-cod+xml	vnd.etsi.ptsad-npvr+xml	vnd.etsi.ptsad-profile+xml
vnd.etsi.mcid+xml	vnd.etsi.sci+xml	vnd.etsi.simservs+xml
vnd.eudora.data	vnd.ezpix-album	vnd.ezpix-package
vnd.f-secure.mobile	vnd.fdf	vnd.fdsn.mseed
vnd.fdsn.seed	vnd.ffsns	vnd.fints
vnd.flographit	vnd.fluxtime.clip	vnd.font-fontforge-sfd
vnd.framemaker	vnd.frogans.fnc	vnd.frogans.ltf
vnd.fsc.weblaunch	vnd.fujitsu.oasys	vnd.fujitsu.oasys2
vnd.fujitsu.oasys3	vnd.fujitsu.oasysgp	vnd.fujitsu.oasysprs
vnd.fujixerox.art-ex	vnd.fujixerox.art4	vnd.fujixerox.ddd
vnd.fujixerox.docuworks	vnd.fujixerox.docuworks.binder	vnd.fujixerox.hbpl
vnd.fut-misnet	vnd.fuzzysheet	vnd.genomatix.tuxedo
vnd.geogebra.file	vnd.geogebra.tool	vnd.geometry-explorer

vnd.gmx	vnd.google-earth.kml+xml	vnd.google-earth.kmz
vnd.grafeq	vnd.gridmp	vnd.groove-account
vnd.groove-help	vnd.groove-identity-message	vnd.groove-injector
vnd.groove-tool-message	vnd.groove-tool-template	vnd.groove-vcard
vnd.handheld-entertainment+xml	vnd.hbci	vnd.hcl-bireports
vnd.hhe.lesson-player	vnd.hp-hpcl	vnd.hp-hpid
vnd.hp-hps	vnd.hp-jlyt	vnd.hp-pcl
vnd.hp-pclxl	vnd.httpphone	vnd.hydrostatix.sof-data
vnd.hzn-3d-crossword	vnd.ibm.afplinedata	vnd.ibm.electronic-media
vnd.ibm.minipay	vnd.ibm.modcap	vnd.ibm.rights-management
vnd.ibm.secure-container	vnd.iccprofile	vnd.igloader
vnd.immervision-ipv	vnd.immervision-ivu	vnd.informedcontrol.rms+xml
vnd.informix-visionary	vnd.intercon.formnet	vnd.intertrust.digibox
vnd.intertrust.nncp	vnd.intu.qbo	vnd.intu.qfx
vnd.iptc.g2.conceptitem+xml	vnd.iptc.g2.knowledgeitem+xml	vnd.iptc.g2.newsitem+xml
vnd.iptc.g2.packageitem+xml	vnd.ipunplugged.rcprofile	vnd.irepository.package+xml
vnd.is-xpr	vnd.jam	vnd.japannet-directory-service
vnd.japannet-jpnstore-wakeup	vnd.japannet-payment-wakeup	vnd.japannet-registration
vnd.japannet-registration-wakeup	vnd.japannet-setstore-wakeup	vnd.japannet-verification
vnd.japannet-verification-wakeup	vnd.jcp.javame.midlet-rms	vnd.jisp
vnd.joost.joda-archive	vnd.kahootz	vnd.kde.karbon
vnd.kde.kchart	vnd.kde.kformula	vnd.kde.kivio
vnd.kde.kontour	vnd.kde.kpresenter	vnd.kde.kspread
vnd.kde.kword	vnd.kenameaapp	vnd.kidspiration
vnd.kinar	vnd.koan	vnd.kodak-descriptor
vnd.liberty-request+xml	vnd.llamagraphics.life-balance.desktop	vnd.llamagraphics.life-balance.exchange+xml
vnd.lotus-1-2-3	vnd.lotus-approach	vnd.lotus-freelance
vnd.lotus-notes	vnd.lotus-organizer	vnd.lotus-screencam
vnd.lotus-wordpro	vnd.macports.portpkg	vnd.marlin drm.actiontoken+xml
vnd.marlin drm.conftoken+xml	vnd.marlin drm.license+xml	vnd.marlin drm.mDCF
vnd.mcd	vnd.medcalldata	vnd.mediastation.cdkey
vnd.meridian-slingshot	vnd.mfer	vnd.mfmp
vnd.micrografx.flo	vnd.micrografx.igx	vnd.mif
vnd.mindjet.mindmanager	vnd.minisoft-hp3000-save	vnd.mitsubishi.misty-guard.trustweb

