



# **Distributed Data Framework**

## **Documentation**

Version 2.9.0. Copyright (c) Codice Foundation

# Table of Contents

1. License .....	1
2. Introduction .....	1
2.1. Applications .....	1
2.2. Documentation Guide .....	3
2.3. DDF Quick Start .....	5
3. Managing DDF .....	9
3.1. Installing DDF .....	9
3.2. Configuring DDF .....	15
3.3. Securing DDF .....	44
3.4. Running DDF .....	111
3.5. Troubleshooting DDF .....	142
4. Using DDF .....	144
4.1. Search .....	144
4.2. Actions .....	148
4.3. Workspaces .....	148
4.4. Notifications .....	149
4.5. Activities .....	150
4.6. Downloads .....	150
4.7. Maps .....	150
5. Managing DDF Admin .....	151
5.1. Installing DDF Admin .....	151
5.2. Configuring DDF Admin .....	151
5.3. Console Commands .....	151
5.4. Administrative User Interface .....	152
5.5. Admin Console Access Control .....	155
6. Managing DDF Catalog .....	157
6.1. Installing DDF Catalog .....	157
6.2. Configuring DDF Catalog .....	157
7. Managing DDF Platform .....	166
7.1. Using DDF Platform Application .....	166
7.2. Installing and Uninstalling Platform .....	166
8. Managing DDF Security .....	168
8.1. Installing DDF Security .....	168
9. Managing DDF Solr .....	171
9.1. DDF Catalog Solr External Provider .....	171
9.2. DDF Catalog Solr Embedded Provider .....	172
9.3. Standalone Solr Server .....	175
10. Managing DDF Spatial .....	179
10.1. Prerequisites .....	179
10.2. Installing .....	179
10.3. Optional Features .....	181
10.4. Console Commands .....	181

11. Managing DDF Search UI .....	183
11.1. Prerequisites .....	183
11.2. Installing DDF Search UI .....	183
11.3. Configuring DDF Search UI .....	183
11.4. Troubleshooting DDF Search UI .....	186
12. Integrating DDF .....	188
12.1. Understanding Metadata and Metacards .....	188
12.2. Populating Metacards (during ingest) .....	188
12.3. Searching Metadata .....	188
12.4. Catalog Search Result Objects .....	189
12.5. Asynchronous Search & Retrieval .....	191
13. Integrating DDF Admin .....	208
13.1. API .....	208
13.2. Implementation Details .....	211
14. Integrating DDF Catalog .....	212
14.1. Design .....	212
14.2. Integrating Endpoints .....	212
14.3. Developing a New Endpoint .....	233
14.4. DDF Data Migration .....	234
14.5. Integrating Catalog Framework .....	235
14.6. DDF Schematron .....	238
14.7. Adding New Attribute Types, Metocard Types, and Validators Using JSON Files .....	239
15. Integrating DDF Platform .....	246
15.1. System Properties .....	246
15.2. Platform UI Settings .....	246
15.3. Landing Page .....	248
15.4. DDF Mime Framework .....	250
15.5. Metrics Collection .....	254
15.6. Metrics Reporting Application .....	257
15.7. Security Core API .....	266
15.8. Compression Services .....	267
16. Integrating DDF Security .....	269
16.1. Security CAS .....	269
16.2. Security Core .....	272
16.3. Security Encryption API .....	274
16.4. Security LDAP .....	276
16.5. Installing the Embedded LDAP Server .....	276
16.6. Security PEP .....	283
16.7. Security STS .....	284
16.8. Security STS Service .....	289
16.9. Security PDP .....	291
16.10. Security PDP AuthZ Realm .....	291
16.11. Security IdP .....	293
17. Integrating DDF Solr .....	295

17.1. Embedded Solr Catalog Provider Pros and Cons .....	295
17.2. Solr Catalog External Provider Pros and Cons .....	296
18. Integrating DDF Spatial .....	298
18.1. Integrating DDF with CSW .....	298
18.2. CSW v2.0.2 Source .....	350
18.3. GMD CSW APISO v2.0.2 Source .....	353
18.4. Integrating DDF with KML .....	356
18.5. KML Network Link Endpoint .....	356
18.6. KML Query Response Transformer .....	359
18.7. KML Metocard Transformer .....	363
18.8. KML Style Mapper .....	367
18.9. Integrating DDF with WFS .....	371
18.10. Working with WFS Sources .....	371
18.11. WFS v1.0.0 Source .....	373
18.12. WFS v2.0.0 Source .....	375
19. Integrating DDF Search UI .....	381
19.1. CometD .....	381
19.2. Notifications .....	382
20. Extending DDF .....	383
20.1. Building DDF .....	383
20.2. Developing for DDF .....	385
20.3. DDF Development Guidelines .....	385
20.4. Development Recommendations .....	389
20.5. "White Box" DDF Architecture .....	390
20.6. OSGi Services .....	393
20.7. Developing DDF Applications .....	402
20.8. Migration API .....	407
20.9. Developing CometD Clients for Asynchronous Search and Retrieval .....	408
20.10. Web Service Security Architecture .....	411
20.11. Expansion Service .....	449
20.12. Securing SOAP .....	450
21. Extending DDF Admin .....	451
21.1. Whitelist .....	451
22. Extending DDF Catalog .....	452
22.1. Whitelist .....	452
22.2. Catalog Architecture .....	453
22.3. Catalog Application Services .....	453
22.4. Catalog Development Fundamentals .....	455
22.5. Working with Filters .....	457
22.6. Extending Catalog Plugins .....	472
22.7. Extending Operations .....	485
22.8. Extending Catalog Framework .....	493
22.9. Catalog Fanout Framework .....	497
22.10. Extending Sources .....	506

22.11. Extending Catalog Transformers .....	516
22.12. Extending Federation .....	550
22.13. Extending Eventing .....	552
22.14. Extending Resource Components .....	561
23. Extending DDF Platform .....	572
23.1. Whitelist .....	572
23.2. Developing Action Components (Action Framework) .....	573
23.3. Developing Migratables .....	574
24. Extending DDF Security .....	577
24.1. Whitelist .....	577
24.2. Developing Token Validators .....	578
25. Extending DDF Solr .....	580
25.1. Whitelist .....	580
26. Extending DDF Spatial .....	581
26.1. Whitelist .....	581
27. Extending DDF Search UI .....	582
27.1. Whitelist .....	582

# 1. License

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

## 2. Introduction

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.

#### *DDF Administrative Application*

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

#### *DDF Catalog Application*

The DDF Catalog 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.

#### *DDF Platform Application*

The Platform application is considered to be a core application of the distribution. The Platform application has fundamental building blocks that the distribution needs to run.

#### *DDF Security Application*

The 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.

#### *DDF Solr Catalog Application*

The Solr Catalog Provider (SCP) is an implementation of the [CatalogProvider](#) interface using [Apache Solr](#) as a data store.

#### *DDF Spatial Application*

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

#### *DDF Standard Search UI*

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 and displayed on a globe or map, providing a visual representation of where the records were found.

## 2.2. Documentation Guide

### 2.2.1. Documentation Outline

The documentation is divided by user needs, with users categorized as Users, Administrators, Integrators, and Developers.

#### *Users*

Users are end users interacting with the applications at the most basic level. User help is available through the **Help** link in the User Interface.

#### *Administrators*

[Managing DDF](#) | Administrators will be installing, maintaining, and supporting existing applications. Sections titled with Managing support administrators.

#### *Integrators*

[Integrating DDF](#) | Integrators will use the existing applications to support their external frameworks. Sections titled with Integrating support integrators.

#### *Developers*

[Extending DDF](#) | Developers will build or extend the functionality of the applications. Sections titled with Extending support developers.

### 2.2.2. Conventions

The following conventions are used within this documentation:

#### Callouts

**TIP** This is a **Tip**, used to provide helpful information.

**NOTE** This is an **Informational Note**, used to emphasize points, remind users of beneficial information, or indicate minor problems in the outcome of an operation.

**IMPORTANT** This is an **Important Warning**, used to alert users about the possibility of an undesirable outcome or condition.

#### 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.

#### Code Values

Java objects, lines of code, or file properties are denoted with the **Monospace** font style. Example:

## 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.

### 2.2.3. Support

#### Questions about DDF

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

#### Documentation Updates

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

## 2.3. DDF Quick Start

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

These steps will demonstrate:

- Prerequisites
- Installation
- Ingesting into the DDF Catalog

### 2.3.1. Prerequisites

- Install/Upgrade to Java 8 [J2SE 8 SDK](#)
- 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.

*Setting JAVA\_HOME on \*NIX*

```
JAVA_HOME=/usr/java/jdk1.8.0_<BUILDNUMBER>
export JAVA_HOME
```

*Setting JAVA\_HOME on Windows*

```
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0
```

\*NIX

**WARNING** Unlink `/usr/bin/java` if it is already linked to a previous version of the JRE: [unlink /usr/bin/java](#)

Verify that the `JAVA_HOME` was set correctly.

\*NIX

**TIP**

```
echo $JAVA_HOME
```

Windows

```
echo %JAVA_HOME%
```

- DDF installation zip file.

- A web browser.
- For Linux systems, increase the file descriptor limit by editing `/etc/sysctl.conf` to include or change the following:

#### *File Descriptor Limit*

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

#### **WARNING**

*Restart*

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

#### *Restart Command*

```
init 6
```

### **2.3.2. Installing DDF**

1. Install DDF by unzipping the zip file.
2. 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 `< DISTRIBUTION_INSTALL_DIR >`. (Substitute the actual directory name.)
3. Start DDF by running the `< DISTRIBUTION_INSTALL_DIR >/bin/ddf` script (or `ddf.bat` on Windows).
4. The Command Console will display.

#### *Command Console Prompt*

```
ddf@local>
```

### **2.3.3. Configuring**

- a. Go to <https://localhost:8993/admin>.
- b. Enter the default username of `admin` and the password of `admin`.
- c. Follow the prompts installer prompts for a standard installation.
  - i. Click start to begin the setup process.
  - ii. Configure guest claims attributes or use defaults.
  - iii. Select Standard Installation.

On the System Configuration page, configure any port or protocol changes desired and add any keystores/truststores needed.

v. Click **Next**

vi. Click **Finish**

### 2.3.4. Catalog Capabilities

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

### 2.3.5. Using the Catalog Content Directory Monitor

Using the Catalog framework's directory monitor, ingest a file so that it is stored in the content repository with a metocard created and inserted into the Catalog.

1. In the {admin-console}, select the **DDF Catalog** application.
2. Select **Configuration** tab.
3. Select the **Content Directory Monitor**.
4. Set the directory path to **inbox**.
5. Set **Processing Directive** to **Store and Process** (default).
6. Click the **Save** button.
7. Copy the attached **geojson** file to the **<DISTRIBUTION\_INSTALL\_DIR>/inbox** directory.

The Catalog Framework will:

- a. ingest the file,
- b. store it in the content repository at **<DISTRIBUTION\_INSTALL\_DIR>/content/store/<GUID>/geojson\_valid.json**,
- c. look up the GeoJson Input Transformer based on the mime type of the ingested file,
- d. create a metocard based on the metadata parsed from the ingested GeoJson file, and
- e. insert the metocard into the Catalog using the **CatalogFramework**.

**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.

# 3. Managing DDF

## 3.1. Installing DDF

### 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.

### 3.1.1. Prerequisites

#### Directory Permissions

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

#### Directory Permissions on Windows

1. Right-click on the file or directory noted below then select **Full Control      Administrators System**.
2. Click **Properties      Security      Advanced** and select Creator Owner for **INSTALLATION\_HOME** (e.g., **C:\ddf**).
3. Restrict access to sensitive files by ensuring that only **System** and **Administrators** have Full Control to the below files by right-clicking on the file or directory below then selecting **Properties      Security      Advanced**
4. Delete any other groups or users listed with access to **INSTALLATION\_HOME/etc** and **INSTALLATION\_HOME/deploy**.
  - Install/Upgrade to Java 8 **J2SE 8 SDK**
  - 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.

#### Setting **JAVA\_HOME** on \*NIX

```
JAVA_HOME=/usr/java/jdk1.8.0  
export JAVA_HOME
```

#### Setting **JAVA\_HOME** on Windows

```
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0
```

**\*NIX**

**WARNING** Unlink `/usr/bin/java` if it is already linked to a previous version of the JRE: `unlink /usr/bin/java`

Verify that the `JAVA_HOME` was set correctly.

**\*NIX**

**TIP**

```
echo $JAVA_HOME
```

*Windows*

```
echo %JAVA_HOME%
```

- DDF installation zip file.
- A web browser.
- For Linux systems, increase the file descriptor limit by editing `/etc/sysctl.conf` to include or change the following:

#### *File Descriptor Limit*

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

**WARNING** *Restart*  
For the change to take effect, a restart is required.

#### *Restart Command*

```
init 6
```

#### **Directory Permissions on \*NIX**

Protect the DDF from unauthorized access.

1. As root, change the owner and group of critical DDF directories to the `NON_ROOT_USER`.

**WARNING** A `NON_ROOT_USER` (e.g., `ddf`) is recommended for installation.

```
chown -R NON_ROOT_USER DDF_HOME DDF_HOME/etc DDF_HOME/data
chgrp -R NON_ROOT_USER DDF_HOME/etc DDF_HOME/data
chmod -R og-w DDF_HOME/etc DDF_HOME/data
```

2. Restrict access to sensitive files by ensuring that the only users with “group” permissions (e.g., `ddf`-

group) have access.

3. Execute the following command on the above files (examples assume `INSTALLATION_HOME` is `/opt/ddf`):

```
chmod -R o /opt/ddf
```

4. As the the application owner (e.g., ddf user), restrict access to sensitive files.

```
chmod 640 /opt/ddf/etc  
chmod 640 /opt/ddf/deploy
```

**IMPORTANT**

The system administrator must restrict certain directories to ensure that the application (user) cannot access restricted directories on the system. For example the `NON_ROOT_USER` should only have read access to `/opt/ddf`.

### 3.1.2. Deployment Guidelines

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

1. Prior to allowing a hot deployment, check the available storage space on the system to ensure the deployment will not exceed the available space.
2. Set maximum storage space on the `INSTALLATION_HOME/deploy` and `INSTALLATION_HOME/system` directories to restrict the amount of space used by deployments.
3. Do not assume the deployment is from a trusted source; verify its origination.
4. Use the source code to verify a deployment is required for DDF to prevent unnecessary/vulnerable deployments.

### 3.1.3. Installing With the DDF Distribution Zip

- After the prerequisites have been met, as a root user if for \*NIX, change the current directory to the desired install location. This will be referred to as `<INSTALL_DIRECTORY>`.

*\*NIX Tip*

**TIP**

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). The non-root user (e.g., ddf-user) can now be used for the remaining installation instructions.

*Example: Create a Directory and Switch User on \*NIX*

```
mkdir new_installation  
chown ddf-user:ddf-group new_installation  
  
su - ddf-user
```

- Change the current directory to location of zip file (ddf-X.Y.zip).

*Example: Where the Zip File may be Located in \*NIX*

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

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

```
cd D:\
```

- Copy ddf-X.Y.zip to <INSTALL\_DIRECTORY>.

*\*NIX*

```
cp ddf-X.Y.zip <INSTALL_DIRECTORY>
```

*Windows*

```
copy ddf-X.Y.zip <INSTALL_DIRECTORY>
```

- Change the current directory to the desired install location.

*\*NIX or Windows*

```
cd <INSTALL_DIRECTORY>
```

- The DDF zip is now located within the <INSTALL\_DIRECTORY>. Unzip ddf-X.Y.zip.

*\*NIX*

```
unzip ddf-X.Y.zip
```

*Example: Use Java to Unzip in Windows*

```
"C:\Program Files\Java\jdk1.8.0\bin\jar.exe" xf ddf-X.Y.zip
```

Run DDF using the appropriate script.

\*NIX

```
<INSTALL_DIRECTORY>/ddf-X.Y/bin/ddf
```

Windows

```
<INSTALL_DIRECTORY>/ddf-X.Y/bin/ddf.bat
```

Execute the following command at the command line for status:

*View Status*

```
ddf@local>list
```

- If the DDF Standalone Solr Server will be installed later, an additional configuration step is required for the DDF kernel. Add the following lines to the bottom of the `<INSTALL_DIR>/etc/org.ops4j.pax.web.cfg` file:

*Additional Configuration Step*

```
# Jetty Configuration  
'org.ops4j.pax.web.config.file=<KARAF.HOME>/etc/jetty.xml'
```

- Run the DDF using the appropriate script.

\*NIX

```
<INSTALL_DIRECTORY>/ddf-2.9.0/bin/ddf
```

Windows

```
<INSTALL_DIRECTORY>/ddf-2.9.0/bin/ddf.bat
```

The distribution takes a few moments to load depending on the hardware configuration. Execute the following command at the command line for status:

*View Status*

```
DDF@local>list
```

The list of bundles should look similar to this:

## *DDF List of Apps Installed*

```
.DDF@local>list
START LEVEL 100 , List Threshold: 50
   ID  State      Blueprint      Spring      Level  Name
[ 111] [Active] [ ] [ ] [ ] [ 80] Commons IO (2.1.0)
[ 113] [Active] [Created] [ ] [ ] [ 80] DDF :: Distribution :: Console
Branding Plugin (2.9.0)
```

### *DDF Application Installation Dependencies*

If completing a non-standard installation, be aware that some applications depend on other DDF applications being installed.

This hierarchy can be shown using the `app:tree` command

#### **WARNING**

```
ddf@local>app:tree
+- opendj-embedded
+- platform-app
|   +- catalog-app
|   |   +- search-ui-app
|   |   |   +- spatial-app
|   |   +- solr-app
|   +- security-services-app
|   |   +- admin-app
```

## **Verifying Installation**

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).

The following command displays the status of all the DDF bundles:

#### *View Status*

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

#### **WARNING**

If displayed, the `DDF :: Distribution :: Web Console` entry should be in the **Resolved** state. This is expected. The DDF Distribution Web Console is an OSGi bundle fragment. Bundle fragments are distinguished from other bundles in the command line console list by a new line under the its bundle status that states `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*

```
[ 261] [Resolved] [ ] [ ] [ 80] DDF :: Distribution :: Web Console  
(2.2.0)
```

Hosts: 76

## 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.

Director y Name	Description
bin	Scripts to start and stop DDF
data	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) informational messages. This log rolls up to 10 times, frequency based on a configuration file (10 MB).
	<code>data/log/ingest.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 audit purposes.
deploy	Hot-deploy directory – KARs and bundles added to this directory will be hot-deployed (Empty upon DDF startup)
document ation	HTML and PDF copies of DDF documentation.
etc	Directory monitored for addition/modification/deletion of <code>.config</code> configuration files or third party <code>.ctn</code> files.
	<code>etc/failed</code>
	If there is a problem with any of the <code>.config</code> files, such as missing tokens, they will be moved here.
	<code>etc/processed</code>
	All successfully processed <code>.config</code> files will be moved here.
	<code>etc/templates</code>
	Template <code>.config</code> files for use in configuring DDF sources, copied to the etc directory.
lib	The system's bootstrap libraries. Includes the <code>ddf-branding.jar</code> file which is used to brand the system on the command line.
licenses	Licensing information related to the system.
system	Local bundle repository. Contains all of the JARs required by DDF, including third-party JARs.

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

## 3.2. Configuring DDF

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. [\[\\_configuring\\_ddf\\_from\\_the\\_admin\\_console\]](#)
- Using a terminal and the Command Console. [\[\\_configuring\\_ddf\\_from\\_the\\_command\\_console\]](#)
- Editing configuration files. [\[\\_configuring\\_ddf\\_from\\_configuration\\_files\]](#)
- Importing the configurations from an existing DDF instance. [\[\\_importing\\_configurations\]](#)

### 3.2.1. Configuring DDF From the Admin Console

#### Accessing the Admin Console

- Open the admin portal. (Default: <https://localhost:8993/admin>)
- Enter Username and Password. (Default: **admin/admin**)

#### Initial Configuration

The first time the DDF Admin Console runs, the initial configuration steps appear.

1. Click **Start** to begin.
2. On the next screen, general configuration settings such as host address, port and site name can all be configured. (See [Configuring DDF Global Settings](#) for important settings to configure)
3. Next, choose between a standard installation and a full installation. (Individual applications can be added, removed or deactivated later)

**WARNING** Platform App, Admin App, and Security Services App CANNOT be selected or unselected as it is installed by default and can cause errors if removed.

#### Viewing Currently Active Applications from Admin Console

##### 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.

Either view has an **>** arrow to view more information about the application as currently configured.

## 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 either required for, 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 Applications

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.

#### IMPORTANT

Deactivating the `platform-app`, `admin-app`, and `security-services-app` will cause errors within the system, so the capabilities to do so have been DISABLED.

### Adding Applications

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 but 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 Features Using the Admin Console

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**.

## Add Feature Repositories

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**, **.jar**, OR
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.

## 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.

## 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 **DDF 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**.

### 3.2.2. Configuring DDF From the Command Console

**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 DDF Admin Console might be denied. It is necessary to configure DDF (e.g., providers, Schematron rulesets, etc.) using `.config` files or the Admin Console. Configuration via the Karaf command line console is not supported and may result in configuration errors. The OSGi container detects the creation of `.config` files in the `etc/` directory. The following sections describe how to configure each DDF item using both of these mechanisms. 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 at a later time the Admin Console is enabled again, all of the configuration done via `.config` files is loaded and displayed. However, note that the name of the `.config` file is not used in the Admin Console. Rather, OSGi assigns a universally unique identifier (UUID) when the DDF item was created and displays this UUID in the console (e.g., `OpenSearchSource.112f298e-26a5-4094-befc-79728f216b9b`)

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

### Configuring Using a `.cfg` File Template

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

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. **Mandatory:** The dash between the PID (e.g., `OpenSearchSource_site.cfg`) and the instance name (e.g., `OpenSearchSource_site.cfg`) is required. The dash is a reserved character used by OSGi that identifies instances of a managed service factory that should be created.
  - b. Not required, but a good practice is to change the instance name (e.g., `federated_source`) of the file to something identifiable (`source1- ddf`).
2. Edit the copied file to etc with the settings for the configuration. (Refer to the table above to determine the configurable properties).
- a. This file is a Java properties file, hence the syntax is `<key> = <value>`.
  - b. Consult the inline comments in the file for guidance on what to modify.
  - c. The Configurable Properties tables in the Integrator's Guide for the Included Catalog Components also describe each field and its value.

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

## Managing Applications From the Command Console

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

*Table 1. App Commands*

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

## Managing Features From 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	Version	Name	Repository
Description			
[installed ] [2.9.0 ] security-handler-api			security-services-app-
2.9.0 API for authentication handlers for web applications.			
[installed ] [2.9.0 ] security-core			security-services-app-
2.9.0 DDF Security Core			
[uninstalled] [2.9.0 ] security-expansion			security-services-app-
2.9.0 DDF Security Expansion			
[uninstalled] [2.9.0 ] security-cas-client			security-services-app-
2.9.0 DDF Security CAS Client.			
[uninstalled] [2.9.0 ] security-cas-tokenvalidator			security-services-app-
2.9.0 DDF Security CAS Validator for the STS.			
[uninstalled] [2.9.0 ] security-cas-cxfservlefilter			security-services-app-
2.9.0 DDF Security CAS Servlet Filter for CXF.			
[installed ] [2.9.0 ] security-pdp-authz			security-services-app-
2.9.0 DDF Security PDP.			
[uninstalled] [2.9.0 ] security-pep-serviceauthz			security-services-app-
2.9.0 DDF Security PEP Service AuthZ			
[uninstalled] [2.9.0 ] security-expansion-user-attributes			security-services-app-
2.9.0 DDF Security Expansion User Attributes Expansion			
[uninstalled] [2.9.0 ] security-expansion-metocard-attributes			security-services-app-
2.9.0 DDF Security Expansion Metocard Attributes Expansion			
[installed ] [2.9.0 ] security-sts-server			security-services-app-
2.9.0 DDF Security STS.			
[installed ] [2.9.0 ] security-sts-realm			security-services-app-
2.9.0 DDF Security STS Realm.			
[uninstalled] [2.9.0 ] security-sts-ldaplogin			security-services-app-
2.9.0 DDF Security STS JAAS LDAP Login.			
[uninstalled] [2.9.0 ] security-sts-ldapclaimshandler			security-services-app-
2.9.0 Retrieves claims attributes from an LDAP store.			

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>)
```

## Uninstall Features

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

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

State	Version	Name	Repository
<b>Description</b>			
[installed]	[2.9.0]	] ddf-core	ddf-2.9.0
[uninstalled]	[2.9.0]	] ddf-sts	ddf-2.9.0
[installed]	[2.9.0]	] ddf-security-common	ddf-2.9.0
[installed]	[2.9.0]	] ddf-resource-impl	ddf-2.9.0
[installed]	[2.9.0]	] ddf-source-dummy	ddf-2.9.0

## 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]	[2.9.0]	] ddf-core	ddf-2.9.0	
[uninstalled]	[2.9.0]	] ddf-sts	ddf-2.9.0	
[installed]	[2.9.0]	] ddf-security-common	ddf-2.9.0	
[installed]	[2.9.0]	] ddf-resource-impl	ddf-2.9.0	
[uninstalled]	[2.9.0]	] ddf-source-dummy	ddf-2.9.0	

## Configuring HTTP Port from the Command Console

### 3.2.3. Configuring HTTP to HTTPS Proxy From the Command Console

**NOTE** If DDF has not been installed, use the [\[How to install this feature using the Admin Console\]](#) guide.

#### 1. Type the command `feature:install platform-http-proxy`

### 3.2.4. Configuring DDF From Configuration Files

### 3.2.5. Configuring DDF Global Settings

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.

*Table 2. Configurable Properties*

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>Truststorejavax.net.ssl.trustStore</code>	String	The trust store used for SSL/TLS connections. Path is relative to <INSTALL_HOME>.	<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	
Global URL Properties						
Protocol	<code>org.codice.ddf.system.protocol</code>	String	Default protocol that should be used to connect to this machine.	<code>https://</code>	Yes	
Hostname	<code>org.codice.ddf.system.hostname</code>	String	<p>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.</p> <p>NOTE: Does not change the address the system runs on.</p>	<code>localhost</code>	Yes	

HTTPS Port	<code>org.codice.ddf.system.httpsPort</code>	String	The https port used by the system.  NOTE: This <b>DOES</b> change the port the system runs on.	8993	Yes
HTTP Port	<code>org.codice.ddf.system.httpPort</code>	String	The http port used by the system.  NOTE: This <b>DOES</b> change the port the system runs on.	8181	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.	8993	Yes
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.id</code>	String	The site name for DDF.	<code>ddfdistribution</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.	2.9.0	Yes
Organization	<code>org.codice.ddf.system.organization</code>	String	The organization responsible for this installation of DDF.	Codice Foundation	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	128	Yes
------------------	---	---------	--	-----	-----

HTTPS Specific Settings						
Cipher Suites	<code>https.cipherSuites</code>	String	Cipher suites to use with secure sockets	<code>TLS_DHE_RSA_WITH_AES_128_CBC_SHA,</code> <code>TLS_DHE_RSA_WITH_AES_128_CBC_SHA,</code> <code>TLS_DHE_DSS_WITH_AES_128_CBC_SHA,</code> <code>TLS_RSA_WITH_AES_128_CBC_SHA</code>	<code>TLS_DHE_RSA_WITH_AES_128_CBC_SHA,</code> <code>TLS_DHE_RSA_WITH_AES_128_CBC_SHA,</code> <code>TLS_DHE_DSS_WITH_AES_128_CBC_SHA,</code> <code>TLS_RSA_WITH_AES_128_CBC_SHA</code>	No
Https Protocols	<code>https.protocols</code>	String	Protocols to allow for secure connections	<code>TLSv1.1,TLSv1.2</code>	<code>TLSv1.1,TLSv1.2</code>	No
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>	<code>false</code>	Yes
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>	<code>org.apache.xerces.jaxp.DocumentBuilderFactoryImpl</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.

#### IMPORTANT

Since certain bundles can only be configured using the `.config` file format, this file format should be used.

#### WARNING

Only root can access ports < 1024 on Unix systems. For suggested ways to run DDF with ports < 1024 see [\[How do I use port 80 as a non-root user?\]](#).

## Configuring DDF .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.

## Configuring Managed Service Factory Bundles

Services that are created using a Managed Service Factory can be configured using `.config` files as well. The configuration files follow a different naming convention however. 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.uniqueID1.config` and `org.codice.ddf.factory.pid.uniqueID2.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 number 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.

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>authenticationTypes = [ "/\=SAML GUEST", "/admin\=SAML basic", "/jolokia\=SAML basic", "/system\=basic", "/solr\=SAML PKI basic", "/sources\=SAML 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 \
- Boolean values will default to `false` if any value other than `true` is provided

## Sample configuration file

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

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

realms=[ "/\=karaf"]

requiredAttributes=[ "/\=", "/admin\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims
/role\=admin}", "/solr\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\=admin
}", "/jolokia\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\=admin}", "/syst
em\={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", "/servic
es/saml", "/proxy", "/services/csw"]
```

### Editing HTTP Ports for Multiple Local DDF Nodes

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.ma nagement.cfg	rmiRegistryPort	1099	1199
	rmiServerPort	44444	44445
etc/system.properties	httpsPort, port	8993	8994
	httpPort	8181	8281

### 3.2.6. Editing DDF Web Service Providers Configuration Files

**IMPORTANT** If the hostname is changed during the install to something other than `localhost` a new keystore and truststore must be provided.