vnd.mobius.daf	vnd.mobius.dis	vnd.mobius.mbk
vnd.mobius.mqy	vnd.mobius.msl	vnd.mobius.plc
vnd.mobius.txf	vnd.mophun.application	vnd.mophun.certificate
vnd.motorola.flexsuite	vnd.motorola.flexsuite.adsi	vnd.motorola.flexsuite.fis
vnd.motorola.flexsuite.gotap	vnd.motorola.flexsuite.kmr	vnd.motorola.flexsuite.ttc
vnd.motorola.flexsuite.wem	vnd.motorola.iprm	vnd.mozilla.xul+xml
vnd.ms-artgalry	vnd.ms-asf	vnd.ms-cab-compressed
vnd.ms-excel	vnd.ms-excel.addin.macroenabled.12	vnd.ms-excel.sheet.2
vnd.ms-excel.sheet.3	vnd.ms-excel.sheet.4	vnd.ms-excel.sheet.binary.macroenabled.12
vnd.ms-excel.sheet.macroenabled.12	vnd.ms-excel.template.macroenabled.12	vnd.ms-excel.workspace.3
vnd.ms-excel.workspace.4	vnd.ms-fontobject	vnd.ms-htmlhelp
vnd.ms-ims	vnd.ms-lrm	vnd.ms-outlook
vnd.ms-outlook-pst	vnd.ms-pki.seccat	vnd.ms-pki.stl
vnd.ms-playready.initiator+xml	vnd.ms-powerpoint	vnd.ms-powerpoint.addin.macroenabled.12
vnd.ms-powerpoint.presentation.macroenabled.12	vnd.ms-powerpoint.slide.macroenabled.12	vnd.ms-powerpoint.slideshow.macroenabled.12
vnd.ms-powerpoint.template.macroenabled.12	vnd.ms-project	vnd.ms-tnef
vnd.ms-visio.drawing	vnd.ms-visio.drawing.macroenabled.12	vnd.ms-visio.stencil
vnd.ms-visio.stencil.macroenabled.12	vnd.ms-visio.template	vnd.ms-visio.template.macroenabled.12
vnd.ms-wmdrm.lic-chlg-req	vnd.ms-wmdrm.lic-resp	vnd.ms-wmdrm.meter-chlg-req
vnd.ms-wmdrm.meter-resp	vnd.ms-word.document.macroenabled.12	vnd.ms-word.template.macroenabled.12
vnd.ms-works	vnd.ms-wpl	vnd.ms-xpsdocument
vnd.mseq	vnd.ms-sign	vnd.multiad.creator
vnd.multiad.creator.cif	vnd.music-niff	vnd.musician
vnd.muvee.style	vnd.ncd.control	vnd.ncd.reference
vnd.nervana	vnd.netfp	vnd.neurolanguage.nlu
vnd.noblenet-directory	vnd.noblenet-sealer	vnd.noblenet-web
vnd.nokia.catalogs	vnd.nokia.comml+wbxml	vnd.nokia.comml+xml
vnd.nokia.ipv.config+xml	vnd.nokia.isds-radio-presets	vnd.nokia.landmark+wbxml
vnd.nokia.landmark+xml	vnd.nokia.landmarkcollection+xml	vnd.nokia.n-gage.ac+xml

vnd.nokia.n-gage.data	vnd.nokia.n-gage.symbian.install	vnd.nokia.ncd
vnd.nokia.pcd+wbxml	vnd.nokia.pcd+xml	vnd.nokia.radio-preset
vnd.nokia.radio-presets	vnd.novadigm.edm	vnd.novadigm.edx
vnd.novadigm.ext	vnd.oasis.opendocument.chart	vnd.oasis.opendocument.chart-template
vnd.oasis.opendocument.database	vnd.oasis.opendocument.formula	vnd.oasis.opendocument.formula-template
vnd.oasis.opendocument.graphics	vnd.oasis.opendocument.graphics-template	vnd.oasis.opendocument.image
vnd.oasis.opendocument.image-template	vnd.oasis.opendocument.presentation	vnd.oasis.opendocument.presentation-template
vnd.oasis.opendocument.spreadsheet	vnd.oasis.opendocument.spreadsheet-template	vnd.oasis.opendocument.text
vnd.oasis.opendocument.text-master	vnd.oasis.opendocument.text-template	vnd.oasis.opendocument.text-web
vnd.obn	vnd.olpc-sugar	vnd.oma-scws-config
vnd.oma-scws-http-request	vnd.oma-scws-http-response	vnd.oma.bcast.associated-procedure-parameter+xml
vnd.oma.bcast.drm-trigger+xml	vnd.oma.bcast.imd+xml	vnd.oma.bcast.ltkm
vnd.oma.bcast.notification+xml	vnd.oma.bcast.provisioningtrigger	vnd.oma.bcast.sgboot
vnd.oma.bcast.sgdd+xml	vnd.oma.bcast.sgdu	vnd.oma.bcast.simple-symbol-container
vnd.oma.bcast.smartcard-trigger+xml	vnd.oma.bcast.sprov+xml	vnd.oma.bcast.stkm
vnd.oma.dcd	vnd.oma.dcde	vnd.oma.dd2+xml
vnd.oma.drm.risd+xml	vnd.oma.group-usage-list+xml	vnd.oma.poc.detailed-progress-report+xml
vnd.oma.poc.final-report+xml	vnd.oma.poc.groups+xml	vnd.oma.poc.invocation-descriptor+xml
vnd.oma.poc.optimized-progress-report+xml	vnd.oma.xcap-directory+xml	vnd.omads-email+xml
vnd.omads-file+xml	vnd.omads-folder+xml	vnd.omaloc-supl-init
vnd.openofficeorg.extension	vnd.openxmlformats-officedocument.presentationml.presentation	vnd.openxmlformats-officedocument.presentationml.slide
vnd.openxmlformats-officedocument.presentationml.slideshow	vnd.openxmlformats-officedocument.presentationml.template	vnd.openxmlformats-officedocument.spreadsheetml.sheet
vnd.openxmlformats-officedocument.spreadsheetml.template	vnd.openxmlformats-officedocument.wordprocessingml.document	vnd.openxmlformats-officedocument.wordprocessingml.template
vnd.osa.netdeploy	vnd.osgi.bundle	vnd.osgi.dp
vnd.otps.ct-kip+xml	vnd.palm	vnd.paos.xml

vnd.pg.format	vnd.pg.osasli	vnd.piaccess.application-licence
vnd.picsel	vnd.poc.group-advertisement+xml	vnd.pocketlearn
vnd.powerbuilder6	vnd.powerbuilder6-s	vnd.powerbuilder7
vnd.powerbuilder7-s	vnd.powerbuilder75	vnd.powerbuilder75-s
vnd.preminet	vnd.previewsystems.box	vnd.proteus.magazine
vnd.publishare-delta-tree	vnd.pvi.ptid1	vnd.pwg-multiplexed
vnd.pwg-xhtml-print+xml	vnd.qualcomm.brew-app-res	vnd.quark.quarkxpress
vnd.rapid	vnd.recordare.musicxml	vnd.recordare.musicxml+xml
vnd.renlearn.rlprint	vnd.rim.cod	vnd.rn-realmedia
vnd.route66.link66+xml	vnd.ruckus.download	vnd.s3sms
vnd.sbm.cid	vnd.sbm.mid2	vnd.scribus
vndsealed.3df	vndsealed.csf	vndsealed.doc
vndsealed.eml	vndsealed.mht	vndsealed.net
vndsealed.ppt	vndsealed.tiff	vndsealed.xls
vndsealedmedia.softseal.html	vndsealedmedia.softseal.pdf	vndseemail
vndsema	vndsemld	vndsemf
vndshana.informed.formdata	vndshana.informed.formtemplate	vndshana.informed.interchange
vndshana.informed.package	vndsimtech-mindmapper	vndsmaf
vndsmart.teacher	vndsoftware602.filler.form+xml	vndsoftware602.filler.form+xml-zip
vndsolent.sdkm+xml	vndspotfire.dxp	vndspotfire.sfs
vndsss-cod	vndsss-dtf	vndsss-ntf
vndstardivision.calc	vndstardivision.draw	vndstardivision.impress
vndstardivision.math	vndstardivision.writer	vndstardivision.writer-global
vndstreet-stream	vndsun.wadl+xml	vndsun.xml.calc
vndsun.xml.calc.template	vndsun.xml.draw	vndsun.xml.draw.template
vndsun.xml.impress	vndsun.xml.impress.template	vndsun.xml.math
vndsun.xml.writer	vndsun.xml.writer.global	vndsun.xml.writer.template
vndsus-calendar	vndsvd	vndswiftview-ics
vnd symbian.install	vndsyncml+xml	vndsyncml.dm+wbxml
vndsyncml.dm+xml	vndsyncml.dm.notification	vndsyncml.ds.notification
vndtao.intent-module-archive	vndtcpdump.pcap	vndtmobile-livetv
vndtrid.tpt	vndtriscape.mxs	vndtrueapp
vndtruedoc	vndufdl	vnduiq.theme
vndumajin	vndunity	vnduoml+xml

vnd.uplanet.alert	vnd.uplanet.alert-wbxml	vnd.uplanet.bearer-choice
vnd.uplanet.bearer-choice-wbxml	vnd.uplanet.cacheop	vnd.uplanet.cacheop-wbxml
vnd.uplanet.channel	vnd.uplanet.channel-wbxml	vnd.uplanet.list
vnd.uplanet.list-wbxml	vnd.uplanet.listcmd	vnd.uplanet.listcmd-wbxml
vnd.uplanet.signal	vnd.vcx	vnd.vd-study
vnd.vectorworks	vnd.vidsoft.vidconference	vnd.visio
vnd.visionary	vnd.vividence.scriptfile	vnd.vsf
vnd.wap.sic	vnd.wap.slc	vnd.wap.wbxml
vnd.wap.wmlc	vnd.wap.wmlscriptc	vnd.webturbo
vnd.wfa.wsc	vnd.wmc	vnd.wmf.bootstrap
vnd.wordperfect	vnd.wqd	vnd.wrq-hp3000-labelled
vnd.wt.stf	vnd.wv.csp+wbxml	vnd.wv.csp+xml
vnd.wv.ssp+xml	vnd.xara	vnd.xfdl
vnd.xfdl.webform	vnd.xmi+xml	vnd.xmpie.pkg
vnd.xmpie.dpkg	vnd.xmpie.plan	vnd.xmpie.pkg
vnd.xmpie.xlim	vnd.yamaha.hv-dic	vnd.yamaha.hv-script
vnd.yamaha.hv-voice	vnd.yamaha.openscoreformat	vnd.yamaha.openscoreformat.osf+xml
vnd.yamaha.smaf-audio	vnd.yamaha.smaf-phrase	vnd.yellowriver-custom-menu
vnd.zul	vnd.zzazz.deck+xml	voicexml+xml
watcherinfo+xml	whoispp-query	whoispp-response
winhlp	wita	wordperfect5.1
wsdl+xml	wspolicy+xml	x-123
x-7z-compressed	x-abiword	x-ace-compressed
x-adobe-indesign	x-adobe-indesign-interchange	x-apple-diskimage
x-appleworks	x-archive	x-arj
x-authorware-bin	x-authorware-map	x-authorware-seg
x-axcrypt	x-bcpio	x-berkeley-db
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-bittorrent	x-bplist	x-bzip
x-bzip2	x-cdlink	x-chat
x-chess-pgn	x-chrome-package	x-compress
x-coredump	x-corelpresentations	x-cpio

x-csh	x-debian-package	x-dex
x-director	x-doom	x-dosexec
x-dtbncx+xml	x-dtbook+xml	x-dtbresource+xml
x-dvi	x-elc	x-elf
x-emf	x-erdas-hfa	x-executable
x-fictionbook+xml	x-filemaker	x-font-adobe-metric
x-font-bdf	x-font-dos	x-font-framemaker
x-font-ghostscript	x-font-libgrx	x-font-linux-psf
x-font-otf	x-font-pcf	x-font-printer-metric
x-font-snf	x-font-speedo	x-font-sunos-news
x-font-ttf	x-font-type1	x-font-vfont
x-foxmail	x-futuresplash	x-gnucash
x-gnumeric	x-grib	x-gtar
x-hdf	x-hwp	x-hwp-v5
x-ibooks+zip	x-isatab	x-isatab-assay
x-isatab-investigation	x-iso9660-image	x-itunes-ipa
x-java-jnilib	x-java-jnlp-file	x-java-pack200
x-kdelnk	x-killustrator	x-latex
x-lha	x-lharc	x-matlab-data
x-matroska	x-mobipocket-ebook	x-ms-application
x-ms-installer	x-ms-wmd	x-ms-wmz
x-ms-xbap	x-msaccess	x-msbinder
x-mscardfile	x-msclip	x-msdownload
x-msdownload	x-msdownload	x-msdownload
x-msdownload	x-msdownload	x-msdownload
x-msmediaview	x-msmetafile	x-msmoney
x-mspublisher	x-msschedule	x-msterminal
x-mswrite	x-mysql-db	x-mysql-misam-compressed-index
x-mysql-misam-data	x-mysql-misam-index	x-mysql-table-definition
x-netcdf	x-object	x-pkcs12
x-pkcs7-certificates	x-pkcs7-certreqresp	x-project
x-prt	x-quattro-pro	x-rar-compressed
x-roxio-toast	x-rpm	x-sas
x-sas-access	x-sas-audit	x-sas-backup
x-sas-catalog	x-sas-data	x-sas-data-index
x-sas-dmdb	x-sas-fdb	x-sas-itemstor
x-sas-mddb	x-sas-program-data	x-sas-putility