**TIP** When changing the hostname for testing or development purposes, the installer can be started with a `?dev=true` URL query parameter. This will cause the system to automatically generate self signed certificates for any hostname that is entered during the install process.

### Configuring Files in HOME Directory Hierarchy

**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.

- In <DDF\_HOME>/etc/users.properties, modify the line:

```
localhost=localhost,group,admin,manager,viewer,webconsole,system-admin
```

To be:

```
<FQDN>=<PASSWORD>,group,admin,manager,viewer,webconsole,system-admin
```

- Next, configure <DDF\_HOME>/etc/system.properties

```
#START DDF SETTINGS
# Set the keystore and truststore Java properties
javax.net.ssl.keyStore=etc/keystores/serverKeystore.jks
javax.net.ssl.keyStorePassword=<NewPassword>
javax.net.ssl.trustStore=etc/keystores/serverTruststore.jks
javax.net.ssl.trustStorePassword=<NewPassword>
javax.net.ssl.keyStoreType=jks

# Set the global url properties
org.codice.ddf.system.protocol=https://
org.codice.ddf.system.hostname=<FQDN>
org.codice.ddf.system.httpsPort=8993
org.codice.ddf.system.httpPort=8181
org.codice.ddf.system.port=8993
org.codice.ddf.system.rootContext=/services

# HTTPS Specific settings. If making a Secure Connection not leveraging the HTTPS Java
libraries and
# classes (e.g. if you are using secure sockets directly) then you will have to set this
directly
https.cipherSuites=TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_
DHE_DSS_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA
https.protocols=TLSv1.1,TLSv1.2
```

### 3.2.7. Configuring Notifications

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

## Using Notifications

DDF notifications are currently being utilized in the DDF Catalog application for resource retrieval. When a user initiates a resource retrieval via the DDF Standard Search UI, DDF opens the channel [/ddf/notification/catalog/downloads](#), where notifications indicating the progress of that resource download are sent. Any client interested in receiving these progress notifications must subscribe to that channel. When 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 Standard 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 Standard 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:8993/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?t=transform=resource&user=1w1ql079j6tscii19jszwp9s2i55` notifications contain the following information:

Parameter Name	Description	Required by DDF Standard UI
<code>application</code>	"Downloads" for resource retrieval. This is used as a "type" or category of messages.	Yes
<code>title</code>	Resource/file name for resource retrieval.	Yes
<code>message</code>	Human-readable message containing status and a more detailed message.	Yes
<code>timestamp</code>	Timestamp in milliseconds of when event occurs.	Yes
<code>user</code>	CometD Session ID or unique User ID.	Yes
<code>status</code>	Status of event.	No
<code>option</code>	Resource retrieval option.	No
<code>bytes</code>	Number of bytes transmitted.	No

## Receive Notifications

- If interested in retrieve resource notifications, a client must subscribe to the CometD [channel/ddf/notification/catalog/downloads](#).
- If interested in all notification types, a client must subscribe to the CometD [channel/ddf/notification/\\*\\*](#)

- A client will only receive notifications for resources they have requested.
- DDF Standard UI is subscribed to all notifications of interest to that `user/browser session: /ddf/notification/**`

### Publish Notifications

Any application running in DDF can publish notifications that can be viewed by the DDF Standard UI or received by another notifications client. Set a properties map containing entries for each of the parameters listed above in the Usage section.

- + . Set the OSGi event topic to `ddf/notification/<application-name>/<notification-type>`. Notice that there is no preceding slash on an OSGi event topic name, while there is one on the CometD channel name. The OSGi event topic corresponds to the CometD channel this is published on.
- + . Post the notification to the OSGi event defined in the previous step.

#### *Example for Publishing Notification*

```
Dictionary<String, Object> properties = new Hashtable<String, Object>();
properties.put("application", "Downloads");
properties.put("title", resourceResponse.getResource().getName());
Long sysTimeMillis = System.currentTimeMillis();
properties.put("message", generateMessage(status, resourceResponse.getResource().getName(),
(), bytes, sysTimeMillis, detail));
properties.put("user", getProperty(resourceResponse, USER));
properties.put("status", "Completed");
properties.put("bytes", 1024);
properties.put("timestamp", sysTimeMillis);

Event event = new Event("ddf/notification/catalog/downloads", properties);

eventAdmin.postEvent(event);
```

### 3.2.8. Configuring Solr Catalog Provider Data Directory

The 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.catalog.data.dir=/opt/solr/catalog/data
```

## Changing the Data Directory after DDF has ingested data

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

*Make new Data Directory*

```
mkdir /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.

*Set the SOLR\_CATALOG\_DATA\_DIR*

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

5. Restart DDF.

## Configuring Thread Pools

The `system.properties` file found under `DDF_HOME/etc` contains properties that will be made available through system properties at the beginning of Karaf's boot process. 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.

### 3.2.9. 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.

- 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 editable in the Admin Console will be exported to `<DDF_HOME>/etc/exported/etc`.

#### IMPORTANT

- All other configuration files (system configurations) will be put under `etc/exported/<ID>` followed by their relative path from `DDF_HOME`. For instance, if a key store file was located under `<DDF_HOME>/keystores/keystore.jks`, it will be exported to `<DDF_HOME>/etc/exported/<ID>/keystore/keystore.jks`.
- To keep the export/import process simple and consistent, all system configuration files are required to be under the `DDF_HOME` directory.

### 3.2.10. Exporting 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 do so, 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

### 3.2.11. Export Configuration Settings from Command Console

To export the current DDF configuration:

1. Using the Command Console, 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 exportedFiles.zip *
```

## Troubleshooting Common Warnings or Failures

### Insufficient Write Permissions

In the following case, the directory the user tried to export to had permissions set to read only.

### Properties Set to Absolute File Paths

In the following case, the user had a property set to an absolute file path. This is not allowed, and can be fixed by updating the property to a value that is relative to DDF home. However, notice that the export did not completely fail. It is simply informing the user that they did not include a specific file.

#### 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.

### 3.2.12. Import Configuration Settings

To import a previously exported configuration:

1. Delete all existing `.config` files from `<DDF_HOME>/etc`: `rm <DDF_HOME>/etc/*.config`
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 Line Console.

### 3.2.13. Managing Features

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.

### 3.2.14. Configuring DDF with New Certificates

DDF ships with a default security certificate configured to identify the instance machine as `localhost`. This allows the distribution to be unzipped and run immediately in a secure manner. 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 was left as `localhost` or the hostname was changed after installation, in order to access the DDF instance from another machine over HTTPS (now the default for many services) the default certificates need to be replaced with a certificates that use the fully qualified hostname of the server running the DDF instance.

*Table 3. Important Terms for Certificates*

Term	Definition	Example Value
<code>&lt;DDF_HOME&gt;</code>	The path to the unzipped DDF distribution	<code>/opt/ddf/ddf-2.9.0</code>
alias	The nickname given to a certificate within a keystore to make it easily identifiable. Normally, the alias should be the FQDN.	<code>localhost</code>
certificate	A combination of an entity's identity information with the entity's public key. The entity can be a person, organization, or something else, but in this case the entity is a computer on the network. To be valid, a certificate must be digitally (cryptographically) signed by a certificate authority. By signing a certificate, the CA attests that the public key truly belongs to the entity and no one else. See also <b>PKIX</b> .	<code>&lt;FQDN&gt;.crt</code>
CN	Common Name - The FQDN of the DDF instance as defined within the Certificate.	<code>search.codice.org</code>

Term	Definition	Example Value
certification path	<p>A list of certificates, starting with the server's certificate and followed by the certificate of the CA who signed the server's CSR.</p> <p>The list of certificates continues, with each subsequent certificate belonging to the CA that signed the current CA's certificate.</p> <p>This chain continues until it reaches a trusted anchor, or root CA certificate.</p> <p>The chain establishes a link between the trust anchor and the server's certificate.</p> <p>See <a href="#">IETF RFC 4158</a> for details.</p>	
chain of trust	See certification path.	
CSR	Certificate Signing Request. A certificate that has not yet been signed by a certificate authority.	<FQDN>.csr
digital certificate	See <b>certificate</b> .	
FQDN	Fully Qualified Domain Name	search.codice.org
HTTPS	<p>Hyper-Text Transfer Protocol Secure.</p> <p>An encrypted alternative to HTTP.</p> <p>The HTTP connection is encrypted over TLS.</p> <p>See <a href="#">IETF RFC 2818</a> for more information.</p>	<a href="https://">https://</a>
JKS	<p>Java <b>keystore</b>.</p> <p>A dictionary of cryptographic objects (e.g. private keys, certificates) referenced by an <b>alias</b>.</p> <p>The JKS format is specific to Java.</p>	
keystore	<p>Refers to either a JKS keystore or a PKCS#12 keystore.</p> <p>For the purposes of these instructions, a keystore is always a file.</p>	

Term	Definition	Example Value
keytool	The Java keytool is a key and certificate management command line utility.	
openssl	The openssl program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell.	
PKCS#12	<p>Personal Information Exchange Syntax.</p> <p>A standard that allows certificates, private keys, and optional attributes to be combined into a single file.</p> <p>See <a href="#">IETF RFC 7292</a> for more information.</p>	<FQDN>.p12
PKIX	<p>A public key infrastructure also known as X.509.</p> <p>It is documented in the <a href="#">IETF RFC 5280</a> and defines what a <b>certificate</b> is.</p>	
PORT	TCP Port of service	8993
security certificate	See <b>certificate</b> .	
TLS	<p>Transport Layer Security protocol.</p> <p>Provides privacy and data integrity between client and server.</p> <p>See <a href="#">IETF RFC 5246</a> for more information.</p>	

### 3.2.15. Configuring DDF 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_DIRECTORY>/etc/system.properties` file. See [Editing DDF Web Service Providers Configuration Files](#)

## Creating and Installing Keys and Certificates

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 <DDF\_HOME>/etc/certs directory. For \*NIX, use the CertNew.sh script.

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

The above command creates a new entry in the keystore for a server named `my.server.com`. To create and install the certificates on Windows, use the `CertNew.cmd` file in the same directory.

```
CertNew <FQDN>
```

To install a certificate signed by a different Certificate Authority, see [Import into a Java Keystore \(JKS\)](#).

## Restart and Test

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
```

## IMPORTANT

The Embedded LDAP has hard-coded values for the keystore path, truststore path, keystore password, and truststore password (<https://github.com/codice/opendj-osgi/blob/d5021cbac4db831467ceb109ffd7ffd2c734dcd4/embedded/opendj-embedded-server/src/main/resources/config/config.ldif>). So if using a non-default keystore and non-default truststore, the Embedded LDAP will not work. You will see errors in `<ddf-home>/etc/org.codice.opendj/ldap/logs/errors` similar to the one below:

```
'21/Jan/2015:08:58:57 -0700] category=CORE severity=NOTICE
msgID=458891 msg=The Directory Server has sent an alert notification
generated by class
org.opens.server.protocols.ldap.LDAPConnectionHandler (alert type
org.opens.server.LDAPHandlerDisabledByConsecutiveFailures, alert ID
2425016): The LDAP connection handler defined in configuration entry
cn=LDAP Connection Handler,cn=Connection Handlers,cn=config has
experienced consecutive failures while trying to accept client
connections: An error occurred while attempting to initialize the
SSL context for use in the LDAP Connection Handler: An error
occurred while trying to load the keystore contents from file
../../../../keystores/serverKeystore.jks: IOException(Keystore was
tampered with, or password was incorrect) (id=1310782)
(LDAPConnectionHandler.java:1324 LDAPConnectionHandler.java:1255
LDAPConnectionHandler.java:1091 LDAPConnectionHandler.java:974).
This connection handler will be disabled'
```

A workaround is to modify `config.ldif` as seen in the steps below and hot deploy `opendj-embedded-app-<version>.kar`.

- 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`

### 3.2.16. Configuring DDF to use 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](#), on the command line to set passwords for your LDAP server. Then change the LDAP Bind User Password in the configurations to use the encrypted password.

### 3.2.17. 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.

- `catalog-core-standardframework`
- `catalog-solr-embedded-provider`
- `catalog-opensearch-endpoint`
- `catalog-opensearch-souce`
- `catalog-rest-endpoint`

#### 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.

#### Description

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.

### 3.2.18. Configuring DDF Logging Service

The maximum number of log events to store can be configured in the Admin Console.

### 3.2.19. Managing Asynchronous Capabilities (Search & Retrieval)

## Installing the Asynchronous Capabilities Endpoint (CometD)

The CometD endpoint enables asynchronous search capabilities.

It is installed by default with the Search UI application.