x-sas-transport	x-sas-utility	x-sas-view
x-sc	x-sfdu	x-sh
x-shapefile	x-shar	x-sharedlib
x-shockwave-flash	x-silverlight-app	x-snappy-framed
x-sqlite3	x-staroffice-template	x-stuffit
x-stuffitx	x-sv4cpio	x-sv4crc
x-tar	x-tex	x-tex-tfm
x-texinfo	x-tika-iworks-protected	x-tika-java-enterprise-archive
x-tika-java-web-archive	x-tika-msoffice	x-tika-msoffice-embedded
x-tika-msoffice-embedded	x-tika-msoffice-embedded	x-tika-msworks-spreadsheet
x-tika-old-excel	x-tika-ooxml	x-tika-ooxml-protected
x-tika-staroffice	x-tika-unix-dump	x-tika-visio-ooxml
x-uc2-compressed	x-ustar	x-vhd
x-vmdk	x-wais-source	x-webarchive
x-x509-ca-cert	x-xfig	x-xmind
x-xpinstall	x-xz	x-zoo
x400-bp	xcap-att+xml	xcap-caps+xml
xcap-el+xml	xcap-error+xml	xcap-ns+xml
xcon-conference-info+xml	xcon-conference-info-diff+xml	xenc+xml
xhtml+xml	xhtml-voice+xml	xml
xml-dtd	xml-external-parsed-entity	xmpp+xml
xop+xml	xquery	xslfo+xml
xslt+xml	xspf+xml	xv+xml
zip	zlib	

audio

32kadpcm	3gpp	3gpp2
ac3	adpcm	amr
amr-wb	amr-wb+	asc
basic	bv16	bv32
clearmode	cn	dat12
dls	dsr-es201108	dsr-es202050
dsr-es202211	dsr-es202212	dvi4
eac3	evrc	evrc-qcp
evrc0	evrc1	evrcb
evrcb0	evrcb1	evrcwb
evrcwb0	evrcwb1	example

g719	g722	g7221
g723	g726-16	g726-24
g726-32	g726-40	g728
g729	g7291	g729d
g729e	gsm	gsm-efr
ilbc	l16	l20
l24	l8	lpc
midi	mobile-xmf	mp4
mp4a-latm	mpa	mpa-robust
mpeg	mpeg4-generic	ogg
opus	parityfec	pcma
pcma-wb	pcmu	pcmu-wb
prc.sid	qcelp	red
rtp-enc-aescm128	rtp-midi	rtx
smv	smv-qcp	smv0
sp-midi	speex	t140c
t38	telephone-event	tone
ulpfec	vdvi	vmr-wb
vnd.3gpp.iufp	vnd.4sb	vnd.adobe.soundbooth
vnd.audiokoz	vnd.celp	vnd.cisco.nse
vnd.cmles.radio-events	vnd.cns.anp1	vnd.cns.inf1
vnd.digital-winds	vnd.dlna.adts	vnd.dolby.heaac.1
vnd.dolby.heaac.2	vnd.dolby.mlp	vnd.dolby.mps
vnd.dolby.pl2	vnd.dolby.pl2x	vnd.dolby.pl2z
vnd.dts	vnd.dts.hd	vnd.everad.plj
vnd.hns.audio	vnd.lucent.voice	vnd.ms-playready.media.pya
vnd.nokia.mobile-xmf	vnd.nortel.vbk	vnd.nuera.ecelp4800
vnd.nuera.ecelp7470	vnd.nuera.ecelp9600	vnd.octel.sbc
vnd.qcelp	vnd.rhetorex.32kadpcm	vndsealedmedia.softseal.mpeg
vnd.vmx.csvd	vorbis	vorbis-config
x-aac	x-adbcm	x-aiff
x-dec-adbcm	x-dec-basic	x-flac
x-matroska	x-mod	x-mpegurl
x-ms-wax	x-ms-wma	x-oggflac
x-oggpcm	x-pn-realaudio	x-pn-realaudio-plugin
x-wav		

chemical		
x-cdx	x-cif	x-cmdf
x-cml	x-csml	x-pdb
x-xyz		
image		
bmp	cgm	example
fits	g3fax	gif
icns	ief	jp2
jpeg	jpm	jpx
naplps	nitf	png
prs.btif	prs.pti	svg+xml
t38	tiff	tiff-fx
vnd.adobe.photoshop	vnd.adobe.premiere	vnd.cns.inf2
vnd.djvu	vnd.dwg	vnd.dxb
vnd.dxf	vnd.dxf	vnd.dxf
vnd.fastbidsheet	vnd.fpx	vnd.fst
vnd.fujixerox.edmics-mmr	vnd.fujixerox.edmics-rlc	vnd.globalgraphics.pgb
vnd.microsoft.icon	vnd.mix	vnd.ms-modi
vnd.net-fpx	vnd.radiance	vndsealed.png
vndsealedmedia.softseal.gif	vndsealedmedia.softseal.jpg	vnd.svf
vnd.wap.wbmp	vnd.xiff	webp
x-bpg	x-cmu-raster	x-cmx
x-freehand	x-jp2-codestream	x-jp2-container
x-ms-bmp	x-niff	x-pcx
x-pict	x-portable-anymap	x-portable-bitmap
x-portable-graymap	x-portable-pixmap	x-raw-adobe
x-raw-canon	x-raw-casio	x-raw-epson
x-raw-fuji	x-raw-hasselblad	x-raw-imacon
x-raw-kodak	x-raw-leaf	x-raw-logitech
x-raw-mamiya	x-raw-minolta	x-raw-nikon
x-raw-olympus	x-raw-panasonic	x-raw-pentax
x-raw-phaseone	x-raw-rawzor	x-raw-red
x-raw-sigma	x-raw-sony	x-rgb
x-xbitmap	x-xcf	x-xpixmap
x-xwindowdump		
message		

cpim	delivery-status	disposition-notification
example	external-body	global
global-delivery-status	global-disposition-notification	global-headers
http	imdn+xml	news
partial	rfc822	s-http
sip	sipfrag	tracking-status
vnd.si.simp	x-emlx	

model

example	iges	mesh
vnd.dwf	vnd.dwf	vnd.dwf
vnd.dwf	vnd.dwfx+xps	vnd.flatland.3dml
vnd.gdl	vnd.gs-gdl	vnd.gs.gdl
vnd.gtw	vnd.moml+xml	vnd.mts
vnd.parasolid.transmit.binary	vnd.parasolid.transmit.text	vnd.vtu
vrml		

multipart

alternative	appledouble	byteranges
digest	encrypted	example
form-data	header-set	mixed
parallel	related	report
signed	voice-message	

text

asp	aspdotnet	calendar
css	csv	directory
dns	ecmascript	enriched
example	html	iso19139+xml
parityfec	plain	prs.fallenstein.rst
prs.lines.tag	red	rfc822-headers
richtext	rtp-enc-aescm128	rtx
sgml	t140	tab-separated-values
troff	ulpfec	uri-list
vnd.abc	vnd.curl	vnd(curl).dcurl
vnd.curl.mcurl	vnd.curl.scurl	vnd(dmclientscript)
vnd.esmertec.theme-descriptor	vnd.fly	vnd.fmi.flexstor
vnd.graphviz	vnd.in3d.3dml	vnd.in3d.spot
vnd.iptc.anpa	vnd.iptc.newsml	vnd.iptc.nitf

vnd.latex-z	vnd.motorola.reflex	vnd.ms-mediapackage
vnd.net2phone.commcenter.command	vnd.si.uricatalogue	vnd.sun.j2me.app-descriptor
vnd.trolltech.linguist	vnd.wap.si	vnd.wap.sl
vnd.wap.wml	vnd.wap.wmlscript	vtt
x-actionscript	x-ada	x-applescript
x-asciidoc	x-aspectj	x-assembly
x-awk	x-basic	x-c++hdr
x-c++src	x-cgi	x-chdr
x-clojure	x-cobol	x-coffeescript
x-coldfusion	x-common-lisp	x-csharp
x-csrc	x-d	x-diff
x-eiffel	x-emacs-lisp	x-erlang
x-expect	x-forth	x-fortran
x-go	x-groovy	x-haml
x-haskell	x-haxe	x-idl
x-ini	x-java-properties	x-java-source
x-jsp	x-less	x-lex
x-log	x-lua	x-matlab
x-ml	x-modula	x-objcsrc
x-ocaml	x-pascal	x-perl
x-php	x-prolog	x-python
x-rexx	x-rsrc	x-rst
x-ruby	x-scala	x-scheme
x-sed	x-setext	x-sql
x-stsrc	x-tcl	x-tika-text-based-message
x-uuencode	x-vbasic	x-vbdotnet
x-vbscript	x-vcalendar	x-vcard
x-verilog	x-vhdl	x-web-markdown
x-yacc	x-yaml	

video

3gpp	3gpp-tt	3gpp2
bmpg	bt656	celb
daala	dv	example
h261	h263	h263-1998
h263-2000	h264	jpeg
jpeg2000	mj2	mp1s

mp2p	mp2t	mp4
mp4v-es	mpeg	mpeg4-generic
mpv	nv	ogg
parityfec	pointer	quicktime
raw	rtp-enc-aescm128	rtx
smpte292m	theora	ulpfec
vc1	vnd.cctv	vnd.dlna.mpeg-tts
vnd.fvt	vnd.hns.video	vnd.ffmpeg.1dparityfec-1010
vnd.ffmpeg.1dparityfec-2005	vnd.ffmpeg.2dparityfec-1010	vnd.ffmpeg.2dparityfec-2005
vnd.ffmpeg.ttsavc	vnd.ffmpeg.ttsmpeg2	vnd.motorola.video
vnd.motorola.videop	vnd.mpegurl	vnd.ms-playready.media.pyv
vnd.nokia.interleaved-multimedia	vnd.nokia.videovoip	vnd.objectvideo
vndsealed.mpeg1	vndsealed.mpeg4	vndsealed.swf
vndsealedmedia.softseal.mov	vnd.vivo	webm
x-dirac	x-f4v	x-flc
x-fli	x-flv	x-jng
x-m4v	x-matroska	x-mng
x-ms-asf	x-ms-wm	x-ms-wmv
x-ms-wmx	x-ms-wvx	x-msvideo
x-oggrgb	x-ogguvs	x-oggyuv
x-ogm	x-sgi-movie	
x-conference		
x-cooltalk		

Appendix D: DDF Dependency List

This list of DDF dependencies is automatically generated:

DDF 2.10.3 Dependency List

- antlr:antlr:jar:2.7.7
- c3p0:c3p0:jar:0.9.1.1
- ca.juliusdavies:not-yet-commons-ssl:jar:0.3.11
- cglib:cglib-nodep:jar:3.2.0
- ch.qos.logback:logback-access:jar:1.1.7
- ch.qos.logback:logback-classic:jar:1.1.7
- ch.qos.logback:logback-core:jar:1.1.2
- ch.qos.logback:logback-core:jar:1.1.7