- The feature used is `search-ui`. To verify the following command may be used:

```
ddf@local>features:list | grep -i search-ui
```

- The bundle is the DDF SearchUI Endpoint

```
DDF :: UI :: Search UI :: Endpoint
```

## Configuring the Product Cache

All caching properties are part of the Catalog Framework Configuration

Property	Type	Description	Default Value	Required
productCacheDirectory	String	<p>Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Catalog Framework will attempt to create those directories.</p> <p>Without configuration, the product cache directory is <code>&lt;INSTALL_DIR&gt;/data/productcache</code>. If a relative path is provided it will be relative to the <code>&lt;INSTALL_DIR&gt;</code>.</p> <p>It is recommended to enter an absolute directory path such as <code>/opt/productcache</code> in Linux or <code>C:\product-cache</code> in Windows.</p>	(empty)	No
cacheEnabled	Boolean	Check to enable caching of retrieved products to provide faster retrieval for subsequent requests for the same product.	true	no
delayBetweenRetryAttempts	Integer	The time to wait (in seconds) between each attempt to retry retrieving a product from the Source.	10	no

Property	Type	Description	Default Value	Required
maxRetryAttempts	Integer	The maximum number of attempts to try and retrieve a product from the Source.	3	no
cachingMonitorPeriod	Integer	The number of seconds allowed for no data to be read from the product data before determining that the network connection to the Source where the product is located is considered to be down.	5	no
cacheWhenCanceled	Boolean	Check to enable caching of retrieved products even if client cancels the download.	false	no

### Invalidating the Product Cache

1. The product cache directory can be administratively invalidated by turning off the product caching using the `cacheEnabled` property.
2. Alternatively, an administrator may manually invalidate products by removing them from the file system. Products are cached at the directory specified in the `productCacheDirectory` property. The following example assumes the `productCacheDirectory` has the default value of `<INSTALL-DIR>/data/product-cache`

Format:

`<INSTALL-DIR>/data/product-cache/<source-id>-<metacard-id>`

Example:

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

1. Set Max Caching Directory Size. The `cacheDirMaxSizeMegabytes` property can be used as a way to evict the oldest products from the cache. By setting this to a low limit, the oldest products in the cache will be removed as new products are placed in the cache to ensure the cache does not go over the max limit.

## 3.3. Securing DDF

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.

### 3.3.1. 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.

#### Configuring Web Context Policy Manager

The karaf realm is the only realm available by default and it authenticates against the `user.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. As you add REST endpoints, you may need to add different types of authentication through the Web Context Policy Manager.

*Table 4. Default Types of Authentication*

Authentication Type	Description
<code>saml</code>	Activates single-sign on (SSO) across all REST endpoints that use.
<code>basic</code>	Activates basic authentication.
<code>PKI</code>	Activates public key infrastructure authentication
<code>IdP</code>	
<code>CAS</code>	
<code>guest</code>	provides guest access

### **3.3.2. IdP/SP**

The DDF 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: **DDF Security    Features    security-idp**

#### **Security IdP Service Provider (SP)**

The IdP client that interacts with the specified Identity Provider.

#### **IdP SSO Configuration**

##### **Configuring**

1. Navigate to Admin Console    DDF Security    Configuration    IdP Client
2. 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>
            MIIDEzCCAnygAwIBAgIJAIZc4FYrIp9mA0GCSqGSIB3DQEBBQUAMhcxCzAJBgNVBAYTA1VTMQswC
            QYDVQQIDAJBWjEMMAoGA1UECgwDRERGMQwwCgYDVQLDANEZXYxGTAXBgNVBAMMEERiBEZW1vIFJvb3QgQ0ExJD
            AiBhgkqhkIG9w0BCQEWFRkZnJvb3RjYUBleGftcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MTh
            aMIGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWfymQwwCgYDVQQKDANEREYxDDAK
            BgNVBAsMA0RldjESMBAGA1UEAwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsB2NhbGhvc3RAZXhhbXBsZS5vc
            mcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZphHQfb5g8FrgBq1RYzV7ikVw/pVGkz8gx3l3A99
            s8WtA4mRAeb6n0vTR9yNB0ekW4nY0iEoq//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pk1r0TDI
            4MRzPPkkTp4w1ORQ/a6CfvNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBvhvCAQ0EGhYYRk9S
            IFRFU1RJTkcguFVSUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrDAfBgvNVHSMEGDAwgbThV
            MeX3wrCv6lfeF47CvkSBe9xjAgBgnVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQ
            ADgYEAtRUUp7fAxU/E6JD2Kj/+CTWqu8Elx1S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx07n1SmOESp5sE3jGTnh0Gt
            YV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0h1SGV0=
          </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>
            MIIDEzCCAnygAwIBAgIJAIZc4FYrIp9mA0GCSqGSIB3DQEBBQUAMhcxCzAJBgNVBAYTA1VTMQswC
            QYDVQQIDAJBWjEMMAoGA1UECgwDRERGMQwwCgYDVQLDANEZXYxGTAXBgNVBAMMEERiBEZW1vIFJvb3QgQ0ExJD
            AiBhgkqhkIG9w0BCQEWFRkZnJvb3RjYUBleGftcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MTh
            aMIGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWfymQwwCgYDVQQKDANEREYxDDAK
            BgNVBAsMA0RldjESMBAGA1UEAwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsB2NhbGhvc3RAZXhhbXBsZS5vc
            mcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZphHQfb5g8FrgBq1RYzV7ikVw/pVGkz8gx3l3A99
            s8WtA4mRAeb6n0vTR9yNB0ekW4nY0iEoq//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pk1r0TDI
            4MRzPPkkTp4w1ORQ/a6CfvNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBvhvCAQ0EGhYYRk9S
            IFRFU1RJTkcguFVSUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrDAfBgvNVHSMEGDAwgbThV
            MeX3wrCv6lfeF47CvkSBe9xjAgBgnVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQ
            ADgYEAtRUUp7fAxU/E6JD2Kj/+CTWqu8Elx1S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx07n1SmOESp5sE3jGTnh0Gt
            YV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0h1SGV0=
          </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>

```

### 3.3.3. Security IdP Server

An internal Identity Provider solution.

#### Configuring IdP Server

1. Navigate to Admin Console    DDF Security    Configuration    IdP Server
2. Click the + next to SP Metadata to add a new entry
3. 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>
            MIIDEzCCAnygAwIBAgIJIAIzc4FYrIp9mMA0GCSqGSib3DQEBBQUAMHcxCzAJBgNVBAYTA1VTMQswCQYDVQQIDAjBWjEMMAoGA1UECgwDRERGMQwwCgYDVQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBgkqhkiG9w0BCQEWFWRkZnJvb3RjYUB1eGFtcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMIGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWfYMQwwCgYDVQQKDANEREYxDDAKBgNVBAsMA0R1djESMBAGA1UEAwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyzNzbCTZphHQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx313A99s8WtA4mRAeb6n0vTR9yNB0ekW4nY0iE0q//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pk1r0TDI4MRzPPkkTp4w1ORQ/a6CfvNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBvhvCAQ0EGhYYRk9SIFRFU1RJTkcgUFVSVUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BrFrssDAfBgNVHSMEGDAwBThVMeX3wrCv61feF47CvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8E1x13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx07n1Sm0E5p5sE3jGTnh0GtYV0D219z/09n90cd/imAEhknJlavyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0h1SGV0=          </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>
            MIIDEzCCAnygAwIBAgIJIAIzc4FYrIp9mMA0GCSqGSib3DQEBBQUAMHcxCzAJBgNVBAYTA1VTMQswCQYDVQQIDAjBWjEMMAoGA1UECgwDRERGMQwwCgYDVQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBgkqhkiG9w0BCQEWFWRkZnJvb3RjYUB1eGFtcGx1Lm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMIGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWfYMQwwCgYDVQQKDANEREYxDDAKBgNVBAsMA0R1djESMBAGA1UEAwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyzNzbCTZphHQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx313A99s8WtA4mRAeb6n0vTR9yNB0ekW4nY0iE0q//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pk1r0TDI4MRzPPkkTp4w1ORQ/a6CfvNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBvhvCAQ0EGhYYRk9SIFRFU1RJTkcgUFVSVUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BrFrssDAfBgNVHSMEGDAwBThVMeX3wrCv61feF47CvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8E1x13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUx07n1Sm0E5p5sE3jGTnh0GtYV0D219z/09n90cd/imAEhknJlavyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0h1SGV0=          </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>
```

## Related Configuration

1. Navigate to Admin Console    DDF Security    Configuration    Web Context Policy Manager
2. 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=SAML | IDP

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

## 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.

## CAS Authentication

**NOTE** CAS Authentication Logging was obtained using a CAS war file deployed to a Tomcat application server. Tomcat allows configuration of the log file, but, by default, the logs below were stored in the \$TOMCAT\_HOME/logs/catalina.out file.

## Username and Password

### *Sample – Successful login*

```
2013-04-24 10:39:45,265 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler successfully authenticated [username: testuser1]>
2013-04-24 10:39:45,265 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <Resolved principal testuser1>
2013-04-24 10:39:45,265 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler@6a4d37e5 authenticated testuser1 with credential [username: testuser1].>
2013-04-24 10:39:45,265 INFO
[com.github.inspektr.audit.support.Slf4jLoggingAuditTrailManager] - <Audit trail record BEGIN
=====
WHO: [username: testuser1]
WHAT: supplied credentials: [username: testuser1]
ACTION: AUTHENTICATION_SUCCESS
APPLICATION: CAS
WHEN: Wed Apr 24 10:39:45 MST 2013
CLIENT IP ADDRESS: 127.0.0.1
SERVER IP ADDRESS: 127.0.0.1
=====
>
```

### *Sample – Failed login*

```
2013-04-24 10:39:17,443 INFO
[org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler] - <Failed to authenticate user testuser1 with error [LDAP: error code 49 - Invalid Credentials]; nested exception is javax.naming.AuthenticationException: [LDAP: error code 49 - Invalid Credentials]>
2013-04-24 10:39:17,443 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler failed authenticating [username: testuser1]>
2013-04-24 10:39:17,443 INFO
[com.github.inspektr.audit.support.Slf4jLoggingAuditTrailManager] - <Audit trail record BEGIN
=====
WHO: [username: testuser1]
WHAT: supplied credentials: [username: testuser1]
ACTION: AUTHENTICATION_FAILED
APPLICATION: CAS
WHEN: Wed Apr 24 10:39:17 MST 2013
CLIENT IP ADDRESS: 127.0.0.1
SERVER IP ADDRESS: 127.0.0.1
=====
>
```

## PKI Certificate

### NOTE

Current testing was performed using the OZone certificates that came with a `testAdmin` and `testUser`, which were signed by a common CA.

### *Sample – Successful login*

```
2013-04-24 15:13:14,388 INFO [org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler] - <Successfully authenticated CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US, SerialNumber=4>
2013-04-24 15:13:14,390 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler successfully authenticated CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US, SerialNumber=4>
2013-04-24 15:13:14,391 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <Resolved principal CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US>
2013-04-24 15:13:14,391 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler@1e5b04ae authenticated CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US with credential CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US, SerialNumber=4.>
2013-04-24 15:13:14,394 INFO [com.github.inspektr.audit.support.Slf4jLoggingAuditTrailManager] - <Audit trail record BEGIN
=====
WHO: CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US, SerialNumber=4
WHAT: supplied credentials: CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US, SerialNumber=4
ACTION: AUTHENTICATION_SUCCESS
APPLICATION: CAS
WHEN: Wed Apr 24 15:13:14 MST 2013
CLIENT IP ADDRESS: 127.0.0.1
SERVER IP ADDRESS: 127.0.0.1
=====
>
```

## Sample – Failed login

The failure was simulated using a filter on the x509 credential handler. This filter looks for a certain CN in the certificate chain and will fail if it cannot find a match. The server was set up to trust the certificate via the Java truststore, but there were additional requirements for logging in. For this test-case, the chain it was looking for is "CN=Hogwarts Certifying Authority.+". Example from the CAS wiki:  
<https://wiki.jasig.org/display/CASUM/X.509+Certificates>.

2013-04-25 14:15:47,477 DEBUG

[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler] - <Evaluating CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US, SerialNumber=4>

2013-04-25 14:15:47,478 DEBUG

[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler] - <.\* matches CN=testUser1, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US == true>

2013-04-25 14:15:47,478 DEBUG

[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler] - <CN=Hogwarts Certifying Authority.+ matches EMAILADDRESS=goss-support@owfgoss.org, CN=localhost, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US == false>

2013-04-25 14:15:47,478 DEBUG

[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler] - <Found valid client certificate>

2013-04-25 14:15:47,478 INFO

[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler] - <Failed to authenticate

org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc>

2013-04-25 14:15:47,478 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] - <org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler failed to authenticate

org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc>

2013-04-25 14:15:47,478 INFO

[com.github.inspektr.audit.support.Slf4jLoggingAuditTrailManager] - <Audit trail record BEGIN

=====

WHO:

org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc

WHAT: supplied credentials:

org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc

ACTION: AUTHENTICATION\_FAILED

APPLICATION: CAS

WHEN: Thu Apr 25 14:15:47 MST 2013

CLIENT IP ADDRESS: 127.0.0.1

SERVER IP ADDRESS: 127.0.0.1

=====

>

## STS Authentication

### Username and Password

*Sample – Successful login*

```
[INFO ] 2014-07-17 14:40:23,340 | qtp1401560510-76 | securityLogger | Username  
[pparker] successfully logged in using LDAP authentication. Request IP: 127.0.0.1, Port:  
52365  
[INFO ] 2014-07-17 14:40:24,074 | qtp1401560510-76 | securityLogger | Security Token  
Service REQUEST  
STATUS: SUCCESS  
OPERATION: Issue  
URL: https://server:8993/services/SecurityTokenService  
WS_SEC_PRINCIPAL:  
1.2.840.113549.1.9.1=#160d69346365406c6d636f2e636f6d,CN=client,OU=I4CE,O=Lockheed  
Martin,L=Goodyear,ST=Arizona,C=US  
ONBEHALFOF_PRINCIPAL: pparker  
TOKENTYPE: http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0  
CLAIMS_SECONDARY: [http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role,  
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]  
Request IP: 127.0.0.1, Port: 52365
```

### *Sample – Failed login*

```
[WARN ] 2014-07-17 14:42:43,627 | qtp1401560510-75 | securityLogger | Username  
[pparker] failed LDAP authentication. Request IP: 127.0.0.1, Port: 52386  
[WARN ] 2014-07-17 14:42:43,632 | qtp1401560510-75 | securityLogger | Security Token  
Service REQUEST  
STATUS: FAILURE  
OPERATION: Issue  
URL: https://server:8993/services/SecurityTokenService  
WS_SEC_PRINCIPAL:  
1.2.840.113549.1.9.1=#160d69346365406c6d636f2e636f6d,CN=client,OU=I4CE,O=Lockheed  
Martin,L=Goodyear,ST=Arizona,C=US  
ONBEHALFOF_PRINCIPAL: pparker  
TOKENTYPE: http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0  
CLAIMS_SECONDARY: [http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role,  
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]  
EXCEPTION: org.apache.cxf.ws.security.sts.provider.STSEException: The specified request  
failed  
Request IP: 127.0.0.1, Port: 52386
```