- com.codahale.metrics:metrics-core:jar:3.0.1
- com.connexta.arbitro:arbitro-core:jar:1.0.0
- com.github.drapostolos:type-parser:jar:0.5.0
- com.github.jai-imageio:jai-imageio-core:jar:1.3.1
- com.github.jai-imageio:jai-imageio-jpeg2000:jar:1.3.0
- com.github.jknack:handlebars-jackson2:jar:1.0.0
- com.github.jknack:handlebars:jar:1.1.2
- com.github.jknack:handlebars:jar:2.0.0
- com.github.rvesse:airline:jar:2.1.0
- com.google.code.findbugs:findbugs:jar:3.0.1
- com.google.code.gson:gson:jar:2.4
- com.google.guava:guava:jar:18.0
- com.google.http-client:google-http-client:jar:1.22.0
- com.googlecode.json-simple:json-simple:jar:1.1.1
- com.hazelcast:hazelcast:jar:3.2.1
- com.lucidworks:banana:war:1.5.1
- com.rometools:rome-utils:jar:1.5.0
- com.rometools:rome:jar:1.5.0
- com.sparkjava:spark-core:jar:2.5
- com.sun.xml.bind:jaxb-core:jar:2.2.11
- com.sun.xml.bind:jaxb-impl:jar:2.2.11
- com.thoughtworks.xstream:xstream:jar:1.4.3
- com.thoughtworks.xstream:xstream:jar:1.4.4
- com.vividsolutions:jts:jar:1.12
- com.vividsolutions:jts:jar:1.13
- com.xebialabs.restito:restito:jar:0.8.2
- commons-beanutils:commons-beanutils:jar:1.9.2
- commons-codec:commons-codec:jar:1.10
- commons-collections:commons-collections:jar:3.2.2
- commons-configuration:commons-configuration:jar:1.10
- commons-fileupload:commons-fileupload:jar:1.3.2
- commons-io:commons-io:jar:2.1
- commons-io:commons-io:jar:2.4
- commons-lang:commons-lang:jar:2.6
- commons-logging:commons-logging:jar:1.2

- commons-net:commons-net:jar:3.5
- commons-pool:commons-pool:jar:1.6
- commons-validator:commons-validator:jar:1.3.1
- de.micromata.jak:JavaAPIforKml:jar:2.2.0
- io.dropwizard.metrics:metrics-core:jar:3.1.2
- io.fastjson:boon:jar:0.30
- io.fastjson:boon:jar:0.33
- javax.inject:javax.inject:jar:1
- javax.servlet:javax.servlet-api:jar:3.1.0
- javax.servlet:servlet-api:jar:2.5
- javax.validation:validation-api:jar:1.1.0.Final
- javax.ws.rs:javax.ws.rs-api:jar:2.0
- javax.ws.rs:javax.ws.rs-api:jar:2.0-m10
- javax.ws.rs:javax.ws.rs-api:jar:2.0.1
- joda-time:joda-time:jar:1.6.2
- joda-time:joda-time:jar:2.2
- joda-time:joda-time:jar:2.5
- junit:junit:jar:4.12
- log4j:log4j:jar:1.2.17
- net.coobird:thumbnailator:jar:0.4.8
- net.jodah:failsafe:jar:0.9.3
- net.jodah:failsafe:jar:0.9.5
- net.lingala.zip4j:zip4j:jar:1.3.2
- net.minidev:json-smart:jar:1.1.1
- net.sf.saxon:Saxon-HE:jar:9.6.0-4
- org.antlr:antlr4-runtime:jar:4.1
- org.antlr:antlr4-runtime:jar:4.3
- org.apache.abdera:abdera-extensions-geo:jar:1.1.3
- org.apache.abdera:abdera-extensions-opensearch:jar:1.1.3
- org.apache.aries.jmx:org.apache.aries.jmx.api:jar:1.1.5
- org.apache.aries.jmx:org.apache.aries.jmx.core:jar:1.1.6
- org.apache.camel:camel-blueprint:jar:2.18.0
- org.apache.camel:camel-context:jar:2.18.0
- org.apache.camel:camel-core-osgi:jar:2.18.0
- org.apache.camel:camel-core:jar:2.18.0

- org.apache.camel:camel-http-common:jar:2.18.0
- org.apache.camel:camel-http4:jar:2.18.0
- org.apache.camel:camel-http:jar:2.18.0
- org.apache.camel:camel-quartz:jar:2.18.0
- org.apache.camel:camel-saxon:jar:2.18.0
- org.apache.camel:camel-servlet:jar:2.18.0
- org.apache.commons:commons-collections4:jar:4.1
- org.apache.commons:commons-exec:jar:1.3
- org.apache.commons:commons-lang3:jar:3.0
- org.apache.commons:commons-lang3:jar:3.1
- org.apache.commons:commons-lang3:jar:3.3.2
- org.apache.commons:commons-lang3:jar:3.4
- org.apache.commons:commons-math:jar:2.2
- org.apache.cxf.services.sts:cxf-services-sts-core:jar:3.1.7
- org.apache.cxf:cxf-core:jar:3.1.7
- org.apache.cxf:cxf-rt-bindings-soap:jar:3.0.4
- org.apache.cxf:cxf-rt-databinding-jaxb:jar:3.0.4
- org.apache.cxf:cxf-rt-frontend-jaxrs:jar:3.1.7
- org.apache.cxf:cxf-rt-frontend-jaxws:jar:3.0.4
- org.apache.cxf:cxf-rt-frontend-jaxws:jar:3.1.7
- org.apache.cxf:cxf-rt-rs-client:jar:3.1.7
- org.apache.cxf:cxf-rt-rs-security-sso-saml:jar:3.1.7
- org.apache.cxf:cxf-rt-rs-security-xml:jar:3.0.4
- org.apache.cxf:cxf-rt-rs-security-xml:jar:3.1.7
- org.apache.cxf:cxf-rt-transports-http:jar:3.1.7
- org.apache.cxf:cxf-rt-ws-policy:jar:3.1.7
- org.apache.cxf:cxf-rt-ws-security:jar:3.1.7
- org.apache.felix:org.apache.felix.configadmin:jar:1.8.8
- org.apache.felix:org.apache.felix.fileinstall:jar:3.5.4
- org.apache.felix:org.apache.felix.framework:jar:5.4.0
- org.apache.felix:org.apache.felix.gogo.command:jar:0.16.0
- org.apache.felix:org.apache.felix.utils:jar:1.8.2
- org.apache.ftpserver:ftplet-api:jar:1.0.6
- org.apache.ftpserver:ftpserver-core:jar:1.0.6
- org.apache.geronimo.specs:geronimo-servlet_3.0_spec:jar:1.0

- org.apache.httpcomponents:httpclient:jar:4.5.2
- org.apache.httpcomponents:httpcore:jar:4.4.5
- org.apache.httpcomponents:httpmime:jar:4.5.2
- org.apache.karaf.bundle:org.apache.karaf.bundle.core:jar:4.0.7
- org.apache.karaf.features:enterprise:xml:features:4.0.7
- org.apache.karaf.features:org.apache.karaf.features.core:jar:4.0.7
- org.apache.karaf.features:standard:xml:features:4.0.7
- org.apache.karaf.jaas:org.apache.karaf.jaas.boot:jar:4.0.7
- org.apache.karaf.jaas:org.apache.karaf.jaas.config:jar:4.0.7
- org.apache.karaf.jaas:org.apache.karaf.jaas.modules:jar:4.0.7
- org.apache.karaf.shell:org.apache.karaf.shell.console:jar:4.0.7
- org.apache.karaf.shell:org.apache.karaf.shell.core:jar:4.0.7
- org.apache.karaf:apache-karaf:tar.gz:4.0.7
- org.apache.karaf:apache-karaf:zip:4.0.7
- org.apache.karaf:org.apache.karaf.util:jar:4.0.7
- org.apache.logging.log4j:log4j-api:jar:2.4.1
- org.apache.lucene:lucene-analyzers-common:jar:6.0.0
- org.apache.lucene:lucene-core:jar:3.0.2
- org.apache.lucene:lucene-core:jar:6.0.0
- org.apache.lucene:lucene-queries:jar:6.0.0
- org.apache.lucene:lucene-queryparser:jar:6.0.0
- org.apache.lucene:lucene-sandbox:jar:6.0.0
- org.apache.lucene:lucene-spatial-extras:jar:6.0.0
- org.apache.lucene:lucene-spatial3d:jar:6.0.0
- org.apache.lucene:lucene-spatial:jar:6.0.0
- org.apache.maven.shared:maven-invoker:jar:2.2
- org.apache.mina:mina-core:jar:2.0.6
- org.apache.pdfbox:fontbox:jar:2.0.2
- org.apache.pdfbox:pdfbox-tools:jar:2.0.2
- org.apache.pdfbox:pdfbox:jar:2.0.2
- org.apache.poi:poi-contrib:jar:3.7-beta3
- org.apache.poi:poi-ooxml:jar:3.9
- org.apache.poi:poi-scratchpad:jar:3.9
- org.apache.poi:poi:jar:3.9
- org.apache.servicemix.bundles:org.apache.servicemix.bundles.poi:jar:3.13_1

- org.apache.servicemix.specs:org.apache.servicemix.specs.jsr339-api-2.0;jar:2.5.0
- org.apache.shiro:shiro-core;jar:1.2.4
- org.apache.solr:solr-core;jar:6.0.0
- org.apache.solr:solr-solrj;jar:6.0.0
- org.apache.tika:tika-bundle;jar:1.13
- org.apache.tika:tika-core;jar:1.13
- org.apache.tika:tika-parsers;jar:1.13
- org.apache.ws.commons.axiom:axiom-api;jar:1.2.14
- org.apache.wss4j:wss4j-bindings;jar:2.1.4
- org.apache.wss4j:wss4j-policy;jar:2.1.4
- org.apache.wss4j:wss4j-ws-security-common;jar:2.1.4
- org.apache.wss4j:wss4j-ws-security-dom;jar:2.1.4
- org.apache.wss4j:wss4j-ws-security-policy-stax;jar:2.1.4
- org.apache.wss4j:wss4j-ws-security-stax;jar:2.1.4
- org.asciidoctor:asciidoctorj;jar:1.5.4
- org.bouncycastle:bcmail-jdk15on;jar:1.54
- org.bouncycastle:bcpkix-jdk15on;jar:1.54
- org.bouncycastle:bcprov-jdk15on;jar:1.54
- org.codehaus.groovy:groovy-all;jar:2.4.7
- org.codehaus.jackson:jackson-core-asl;jar:1.9.13
- org.codehaus.jackson:jackson-mapper-asl;jar:1.9.13
- org.codehaus.woodstox:woodstox-core-asl;jar:4.4.1
- org.codice.geowebcache:geowebcache-server-standalone:war:0.6
- org.codice.geowebcache:geowebcache-server-standalone:xml:geowebcache:0.6
- org.codice.httpproxy:proxy-camel-route;jar:2.10.0-SNAPSHOT
- org.codice.httpproxy:proxy-camel-servlet;jar:2.10.0-SNAPSHOT
- org.codice.opendj.embedded:opendj-embedded-app:xml:features:1.3.3
- org.codice.org.forgerock.commons:forgerock-util;jar:2.0.0.ALPHA1
- org.codice.org.forgerock.opendj:opendj-core;jar:3.0.0.ALPHA2
- org.codice.org.forgerock.opendj:opendj-grizzly;jar:3.0.0.ALPHA2
- org.codice.thirdparty:cas-client-core;jar:3.1.10_1
- org.codice.thirdparty:commons-httpclient;jar:3.1.0_1
- org.codice.thirdparty:ffmpeg:zip:bin:3.1.1_1
- org.codice.thirdparty:geotools-suite;jar:8.4_2
- org.codice.thirdparty:gt-opengis;jar:8.4_1