### **PKI Certificate**

#### *Sample – Successful login*

```
[INFO ] 2014-07-17 15:03:39,379 | qtp1401560510-74 | securityLogger | Security Token  
Service REQUEST  
STATUS: SUCCESS  
OPERATION: Issue  
URL: https://localhost:8993/services/SecurityTokenService  
WS_SEC_PRINCIPAL:  
1.2.840.113549.1.9.1=#160d69346365406c6d636f2e636f6d,CN=client,OU=I4CE,O=Lockheed  
Martin,L=Goodyear,ST=Arizona,C=US  
TOKENTYPE: http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0  
CLAIMS_SECONDARY: [http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role,  
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]  
Request IP: 127.0.0.1, Port: 52573
```

### *Sample – Failed login*

```
[WARN ] 2014-07-17 15:05:46,061 | qtp1401560510-75 | securityLogger | Security Token  
Service REQUEST  
STATUS: FAILURE  
OPERATION: Issue  
URL: N.A.  
TOKENTYPE: N.A.  
APPLIES TO: <null>  
EXCEPTION: org.apache.cxf.ws.security.sts.provider.STSEException: The request was invalid  
or malformed  
Request IP: 127.0.0.1, Port: 52582
```

### **Binary Security Token (CAS)**

#### *Sample – Successful Login*

```
15:27:48,098 | INFO | tp1343209378-282 | securityLogger |  
rity.common.audit.SecurityLogger 156 | 247 - security-core-api - 2.2.0.RC6-SNAPSHOT |  
Telling the STS to request a security token on behalf of the binary security token:  
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<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-wssecuri-  
tyext-1.0.xsd" xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
wssecurity-utility-  
1.0.xsd">U1QtMTctQmw0aGRrS05jaTV3cE82Zm11VE0tY2FzfGh0dHBz0i8vdG9rZW5pc3N1ZXi60Dk5My9zZXJ2  
aWNlcyc9TZWN1cmlo eVRva2VuU2VydmljZQ==</BinarySecurityToken>  
Request IP: 0:0:0:0:0:0:1%, Port: 53363  
15:27:48,351 | INFO | tp1343209378-282 | securityLogger |  
rity.common.audit.SecurityLogger 156 | 247 - security-core-api - 2.2.0.RC6-SNAPSHOT |  
Finished requesting security token. Request IP: 127.0.0.1, Port: 53363  
  
**This message will show when DEBUG is on**  
15:27:48,355 | DEBUG | tp1343209378-282 | securityLogger |  
rity.common.audit.SecurityLogger 102 | 247 - security-core-api - 2.2.0.RC6-SNAPSHOT |  
<?xml version="1.0" encoding="UTF-16"?>  
<saml2:Assertion>  
SAML ASSERTION WILL BE LOCATED HERE
```

## Sample – Failed Login

```
10:54:21,772 | INFO | qtp995500086-618 | securityLogger          |
rity.common.audit.SecurityLogger 143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Telling the STS to request a security token on behalf of the binary security
token:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<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">U1QtMjctOU43RUlkNHkzVFoxQmZCb0RIdkItY2Fz</BinarySecurityToken>
10:54:22,119 | INFO | qtp995500086-141 | securityLogger          |
rity.common.audit.SecurityLogger 143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Validating ticket [ST-27-9N7EId4y3TZ1BfBoDHvB-cas] for service
[https://server:8993/services/SecurityTokenService]. Request IP: 127.0.0.1, Port: 64548
10:54:22,169 | INFO | qtp995500086-141 | securityLogger          |
rity.common.audit.SecurityLogger 143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Unable to validate CAS token. Request IP: 127.0.0.1, Port: 64548
10:54:22,244 | INFO | qtp995500086-618 | securityLogger          |
rity.common.audit.SecurityLogger 143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Error requesting the security token from STS at:
https://server:8993/services/SecurityTokenService.
```

### 3.3.4. CAS SSO Configuration

The Web Service Security (WSS) Implementation that comes with DDF was built to run independent of an SSO or authentication mechanism. Testing out the security functionality of DDF was performed by using the Central Authentication Server (CAS) software. This is a popular SSO appliance and allowed DDF to be tested using realistic use cases. This page contains configurations and settings that were used to help enable CAS to work within the DDF environment.

#### General Server Setup and Configuration

**NOTE** The following procedure defines the steps for installing CAS to a Tomcat 7.x server running in Linux and Windows. Newer versions of tomcat (8.x) are incompatible with the included `server.xml` file and will need additional changes.

#### Install using DDF CAS WAR

DDF comes with a custom distribution of the CAS Web application that comes with LDAP and X.509 support configured and built-in. Using this configuration may save time and make setup easier.

**NOTE**

The CAS Web Application can be downloaded from Nexus. To find the latest version, execute a search for "cas-distribution". Link to the first release: <http://artifacts.codice.org/content/repositories/releases/org/codice/cas/distribution/cas-distribution/1.0.0/cas-distribution-1.0.0.war>

1. Download and unzip Tomcat Distribution. The installation location is referred to as <TOMCAT\_INSTALLATION\_DIR>.

```
$ unzip apache-tomcat-7.0.39.zip
```

2. Clone <https://github.com/codice/cas-distribution> to a convenient location. This folder will be referred to as **cas-distribution**.
3. Set up Keystores and enable SSL. There are sample configurations located within the **security-cas-server-webapp** project.

- a. Copy setenv (**cas-distribution/src/main/resources/tomcat**) to **TOMCAT/bin**

*Linux*

```
$ cp cas-distribution/src/main/resources/tomcat/setenv.sh  
<TOMCAT_INSTALLATION_DIR>/bin/
```

*Windows*

```
copy cas-distribution\src\main\resources\tomcat\setenv.bat  
<TOMCAT_INSTALLATION_DIR>\bin\
```

- b. Copy server.xml (**cas-distribution/src/main/resources/tomcat/conf**) to <TOMCAT\_INSTALLATION\_DIR>/conf

*Linux*

```
$ cp cas-distribution/src/main/resources/tomcat/conf/server.xml  
<TOMCAT_INSTALLATION_DIR>/conf/
```

*Windows*

```
$ cp cas-distribution\src\main\resources\tomcat\conf\server.xml  
<TOMCAT_INSTALLATION_DIR>\conf\
```

- c. The above files point to <TOMCAT\_INSTALLATION\_DIR>/certs/keystore.jks as the default keystore location to use. This file does not come with Tomcat and needs to be created or the files copied above (setenv.sh and server.xml) need to be modified to point to the correct keystore.

```
mkdir <TOMCAT_INSTALLATION_DIR>/certs
```

Copy `casKeystore.jks` from the DDF installation directory into `<TOMCAT_INSTALLATION_DIR>/certs/`. This will allow CAS to use a "cas" private key and to trust anything signed by "server", "ca", or "ca-root". Linux Expand source Windows Expand source

#### 4. Start Tomcat.

```
$ cd <TOMCAT_INSTALLATION_DIR>/bin/  
$ ./startup.sh
```

**WARNING** Make sure to run `startup.bat` instead of `startup.sh` if Windows is running on a Window machine. If `setenv.sh` was not converted to a `.bat` above, `startup.bat` will not function correctly.

If the Tomcat log has can exception like the following, or if you cannot access cas via port 8443 after completing the steps below:

```
SEVERE: Failed to initialize end point associated with ProtocolHandler ["http-apr-8443"]  
java.lang.Exception: Connector attribute SSLCertificateFile must be defined when using  
SSL with APR
```

uncomment the following in `server.xml`:

```
<Listener className="org.apache.catalina.security.SecurityListener" />
```

then comment out:

```
<Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
```

#### 5. Deploy the DDF CAS WAR to Tomcat.

- Obtain the CAS WAR by building it from `cas-distribution`.
- Copy it into the `webapps` folder on Tomcat:

```
$ cp cas-distribution/target/cas.war <TOMCAT_INSTALLATION_DIR>/webapps/
```

CAS should now be running on the Tomcat server. To verify it started without issues, check the Tomcat

log and look for lines similar to the following:

```
Apr 25, 2013 10:55:39 AM org.apache.catalina.startup.HostConfig deployWAR
INFO: Deploying web application archive /apache-tomcat-7.0.39/webapps/cas.war
2013-04-25 10:55:42,831 INFO [org.jasig.cas.services.DefaultServicesManagerImpl] -
<Loaded 1 services.>
2013-04-25 10:55:43,540 INFO [org.jasig.cas.util.AutowiringSchedulerFactoryBean] -
<Starting Quartz Scheduler now>
```

CAS will try to authenticate first with X.509 (using the keystore provided as the truststore) and failover to LDAP username/password. The DDF distribution of CAS is configured to use the embedded DDF instance running on localhost. Configuring the LDAP location may be performed by modifying the bottom of the `cas.properties` file located in `TOMCAT/webapps/cas/WEB-INF/` after the web application is deployed.

### Configure an Existing CAS Installation

If upgrading an existing CAS installation or using the standard CAS web application, refer to the Configure CAS for LDAP page or the Configure CAS for X509 User Certificates page for directions on specific configurations that need to be performed.

#### WARNING

As part of setting up the server, it is critical to make sure that Tomcat trusts the DDF server certificate and that DDF trusts the certificate from Tomcat. If this is not done correctly, CAS and/or DDF will throw certificate warnings in their logs and will not allow access.

### Configure CAS for DDF

When configuring CAS to integrate with DDF, there are two main configurations that need to be modified. By default, DDF uses 'server' as the hostname for the local DDF instance and 'cas' as the hostname for the CAS server.

#### CAS Client

The CAS client bundle contains CAS client code that can be used by other bundles when validating and retrieving tickets from CAS. This bundle is extensively used when performing authentication.

When setting up DDF, the 'Server Name' and 'Proxy Callback URL' must be set to the hostname of the local DDF instance.

The 'CAS Server URL' configuration should point to the hostname of the CAS server and should match the SSL certificate that it is using.

#### CAS Token Validator

The 'CAS Server URL' configuration should point to the hostname of the CAS server and should match the SSL certificate that it is using.

## **Additional Configuration**

Information on each of the CAS-specific bundles that come with DDF, as well as their configurations, can be found on the Security CAS application page.

## **Example Workflow**

The following is a sample workflow that shows how CAS integrates within the DDF WSS Implementation.

1. User points browser to DDF Query Page.
2. CAS servlet filters are invoked during request.
3. Assuming a user is not already signed in, the user is redirected to CAS login page.
  - a. For X.509 authentication, CAS will try to obtain a certificate from the browser. Most browsers will prompt the user to select a valid certificate to use.
  - b. For username/password authentication, CAS will display a login page.
4. After successful sign-in, the user is redirected back to DDF Query page.
5. DDF Query Page obtains the Service Ticket sent from CAS, gets a Proxy Granting Ticket (PGT), and uses that to create a Proxy Ticket for the STS.
6. The user fills in search phrase and selects **Search**.
7. The Security API uses the incoming CAS proxy ticket to create a RequestSecurityToken call to the STS.
8. The STS validates the proxy ticket to CAS and creates SAML assertion.
9. The Security API returns a Subject class that contains the SAML assertion.
10. The Query Page creates a new QueryRequest and adds the Subject into the properties map.

From step 10 forward, the message is completely decoupled from CAS and will proceed through the framework properly using the SAML assertion that was created in step 8.

### **3.3.5. Configuring CAS for LDAP**

#### **Install and Configure LDAP**

DDF comes with an embedded LDAP instance that can be used for testing. During internal testing this LDAP was used extensively.

More information on configuring the LDAP and a list of users and attributes can be found at the Embedded LDAP Configuration page.

## Add cas-server-support-ldap-3.3.1\_1.jar to CAS

Copy `thirdparty/cas-server-support-ldap-3.3.1/target/cas-server-support-x509-3.3.1_1.jar` to `<OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/webapps/cas/WEB-INF/lib/cas-server-support-ldap-3.3.1_1.jar`.

## Add spring-ldap-1.2.1\_1.jar to CAS

Copy `thirdparty/spring-ldap-1.2.1/target/spring-ldap-1.2.1_1.jar` to `<OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/webapps/cas/WEB-INF/lib/spring-ldap-1.2.1_1.jar`.

## Modify developerConfigContext.xml

1. In `<OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/webapps/cas/WEB-INF/deployerConfigContext.xml`, add the `FastBindLdapAuthenticationHandler` bean definition to the `<list>` in the property stanza with name `authenticationHandlers` of the bean stanza with id `authenticationManager`:

*deployerConfigContext.xml*

```
<bean id="authenticationManager" class="org.jasig.cas.authentication.AuthenticationManagerImpl">

    <!-- other property definitions -->

    <property name="authenticationHandlers">
        <list>
            <bean class="org.jasig.cas.adaptors.ldap.FastBindLdapAuthenticationHandler"
>
                <property name="filter" value="uid=%u,ou=users,dc=example,dc=com" />
                <property name="contextSource" ref="contextSource" />
            </bean>

            <!-- other bean definitions -->

        </list>
    </property>
</bean>
```

2. In `<OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/webapps/cas/WEB-INF/deployerConfigContext.xml`, remove the bean stanza with class `ozone3.cas.adaptors.UserPropertiesFileAuthenticationHandler` from the `<list>` of the property stanza with name `authenticationHandlers`.
3. In `<OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/webapps/cas/WEB-INF/deployerConfigContext.xml`, add the `contextSource` bean stanza to the beans stanza:

## *deployerConfigContext.xml*

```
<bean id="contextSource" class="org.jasig.cas.adaptors.ldap.util.AuthenticatedLdapContextSource">
    <property name="urls">
        <list>
            <value>ldap://localhost:1389</value>
        </list>
    </property>
    <property name="userDn" value="uid=admin,ou=system"/>
    <property name="password" value="secret"/>
</bean>
```

## Configure Ozone

Ozone is also set up to work in LDAP. This section is a reference when Ozone is being used in conjunction with CAS. The following settings were used for internal testing and should only be used as a reference.

### 1. Modify OWFsecurityContext.xml

- In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/OWFsecurityContext.xml, change the sec:x509 stanza to the following:

#### *OWFsecurityContext.xml*

```
<sec:x509 subject-principal-regex="CN=(.*?)," user-service-ref="ldapUserService" />
```

- In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/OWFsecurityContext.xml, remove the following import:

#### *OWFsecurityContext.xml*

```
<import resource="ozone-security-beans/UserServiceBeans.xml" />
```

- In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/OWFsecurityContext.xml, add the following import:

#### *OWFsecurityContext.xml*

```
<import resource="ozone-security-beans/LdapBeans.xml" />
```

### 2. Modify LdapBeans.xml

- In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/ozone-security-beans/LdapBeans.xml, change the bean stanza with id **contextSource** to the following:

*LdapBeans.xml*

```
<bean id="contextSource" class="org.springframework.security.ldap.DefaultSpringSecurityContextSource">
    <!-- The URL of the ldap server, along with the base path that all other ldap
        path will be relative to -->
    <constructor-arg value="ldap://localhost:1389/dc=example,dc=com"/>
</bean>
```

- b. In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/ozone-security-beans/LdapBeans.xml, change the bean stanza with id **authoritiesPopulator** to the following:

*LdapBeans.xml*

```
<bean id="authoritiesPopulator" class="org.springframework.security.ldap.userdetails.DefaultLdapAuthoritiesPopulator">
    <constructor-arg ref="contextSource"/>
    <!-- search base for determining what roles a user has -->
    <constructor-arg value="ou=roles"/>
</bean>
```

- c. In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/ozone-security-beans/LdapBeans.xml, change the bean stanza with id **ldapUserSearch** to the following:

*LdapBeans.xml*

```
<bean id="ldapUserSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
    <!-- search base for finding User records -->
    <constructor-arg value="ou=users" />
    <constructor-arg value="(uid={0})" /> <!-- filter applied to entities under the
        search base in order to find a given user.
                                            this default searches for an entity
        with a matching uid -->
    <constructor-arg ref="contextSource" />
</bean>
```

- d. In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/ozone-security-beans/LdapBeans.xml, change the bean stanza with id **userDetailsMapper** to the following:

### LdapBeans.xml

```
<bean id="userDetailsMapper" class="ozone.securitysample.authentication.ldap.OWFUserDetailsContextMapper">
    <constructor-arg ref="contextSource" />
    <!-- search base for finding OWF group membership -->
    <constructor-arg value="ou=groups" />
    <constructor-arg value="(member={0})" /> <!-- filter that matches only groups
that have the given username listed
                                         as a "member" attribute -->
</bean>
```

### 3. Modify OWFCASBeans.xml

- In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/lib/ozone-security-beans/OWFCasBeans.xml, change the bean stanza with id `casAuthenticationProvider` to the following:

#### OWFCasBeans.xml

```
<bean id="casAuthenticationProvider" class="org.springframework.security.cas.authentication.CasAuthenticationProvider">
    <property name="userDetailsService" ref="ldapUserService" />
    <property name="serviceProperties" ref="serviceProperties" />
    <property name="ticketValidator" ref="ticketValidator" />
    <property name="key" value="an_id_for_this_auth_provider_only" />
</bean>
```

### 3.3.6. Configuring CAS for X509 User Certificates

The follow settings were tested with CAS version 3.3.1. If any issues occur while configuring for newer versions, check the External Links section at the bottom of this page for the CAS documentation, which explains setting up certification authentication.

#### Add the `cas-server-support-x509-3.3.1.jar` to CAS

Copy `thirdparty/cas-server-support-x509-3.3.1/target/cas-server-support-x509-3.3.1.jar` to `apache-tomcat-<VERSION>/webapps/cas/WEB-INF/lib/cas-server-support-x509-3.3.1.jar`.

#### Configure Web Flow

- In `apache-tomcat-<VERSION>/webapps/cas/WEB-INF/login-workflow.xml`, make the following modifications:

- Remove the XML comments around the action-state stanza with id `startAuthenticate`.

### *startAuthenticate*

```
<action-state id="startAuthenticate">
  <action bean="x509Check" />
  <transition on="success" to="sendTicketGrantingTicket" />
  <transition on="error" to="viewLoginForm" />
</action-state>
```

- b. Modify the decision-state stanza with id `renewRequestCheck` as follows.

### *renewRequestCheck*

```
<decision-state id="renewRequestCheck">
  <if test="{externalContext.requestParameterMap['renew'] != '' &&
  externalContext.requestParameterMap['renew'] != null}" then="startAuthenticate"
  else="generateServiceTicket" />
</decision-state>
```

- c. Modify the decision-state stanza with id `gatewayRequestCheck` as follows.

### *gatewayRequestCheck*

```
<decision-state id="gatewayRequestCheck">
  <if test="{externalContext.requestParameterMap['gateway'] != '' &&
  externalContext.requestParameterMap['gateway'] != null && flowScope.service != null}" then="redirect" else="startAuthenticate" />
</decision-state>
```

2. In `apache-tomcat-<VERSION>/webapps/cas/WEB-INF/cas-servlet.xml` make the following modifications:

- a. Define the `x509Check` bean.

### *x509Check*

```
<bean
  id="x509Check"
  p:centralAuthenticationService-ref="centralAuthenticationService"
  class="org.jasig.cas.adaptors.x509.web.flow.X509CertificateCredentialsNonInteractiveAction" >
  <property name="centralAuthenticationService" ref="centralAuthenticationService" />
</bean>
```

## Configure the Authentication Handler

In `apache-tomcat-<VERSION>/webapps/cas/WEB-INF/deployerConfigContext.xml`, make the following

modifications:

1. In the **list** stanza of the property stanza with name **authenticationHandlers** of the bean stanza with id **authenticationManager**, add the **X509CredentialAuthenticationHandler** bean definition.

#### *X509CredentialAuthenticationHandler*

```
<bean id="authenticationManager"
      class="org.jasig.cas.authentication.AuthenticationManagerImpl">

    <!-- Other property definitions -->

    <property name="authenticationHandlers">
      <list>

        <!-- Other bean definitions -->

        <bean
          class="org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthenticationHandler">
          <property name="trustedIssuerDnPattern" value=".*" />
          <!--
              <property name="maxPathLength" value="3" />
              <property name="checkKeyUsage" value="true" />
              <property name="requireKeyUsage" value="true" />
          -->
        </bean>
      </list>
    </property>
  </bean>
```

## Configure the Credentials to the Principal Resolver

In `apache-tomcat-<VERSION>/webapps/cas/WEB-INF/deployerConfigContext.xml`, make the following modifications:

1. In the list stanza of the property stanza with name **credentialsToPrincipalResolver** of the bean stanza with id **AuthenticationManager**, add the **X509CertificateCredentialsToIdentifierPrincipalResolver** bean definition. The pattern in the value attribute on the property stanza can be modified to suit your needs. The following is a simple example that uses the first CN field in the DN as the Principal.

## X509CertificateCredentialsToIdentifierPrincipalResolver

```
<bean id="authenticationManager"
  class="org.jasig.cas.authentication.AuthenticationManagerImpl">
<property name="credentialsToPrincipalResolvers">
  <list>

    <!-- Other bean definitions -->

    <bean
      class="org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCred
      entialsToIdentifierPrincipalResolver">
      <property name="identifier" value="$OU $CN" />
    </bean>
  </list>
</property>
<!-- Other property definitions -->

</bean>
```

2. In addition to the **PrincipalResolver** mentioned above, CAS comes with other resolvers that can return different representations of the user identifier. This list was obtained from the official CAS Documentation site linked at the bottom of this page.

Resolver Class	Identifier Output
X509CertificateCredentialsToDistinguishedNamePrincipa lResolver	Retrieve the complete distinguished name and use that as the identifier.
X509CertificateCredentialsToIdentifierPrincipa lResolver	Transform some subset of the identifier into the ID for the principal.
X509CertificateCredentialsToSerialNumberPrinci palResolver	Use the unique serial number of the certificate.
X509CertificateCredentialsToSerialNumberAndIss uerDNPrincipa lResolver	Create a most-likely globally unique reference to this certificate as a DN-like entry, using the CA name and the unique serial number of the certificate for that CA.

Different resolvers should be used depending on the use-case for the server. When performance external attribute lookup (e.g., attribute lookup via DIAS) it is necessary to have CAS return the full DN as the identifier and the class **X509CertificateCredentialsToDistinguishedNamePrincipalResolver** should be used. When using a local LDAP, however, the **X509CertificateCredentialsToIdentifierPrincipalResolver** class can be used to only return the username that maps directly to the LDAP username.

## Default Certificates

To verify certificate authentication with the default CAS files you must make sure that the included testUser and testAdmin certificates are installed into your web browser. This has only been tested to work with Firefox. These certificates were provided in the Ozone Widget Framework and can be used in development environments.

- The sample certificate for testUser1 is <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/certs/testUser1.p12
  - password: password
- The sample certificate for testAdmin1 is <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/certs/testAdmin1.p12
  - password: password

## External Links

For more information on CAS configuration options and what each setting means, refer to their [documentation page](#).

### 3.3.7. Certificate Management

DDF uses certificates in two ways:

- i. Transmitting and receive encrypted messages.
- ii. Performing authentication of an incoming user request.

This page details general management operations of using certificates in DDF.

## Default Certificates

DDF comes with a default keystore that contains certificates.

The keystore is used for different services and the certificate contained within it is aliased to **localhost**.

<b>Host</b>
<b>Keystore</b>
<b>Truststore</b>
<b>Configuration Location</b>
<b>Usage</b>
<b>localhost</b>
<b>serverKeystore.jks</b>

<b>Host</b>
serverTruststore.jks
etc/org.ops4j.pax.web.cfg
etc/system.properties
etc/ws-security/server/encryption.properties
etc/ws-security/server/signature.properties
etc/ws-security/issuer/encryption.properties
etc/ws-security/issuer/signature.properties
etc/system.properties
Used to secure (SSL) all of the endpoints for DDF, to perform outgoing SSL requests, and sign STS SAML assertions. This also includes the Admin Console and any other web service that is hosted by DDF.

## Configuring a Java Keystore for Secure Communications

**NOTE** The following information was sourced from <https://www.racf.bnl.gov/terapaths/software/the-terapaths-api/example-java-client/java-client/setting-up-keystores-with-jetty-and-keytool>.

### Creating a Client Keystore

**WARNING** This walk-through details how to use a PKCS12 store. This is the most popular format used when exporting from a web browser.

- i. Obtain a personal ECA cert (client certificate)
  1. To do this, open Internet Explorer > Tools > Options.
  2. Select the Content tab.
  3. Click Certificates
  4. Select the Personal tab.
  5. Select the certificate needed to export. There should be the one without a "Friendly Name" and it is not the "Encryption Cert".
  6. Click Export.
  7. Follow Certificate Export Wizard.
  8. When a prompt requests to export the private key, select Yes.
- ii. Add a cert to a new java keystore, replacing cert with the name of the PKCS12 Keystore needed to

convert, and replace MY\_KEYSTORE.jks with the desired name of the Java Keystore:

```
keytool -importkeystore -srckeystore [MY_FILE.p12] -srcstoretype pkcs12  
-srcalias [ALIAS_SRC] -destkeystore [MY_KEYSTORE.jks]  
-deststoretype jks -deststorepass [PASSWORD_JKS] -destalias [ALIAS_DEST]
```

1. The command prompts for two passwords:

1. Input keystore passphrase is the passphrase that is used to protect cert.p12
2. Output keystore passphrase is the passphrase that is set for the new java keystore serverKeystore.jks
2. It is recommended that the private key password be the same as the keystore password due to limitations in java.

1. Run the following command to determine the alias name of the added current entry. It is listed after "Alias Name:"

```
keytool -list -v -keystore serverKeystore.jks
```

a. Clone the existing key using the java keytool executable, filling in <CurrentAlias>, <NewAlias>, serverKeystore.jks, and password with the correct names.

```
keytool -keyclone -alias "<CurrentAlias>" -dest "<NewAlias>" -keystore serverKeystore.jks  
-storepass password
```

- a. When prompted for a password, use the same password used when the keystore was created.
- b. Delete the original alias

```
keytool -delete -alias "<CurrentAlias>" -keystore serverKeystore.jks -storepass password
```

**NOTE**

After the keystore is successfully created, delete the jetty files used to perform the import.

## Creating a Truststore

**WARNING** This walk-through details how to import a .cer certificate

- i. Import the certificate into a java keystore as a trusted ca certificate

```
keytool -import -trustcacerts -alias "Trusted Cert" -file trustcert.cer -keystore serverTruststore.jks
```

1. Enter in a keystore password when prompted.

### **Adding a certificate to an existing Keystore**

- i. Import the certificate into a java keystore as a certificate.

```
keytool -importcert -file newcert.cer -keystore serverKeystore.jks -alias "New Alias"
```

1. Enter in the keystore password if prompted.

### **Updating Truststore and Keystore information**

To limit dependencies between individual applications, within the DDF there are several locations where truststore and keystore location is kept. Making a change to security info may mean changing this information in multiple places.

## **Locations**

Resources for changing truststore values can be found in the DDF Certificate Management documentation.

### **Certificate Configuration Management**

Configuration management includes configuring DDF to use existing certificates and defining configuration options for the system. This includes configuration certificate revocation and keystores.