- org.codice.thirdparty:jts:jar:1.12_1
- org.codice.thirdparty:lucene-core:jar:3.0.2_1
- org.codice.thirdparty:ogc-filter-v_1_1_0-schema:jar:1.1.0_2
- org.codice.thirdparty:ogc-filter-v_1_1_0-schema:jar:1.1.0_3
- org.codice.thirdparty:picocontainer:jar:1.2_1
- org.codice.thirdparty:vecmath:jar:1.3.2_1
- org.codice:lux:jar:1.2
- org.cometd.java:bayeux-api:jar:3.0.9
- org.cometd.java:cometd-java-annotations:jar:3.0.9
- org.cometd.java:cometd-java-client:jar:3.0.7
- org.cometd.java:cometd-java-client:jar:3.0.9
- org.cometd.java:cometd-java-common:jar:3.0.9
- org.cometd.java:cometd-java-server:jar:3.0.9
- org.eclipse.jetty:jetty-http:jar:9.2.19.v20160908
- org.eclipse.jetty:jetty-server:jar:9.2.19.v20160908
- org.eclipse.jetty:jetty-servlet:jar:9.2.19.v20160908
- org.eclipse.jetty:jetty-util:jar:9.2.19.v20160908
- org.forgerock.commonsl18n-core:jar:1.4.2
- org.forgerock.commonsl18n-slf4j:jar:1.4.2
- org.geotools.xsd:gt-xsd-gml3:jar:8.4
- org.geotools:gt-cql:jar:13.0
- org.geotools:gt-cql:jar:8.4
- org.geotools:gt-epsg-hsql:jar:8.4
- org.geotools:gt-jts-wrapper:jar:8.4
- org.geotools:gt-main:jar:8.4
- org.geotools:gt-opengis:jar:8.4
- org.geotools:gt-referencing:jar:8.4
- org.geotools:gt-shapefile:jar:8.4
- org.geotools:gt-xml:jar:8.4
- org.glassfish.grizzly:grizzly-framework:jar:2.3.14
- org.glassfish.grizzly:grizzly-http-server:jar:2.3.17
- org.imgscalr:imgscalr-lib:jar:4.2
- org.jasig.cas:cas-client-core:jar:3.1.10
- org.jasypt:jasypt:jar:1.9.0
- org.jcodec:jcodec:jar:0.2.0_1

- org.jdom:jdom:jar:2.0.2
- org.joda:joda-convert:jar:1.2
- org.jolokia:jolokia-osgi:jar:1.2.3
- org.jscience:jscience:jar:4.3.1
- org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.6.0
- org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.9.4
- org.jvnet.ogc:filter-v_2_0_0-schema:jar:1.1.0
- org.jvnet.ogc:gml-v_3_1_1-schema:jar:1.0.2
- org.jvnet.ogc:gml-v_3_1_1-schema:jar:1.0.3
- org.jvnet.ogc:gml-v_3_1_1-schema:jar:1.1.0
- org.jvnet.ogc:gml-v_3_2_1-schema:jar:1.1.0
- org.jvnet.ogc:gml-v_3_2_1:pom:1.1.0
- org.jvnet.ogc:ogc-tools-gml-jts:jar:1.0.3
- org.jvnet.ogc:ows-v_1_0_0-schema:jar:1.1.0
- org.jvnet.ogc:ows-v_1_1_0-schema:jar:1.1.0
- org.jvnet.ogc:wcs-v_1_0_0-schema:jar:1.1.0
- org.keyczar:keyczar:jar:0.66
- org.locationtech.spatial4j:spatial4j:jar:0.6
- org.noggit:noggit:jar:0.6
- org.objenesis:objenesis:jar:2.1
- org.openexi:nagasena-rta:jar:0000.0002.0049.0
- org.openexi:nagasena:jar:0000.0002.0049.0
- org.opensaml:opensaml-core:jar:3.1.1
- org.opensaml:opensaml-soap-impl:jar:3.1.1
- org.opensaml:opensaml-xmlsec-api:jar:3.1.1
- org.ops4j.pax.exam:pax-exam-container-karaf:jar:4.8.0
- org.ops4j.pax.exam:pax-exam-junit4:jar:4.8.0
- org.ops4j.pax.exam:pax-exam-link-mvn:jar:4.8.0
- org.ops4j.pax.exam:pax-exam:jar:4.8.0
- org.ops4j.pax.swissbox:pax-swissbox-extender:jar:1.3.1
- org.ops4j.pax.tinybundles:tinybundles:jar:2.1.1
- org.ops4j.pax.url:pax-url-aether:jar:2.4.5
- org.ops4j.pax.url:pax-url-wrap:jar:2.4.5
- org.osgi:org.osgi.compendium:jar:4.3.1
- org.osgi:org.osgi.compendium:jar:5.0.0

- org.osgi:org.osgi.core:jar:4.3.1
- org.osgi:org.osgi.core:jar:5.0.0
- org.ow2.asm:asm:jar:5.0.4
- org.parboiled:parboiled-java:jar:1.1.7
- org.quartz-scheduler:quartz:jar:2.1.7
- org.rrd4j:rrd4j:jar:2.2
- org.slf4j:slf4j-api:jar:1.7.1
- org.slf4j:slf4j-api:jar:1.7.12
- org.slf4j:slf4j-ext:jar:1.7.1
- org.slf4j:slf4j-log4j12:jar:1.7.7
- org.slf4j:slf4j-simple:jar:1.7.1
- org.slf4j:slf4j-simple:jar:1.7.5
- org.springframework.ldap:spring-ldap-core:jar:1.3.2.RELEASE
- org.springframework.osgi:spring-osgi-core:jar:1.1.0
- org.springframework:spring-core:jar:4.2.5.RELEASE
- org.taktik:mpegtts-streamer:jar:0.1.0_1
- us.bpsm:edn-java:jar:0.4.4
- xalan:serializer:jar:2.7.2
- xalan:xalan:jar:2.7.2
- xerces:xercesImpl:jar:2.11.0
- xerces:xercesImpl:jar:2.9.1
- xml-apis:xml-apis:jar:1.4.01
- xmlpull:xmlpull:jar:1.1.3.1
- xpp3:xpp3:jar:1.1.4c