### **Certificate Revocation Configuration**

#### **Enabling Revocation**

**NOTE**

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

- i. Place the CRL in <DDF.home>/etc/keystores.
- ii. Uncomment the following line in <DDF.home>/etc/ws-security/server/encryption.properties, <DDF\_HOME>/etc/ws-security/server/encryption.properties, <ddf.home>/etc/ws-security/issuer/encryption.properties, <DDF\_HOME>/etc/ws-security/server/signature.properties, and <DDF\_HOME>/etc/ws-security/issuer/signature.properties and replace the filename with the CRL file used in previous step.

```
#org.apache.ws.security.crypto.merlin.x509crl.file=etc/keystores/crlTokenIssuerValid.pem
```

Uncommenting 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=PKI|GUEST` will disable basic authentication, and require a certificate for the search UI. If a certificate is not in the CRL, it will be allowed through, otherwise it will get a 401 error. Not providing a cert will pass it to the guest handler and the user will be granted 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 cert 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 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 a new 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 is taking place**

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.apac  
he.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.validate.SignatureTrustValidator.verifyTrustInCert(SignatureTrustV  
alidator.java:213)[161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.validate.SignatureTrustValidator.validate(SignatureTrustValidator.  
java:72)[161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.validate.SamlAssertionValidator.verifySignedAssertion(SamlAssertio  
nValidator.java:121)[161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.validate.SamlAssertionValidator.validate(SamlAssertionValidator.ja  
va:100)[161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.processor.SAMLTokenProcessor.handleSAMLToken(SAMLTokenProcessor.ja  
va:188)[161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.processor.SAMLTokenProcessor.handleToken(SAMLTokenProcessor.java:7  
8)[161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.ws.security.WSSecurityEngine.processSecurityHeader(WSSecurityEngine.java:396)[  
161:org.apache.ws.security.wss4j:1.6.9]  
    at  
org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor.handleMessage(WSS4JInInterceptor.java  
:274)[164:org.apache.cxf.cxf-rt-ws-security:2.7.3]  
    at  
org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor.handleMessage(WSS4JInInterceptor.java  
:93)[164:org.apache.cxf.cxf-rt-ws-security:2.7.3]  
    at  
org.apache.cxf.phase.PhaseInterceptorChain.doIntercept(PhaseInterceptorChain.java:271)[12  
3:org.apache.cxf.cxf-api:2.7.3]  
    at  
org.apache.cxf.transport.ChainInitiationObserver.onMessage(ChainInitiationObserver.java:1  
21)[123:org.apache.cxf.cxf-api:2.7.3]  
    at  
org.apache.cxf.transport.http.AbstractHTTPDestination.invoke(AbstractHTTPDestination.java  
:239)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]  
    at  
org.apache.cxf.transport.servlet.ServletController.invokeDestination(ServletController.ja  
va:218)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]  
    at
```

```
org.apache.cxf.transport.servlet.ServletController.invoke(ServletController.java:198)[130
:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.ServletController.invoke(ServletController.java:137)[130
:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.CXFNonSpringServlet.invoke(CXFNonSpringServlet.java:158)
[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.AbstractHTTPServlet.handleRequest(AbstractHTTPServlet.ja
va:243)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.AbstractHTTPServlet.doPost(AbstractHTTPServlet.java:163)
[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
javax.servlet.http.HttpServlet.service(HttpServlet.java:713)[52:org.apache.geronimo.specs
.geronimo-servlet_2.5_spec:1.1.2]
    at
org.apache.cxf.transport.servlet.AbstractHTTPServlet.service(AbstractHTTPServlet.java:219
)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:547)[63:org.eclipse.je
ty.servlet:7.5.4.v20111024]
    at
org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:480)[63:org.eclipse
.jetty.servlet:7.5.4.v20111024]
    at
org.ops4j.pax.web.service.jetty.internal.HttpServiceServletHandler.doHandle(HttpServiceSe
rvletHandler.java:70)[73:org.ops4j.pax.web.pax-web-jetty:1.0.11]
    at
org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:119)[61:org.ecli
pse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:520)[62:org.eclips
e.jetty.security:7.5.4.v20111024]
    at
org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:227)[61:org.
eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:941)[61:org.
eclipse.jetty.server:7.5.4.v20111024]
    at
org.ops4j.pax.web.service.jetty.internal.HttpServiceContext.doHandle(HttpServiceContext.j
ava:117)[73:org.ops4j.pax.web.pax-web-jetty:1.0.11]
    at
org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:409)[63:org.eclipse.
jetty.servlet:7.5.4.v20111024]
    at
```

```
org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:186)[61:org.eclip
se.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:875)[61:org.eclip
se.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:117)[61:org.ecli
pse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.handler.HandlerCollection.handle(HandlerCollection.java:149)[61:
org.eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:110)[61:org.ec
lipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.Server.handle(Server.java:349)[61:org.eclipse.jetty.server:7.5.4
.v20111024]
    at
org.eclipse.jetty.server.HttpConnection.handleRequest(HttpConnection.java:441)[61:org.ecl
ipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.HttpConnection$RequestHandler.content(HttpConnection.java:936)[6
1:org.eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.http.HttpParser.parseNext(HttpParser.java:893)[57:org.eclipse.jetty.htt
p:7.5.4.v20111024]
    at
org.eclipse.jetty.http.HttpParser.parseAvailable(HttpParser.java:218)[57:org.eclipse.jett
y.http:7.5.4.v20111024]
    at
org.eclipse.jetty.server.BlockingHttpConnection.handle(BlockingHttpConnection.java:50)[61
:org.eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.bio.SocketConnector$ConnectorEndPoint.run(SocketConnector.java:2
45)[61:org.eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.server.ssl.SslSocketConnector$SslConnectorEndPoint.run(SslSocketConnect
or.java:663)[61:org.eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:598)[55:org.e
clipse.jetty.util:7.5.4.v20111024]
    at
org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java:533)[55:org.ec
lipse.jetty.util:7.5.4.v20111024]
    at java.lang.Thread.run(Thread.java:662)[:1.6.0_33]
Caused by: java.security.cert.CertPathValidatorException: Certificate has been revoked,
reason: unspecified
    at
```

```
sun.security.provider.certpath.PKIXMasterCertPathValidator.validate(PKIXMasterCertPathValidator.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(PKIXCertPathValidator.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.apache.ws.security.wss4j:1.6.9]
... 45 more
```

## Certificate File Management

File management includes creating and configuring the files that contain the certificates. In DDF, these files are generally Java Keystores (**jks**) and Certificate Revocation Lists (**crl**). This includes commands and tools that can be used to perform these operations.

The following tools are used:

- openssl
  - Windows users can use: [openssl](#) for windows.
- The standard Java [keytool](#) certificate management utility.
- [Portecle](#) can be used for **keytool** operations if a GUI is preferred over a command line interface.

### General Certificates

#### Create a CA Key and Certificate

The following steps demonstrate creating a root CA to sign certificates.

1. Create a key pair.

```
$> openssl genrsa -aes128 -out root-ca.key 1024
```

2. Use the key to sign the CA certificate.

```
$> openssl req -new -x509 -days 3650 -key root-ca.key -out root-ca.crt
```

#### Using the CA to Sign Certificates

The following steps demonstrate signing a certificate for the **tokenissuer** user by a CA.

1. Generate a private key and a Certificate Signing Request (CSR).

```
$> openssl req -newkey rsa:1024 -keyout tokenissuer.key -out tokenissuer.req
```

2. Sign the certificate by the CA.

```
$> openssl ca -out tokenissuer.crt -infiles tokenissuer.req
```

### Java Keystore (JKS)

#### Create a New Keystore/Truststore with an Existing Certificate and Private Key

1. Using the private key, certificate, and CA certificate, create a new keystore containing the data from the new files.

```
cat client.crt >> client.key  
openssl pkcs12 -export -in client.key -out client.p12  
keytool -importkeystore -srckeystore client.p12 -destkeystore serverKeystore.jks  
-srcstoretype pkcs12 -alias 1  
keytool -changealias -alias 1 -destalias client -keystore serverKeystore.jks  
keytool -importcert -file ca.crt -keystore serverKeystore.jks -alias "ca"  
keytool -importcert -file ca-root.crt -keystore serverKeystore.jks -alias "ca-root"
```

2. Create the truststore using only the CA certificate. Based on the concept of CA signing, the CA should be the only entry needed in the truststore.

```
keytool -import -trustcacerts -alias "ca" -file ca.crt -keystore truststore.jks  
keytool -import -trustcacerts -alias "ca-root" -file ca-root.crt -keystore  
truststore.jks
```

3. Create a PEM file using the certificate, as some applications require that format.

```
openssl x509 -in client.crt -out client.der -outform DER  
openssl x509 -in client.der -inform DER -out client.pem -outform PEM
```

### Import into a Java Keystore (JKS)

The following steps demonstrate importing a PKCS12 keystore generated by openssl into a Java keystore (JKS).

1. Put the private key and the certificate into one file.

```
$> cat tokenissuer.crt >> tokenissuer.key
```

2. Put the private key and the certificate in a PKCS12 keystore.

```
$> openssl pkcs12 -export -in tokenissuer.key -out tokenissuer.p12
```

3. Import the PKCS12 keystore into a JKS.

```
$> keytool -importkeystore -srckeystore tokenissuer.p12 -destkeystore stsKeystore.jks  
-srcstoretype pkcs12 -alias 1
```

#### 4. Change the alias.

```
$> keytool -changealias -alias 1 -destalias tokenissuer
```

### Certificate Revocation List (CRL)

#### Creating a Certificate Revocation List (CRL)

1. Using the CA create in the above steps, create a CRL in which the `tokenissuer` s certificate is valid.

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

#### 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 SSL for Sources

#### Updating Key Store / Trust Store for all sources

By default, all newly created sources use the following locations:

Trust Store: `$INSTALL_LOCATION/etc/keystores/serverTruststore.jks`

Key Store: `$INSTALL_LOCATION/etc/keystores/serverKeystore.jks`

These files come included with the distribution and are loaded with self-signed certificates for the `localhost` hostname. Certificates can be added and removed from these java keystores by using the keytool application that comes with java.

#### Updating Key Store / Trust Store via the Admin Console

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

**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.

**NOTE***Reference*

- [Oracle Keytool Documentation](#)

**Enabling SSL/TLS for Services****NOTE**

SSL/TLS is enabled out of the box for DDF.

**IMPORTANT**

Do not use the Admin Console to SSL enable the DDF services. While the Admin Console offers this configuration option, it has proven to be unreliable and may crash the system.

Edit the provided configuration file `<DDF_INSTALL_DIR>/etc/org.ops4j.pax.web.cfg` with the settings for the desired configuration.

*Table 5. Pax Web Configuration Settings*

Property	Sample Value	Description
<code>org.osgi.service.http.enabled</code>	<code>false</code>	Set this to false to disable HTTP without SSL
<code>org.osgi.service.http.secure.enabled</code>	<code>true</code>	Set this to true to SSL enable the DDF services
<code>org.osgi.service.http.port.secure</code>	<code>8993</code>	Set this to the HTTPS port number.  (Verify this port does not conflict with any other secure ports being used in the network. For example, JBoss and other application servers use port 8443 by default)
<code>org.ops4j.pax.web.ssl.key.store.type</code>	<code>jks</code>	Set this to the type of keystore (most likely jks)
<code>org.ops4j.pax.web.ssl.key.store</code>	<code>/opt/ddf/keystore.jks</code>	Set this to the fully-qualified path to the SSL keystore file
<code>org.ops4j.pax.web.ssl.key.password</code>	<code>password1</code>	Set this to the password for the user's private key
<code>org.ops4j.pax.web.ssl.password</code>	<code>password2</code>	Set this to the password for overall keystore integrity checking

*Example .cfg file:*

```
#####
# HTTP settings
#####

# Disable HTTP
org.osgi.service.http.enabled=false

# HTTP port number
org.osgi.service.http.port=8181


#####
# HTTPS settings
#####

# Enable HTTPS
org.osgi.service.http.secure.enabled=true

# HTTPS port number
# (Verify this port does not conflict with any other secure ports being used in the
# network. For example, JBoss and other application servers use port 8443 by default)

org.osgi.service.http.port.secure=8993

# Fully-qualified path to the SSL keystore
org.ops4j.pax.web.ssl.keystore=/opt/ddf/keystore.jks

# SSL Keystore Type
org.ops4j.pax.web.ssl.keystore.type=jks

# Keystore Integrity Password
org.ops4j.pax.web.ssl.password=abc123

# Keystore Password
org.ops4j.pax.web.ssl.keypassword=abc123
```

All .cfg files follow a strict formatting structure in that every entry is a **key=value** pair. There should be no whitespace before the key, around the equals sign (=), or after the value. Otherwise, the key or value may be misinterpreted.

**WARNING**

Also, take care if .cfg files originated on an operating system other than the operating system DDF is currently running on. Hidden characters, e.g., ^M, can be added during the file transfer between the operating systems. This often occurs when a DDF zip install file from a Unix operating system is transferred to a Windows operating system and installed.

**NOTE** Optional: Disable HTTP for the DDF services and only use HTTPS by setting the `org.osgi.service.http.enabled` property to `false`. After this, all DDF clients must use SSL/TLS for HTTP communication.

*Reference*

**NOTE** Configuring a Java Keystore for Secure Communications

Additional Pax-Web SSL configuration info: [SSL Configuration](#)

### 3.3.8. Updating System Users

By default, all system users are located in the `<DDF_INSTALL_DIR>/etc/users.properties` and `<DDF_INSTALL_DIR>/etc/users.attributes` files. The default users included in these two files are "admin" and "localhost". The `users.properties` file contains username, password, and role information; while the `users.attributes` file is used to mix in additional attributes. The `users.properties` file must also contain the user corresponding to the fully qualified domain name (FQDN) of the system where DDF is running. This FQDN user represents this host system internally when making decisions about what operations the system is capable of performing. For example, when performing a DDF Catalog Ingest, the system's attributes will be checked against any security attributes present on the metocard, prior to ingest, to determine whether or not the system should be allowed to ingest that metocard.

Additionally, the `users.attributes` file can contain user entries in a regex format. This allows an administrator to mix in attributes for external systems that match a particular regex pattern. The FQDN user within the `users.attributes` file should be filled out with attributes sufficient to allow the system to ingest the expected data. The `users.attributes` file uses a JSON format as shown below:

```
{  
    "admin" : {  
        "test" : "testValue",  
        "test1" : [ "testing1", "testing2", "testing3" ]  
    },  
    "localhost" : {  
  
    },  
    ".*host.*" : {  
        "reg" : "ex"  
    }  
}
```

For this example, the "admin" user will end up with two additional claims of "test" and "test1" with values of "testValue" and [ "testing1", "testing2", "testing3" ] respectively. Also, any host matching the regex ".host." would end up with the claim "reg" with the single value of "ex". The "localhost" user would have no additional attributes mixed in.

**WARNING**

It is possible for a regex in `users.attributes` to match users as well as a system, so verify that the regex pattern's scope will not be too great when using this feature.

**WARNING**

If your data will contain security markings, and these markings are being parsed out into the METACARD.security attribute via a PolicyPlugin, then the FQDN user **MUST** be updated with attributes that would grant the privileges to ingest that data. Failure to update the FQDN user with sufficient attributes will result in an error being returned for any ingest request.

### 3.3.9. Encryption Service

#### 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:encrypted command used to encrypt the plain text `myPasswordToEncrypt`. The output is the encrypted value.

```
ddf@local>security:encrypt myPasswordToEncrypt  
ddf@local>bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

### 3.3.10. Catalog Operation Authorization

All DDF Catalog operations can be restricted to users with certain attributes.

#### Configuring Operation Authorization

1. Open the Admin Console at <https://localhost:8993/admin>
2. Click the DDF Catalog application tile
3. Click the **Configuration** tab
4. Click on the **Catalog Policy Plugin** configuration.
5. Add any required attributes for each operation type

Only users with the attributes listed on the **Catalog Policy Plugin** configuration will be allowed to access those operations. By default, all operations only require a role of "guest" which every user has mixed into their attributes by way of the **Guest Claims Handler**. If the default Guest user role is changed to something other than "guest" this configuration will need to be updated to allow users to continue using DDF Catalog operations.

### 3.3.11. Catalog Filtering

Filtering is performed in 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 PDP ([feature:install security-pdp-authz](#)). 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.

## Filter Policies

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. Open the Admin Console at <https://localhost:8993/admin>
2. Click the DDF Security application tile
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 PolicyPlugins for catalog filtering exist currently: **Metocard Attribute Security Policy Plugin** and **XML Attribute Security Policy Plugin**. These PolicyPlugin implementations allow an administrator to easily add filtering capabilities to some standard Metocard types for all DDF Catalog operations. These plugins will place policy information on the Metocard itself that allows the FilterPlugin to restrict unauthorized users from viewing content they are not allowed to view.

The **XML Attribute Security Policy Plugin** will parse 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 computer the intersection of values for each attribute defined.

The **Metocard Attribute Security Policy Plugin** will pull 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.

To configure these policy plugins: . Open the Admin Console at <https://localhost:8993/admin> . Click the DDF Catalog application tile . Click the **Configuration** tab . Click on either the **Metocard Attribute Security Policy Plugin** or **XML Attribute Security Policy Plugin** configuration.

## 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.

## Attributes

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

### 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.

### 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.

## Example Requests and Responses

The following items are a sample request, response, and the corresponding policy. The XACML processing engine reads the policy and outputs a response.

### Policy

This is the sample policy that was used for the following sample request and responses. The policy was made to handle the following actions: `filter`, `query`, and `LocalSiteName`. The filter action is used to compare subject's `SUBJECT_ACCESS` attributes to a metocard's `RESOURCE_ACCESS` attributes. The `query` and `LocalSiteName` actions differ, as they are used to perform service authorization. For a query, the user must be associated with the country code ATA (Antarctica), and a `LocalSiteName` action can be performed by anyone.

## Policy

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="xpath-target-single-req" RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides" Version="1.0">
    <PolicyDefaults>
        <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
    </PolicyDefaults>
    <Target>
        <AnyOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
>filter</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </Match>
            </AllOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
>query</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </Match>
            </AllOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
>LocalSiteName</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </Match>
            </AllOf>
            <AnyOf>
        </Target>
        <Rule Effect="Permit" RuleId="permit-filter">
            <Target>
                <AnyOf>
                    <AllOf>
                        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                            <AttributeValue DataType="
```

```

http://www.w3.org/2001/XMLSchema#string">filter</AttributeValue>
    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </Match>
    </AllOf>
</AnyOf>
</Target>
<Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-subset">
        <AttributeDesignator AttributeId="RESOURCE_ACCESS" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            <AttributeDesignator AttributeId="SUBJECT_ACCESS" Category=
"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </Apply>
    </Condition>
</Rule>
<Rule Effect="Permit" RuleId="permit-action">
    <Target>
        <AnyOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">ATA</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" Category=
"urn:oasis:names:tc:xacml:1.0:action:action-id" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
                </Match>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">query</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
                </Match>
            </AllOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">LocalSiteName</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=

```

```
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
    </Match>
    </AllOf>
    <AnyOf>
        <Target>
            <Rule>
                <Rule Effect="Deny" RuleId="deny-read"/>
            </Rule>
        </Target>
    </AnyOf>
</Policy>
```

## Metocard Authorization

### Subject Permitted

All of the resource's RESOURCE\_ACCESS attributes were matched with the Subject's SUBJECT\_ACCESS attributes.

## Request

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="false" CombinedDecision="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">filter</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">users</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" IncludeInResult=
>false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">testuser1</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" IncludeInResult="false"
">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">B</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/fedata/CountryOfCitizenship">
```

```
IncludeInResult="false">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
ATA</AttributeValue>
    </Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

### *Response*

```
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

### **Subject Denied**

The resource had an additional **RESOURCE\_ACCESS** attribute 'C' that the subject did not have.

## Request

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="false" CombinedDecision="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">filter</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">users</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" IncludeInResult="false"
">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">testuser1</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">B</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/fedata/CountryOfCitizenship">
```

```

IncludeInResult="false">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
ATA</AttributeValue>
    </Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
    <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
B</AttributeValue>
        </Attribute>
    <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
C</AttributeValue>
        </Attribute>
    </Attributes>
</Request>

```

### *Response*

```

<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>

```

## 3.3.12. 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, create the following two files within the <INSTALLATION\_HOME> directory:

- `security/ddf-metocard-attribute-ruleset.cfg`
- `security/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 6. 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
<code>security:expand</code>	Runs the expansion service on the provided data returning the expanded value.
<code>security:expansions</code>	Dumps the ruleset for each active expansion service.

## 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]
```

### 3.3.13. Configure 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	serverKeystore.jks	Keystore used for SSL/TLS connections.

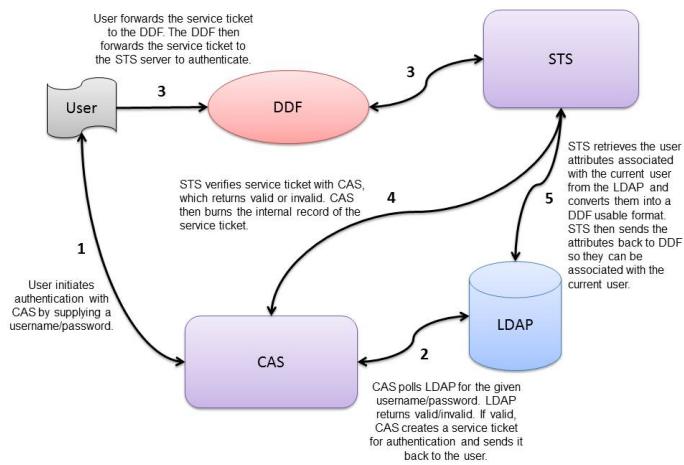


Figure 1. Login Authentication Scheme

### 3.3.14. Authentication Components

It is assumed that the three authentication components identified below are installed on three separate servers. Therefore, it is important to keep track of the DNS hostnames used on each server for certificate authentication purposes.

## LDAP

An LDAP server can be used to maintain a list of DDF users and the attributes associated with them. The STS can use an LDAP server as an attribute store and convert those attributes to SAML claims.

1. Start the DDF distribution.
2. Run the installer at <https://localhost:8993/admin> (default username/password is **admin/admin**)
3. After the installer completes, Click the **Manage** button in the upper right hand corner of the Admin Console
4. Click the **Play** button on the **Opendj Embedded** tile to install the application.

## STS

To run an STS-only DDF installation, uninstall unused catalog components to increase performance. A list of unneeded components can be found on the STS page. The STS is installed by default in DDF.

1. Verify that the **serverKeystores.jks** file in **/etc/keystores** trusts the hostnames used in your environment (the hostnames of LDAP, and any DDF users that make use of this STS server).
2. Start the distribution.
3. Run the installer at <https://localhost:8993/admin> (if needed)
4. After the installer completes, click the DDF Security application tile
5. Click the **Features** tab
6. Install **security-sts-ldaplogin** and **security-sts-ldapclaimshandler** features by clicking the **Play** button next to each
7. Open the Admin Console as an administrator (<https://localhost:8993/admin>).
8. Click the **DDF Security** application tile
9. Click the **Configuration** tab
10. Open the **Security STS LDAP Login** configuration.
  11. Verify that the **LDAP URL**, **LDAP Bind User DN**, and **LDAP Bind User Password** fields match your LDAP server's information. The default DDF LDAP settings will match up with the default settings of the OpenDJ embedded LDAP server. If not using the embedded LDAP server, change these values to map to the location and settings of the LDAP server being used.
  12. Select the **Save changes** button if changes were made.
  13. Open the **Security STS LDAP and Roles Claims Handler** configuration.

14.Populate the same URL, user, and password fields with your LDAP server information.

15.Select the **Save Changes** button.

All of the authentication components should be running and configured at this point. The final step is to configure a DDF instance, so this authentication scheme is used.

### 3.3.15. Configuring DDF Authentication Scheme

Once everything is configured and running, hooking up an existing DDF instance to the authentication scheme is performed by setting a few configuration properties:

1. Open the **Web Context Policy Manager** configuration.

a. Under **Context Realms** add the contexts that should be protected under the ldap realm.

i. The default setting is `/=karaf`, the `karaf` realm refers to the `user.properties` user store file located in the `<DDF_HOME>/etc` directory. This can be changed to `/=ldap`, if it is desired that the entire container be protected under ldap. If the `/admin` context is changed to something other than the default (`karaf`), it will be required that you refresh the page in order to log in again, or your changes may not be saved. This includes changing the root context to something other than `karaf`, without specifically setting `/admin` to a realm. The policies for all contexts will roll up, for example: the `/admin` context policy will roll up to the karaf realm with the default configuration because `/` is higher in the context heirarchy than `/admin` and no realm is specifically set for `/admin`.

b. Under **Authentication Types**, make any desired authentication changes to contexts.

i. In order to use the SAML 2.0 Web SSO profile against a context, you must specify only the IdP authentication type

2. Open the **Security STS Client** configuration. Verify that the host/port information in the **STS Address** field points to your STS server. If you are using the default bundled STS, this information will already be correct.

The DDF should now use the SSO/STS/LDAP servers when it attempts to authenticate a user upon an attempted log in.

#### Enable SSL for Clients

In order for outbound secure connections (HTTPS) to be made from components like Federated Sources and Resource Readers configuration may need to be updated with keystores and security properties. These values are configured in the `<DDF_INSTALL_DIR>/etc/system.properties` file. The following values can be set:

Property	Sample Value	Description
javax.net.ssl.trustStore	etc/keystores/serverTruststore.jks	The java keystore that contains the trusted public certificates for Certificate Authorities (CAs) that can be used to validate SSL Connections for outbound TLS/SSL connections (e.g. HTTPS). When making outbound secure connections a handshake will be done with the remote secure server and the CA that is in the signing chain for the remote server's certificate must be present in the trust store for the secure connection to be successful.
javax.net.ssl.trustStorePassword	changeit	This is the password for the truststore listed in the above property
javax.net.ssl.keyStore	etc/keystores/serverKeystore.jks	The keystore that contains the private key for the local server that can be used for signing, encryption, and SSL/TLS.
javax.net.ssl.keyStorePassword	changeit	The password for the keystore listed above
javax.net.ssl.keyStoreType	jks	The type of keystore
https.cipherSuites	TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA	The cipher suites that are supported when making outbound HTTPS connections
https.protocols	TLSv1.1,TLSv1.2	The protocols that are supported when making outbound HTTPS connections

**NOTE**      <INSTALLATION\_HOME>  
DDF is installed in the <INSTALLATION\_HOME> directory.

### 3.3.16. Auditing

**NOTE**      The Audit Log default location is `DISTRIBUTION_HOME/data/log/security.log`

#### Configuring Audit Plugin

DDF provides an optional **Audit Plugin** that logs all catalog transactions to the `security.log`. Information captured includes user identity, query information, and resources retrieved.

The Audit plugin is not enabled by default. To enable, sign into the Admin Console.

1. Select **DDF Catalog**
2. Select **Features** tab

3. Install both [catalog-security-logging](#) and [catalog-security-auditplugin](#) features.

### 3.3.17. 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 the purposes of this example we'll sign and validate using a self signed certificate which can be generated with the keytool utility. In production a certificate issued from a trusted Certificate Authority should 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
   1303 Mon Oct  6 17:05:46 MST 2014 META-INF/SELFSIGN.RSA
      0 Wed Sep 17 17:14:06 MST 2014 META-INF/
      0 Wed Sep 17 17:14:10 MST 2014 META-INF/maven/
      0 Wed Sep 17 17:14:10 MST 2014 META-INF/maven/ddf.catalog/
      0 Wed Sep 17 17:14:10 MST 2014 META-INF/maven/ddf.catalog/catalog-app/
smk   4080 Wed Sep 17 16:54:18 MST 2014 META-INF/maven/ddf.catalog/catalog-app/pom.xml
smk   107 Wed Sep 17 17:14:06 MST 2014 META-INF/maven/ddf.catalog/catalog-
app/pom.properties
      0 Wed Sep 17 17:14:06 MST 2014 repository/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/catalog-app/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/catalog-app/2.5.1/
smk 12543 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/catalog-
app/2.5.1/catalog-app-2.5.1-features.xml
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/core/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/core/catalog-core-api/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/core/catalog-core-
api/2.5.1/
smk 188995 Wed Sep 17 16:55:28 MST 2014 repository/ddf/catalog/core/catalog-core-
api/2.5.1/catalog-core-api-2.5.1.jar
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/core/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/core/mime-core-api/
      0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/core/mime-core-api/2.5.0/
smk 4396 Wed Sep 10 12:38:24 MST 2014 repository/ddf/mime/core/mime-core-
api/2.5.0/mime-core-api-2.5.0.jar
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-core/
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-core/1.2/
smk 463945 Thu Feb 13 09:26:04 MST 2014 repository/org/apache/tika/tika-core/1.2/tika-
core-1.2.jar
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-bundle/
      0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-bundle/1.2/
smk 22360866 Thu Feb 13 09:26:54 MST 2014 repository/org/apache/tika/tika-
bundle/1.2/tika-bundle-1.2.jar
      0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/
      0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/
      0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/gt-opengis/
      0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/gt-
opengis/8.4_1/
```

smk 2335529 Thu Feb 13 09:32:42 MST 2014 repository/org/codice/thirdparty/gt-opengis/8.4\_1/gt-opengis-8.4\_1.jar  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/catalog/core/catalog-core-commons/  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/catalog/core/catalog-core-commons/2.5.1/  
smk 38441 Wed Sep 17 16:56:10 MST 2014 repository/ddf/catalog/core/catalog-core-commons/2.5.1/catalog-core-commons-2.5.1.jar  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/catalog/core/catalog-core-camelcomponent/  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/catalog/core/catalog-core-camelcomponent/2.5.1/  
smk 103672 Wed Sep 17 16:57:30 MST 2014 repository/ddf/catalog/core/catalog-core-camelcomponent/2.5.1/catalog-core-camelcomponent-2.5.1.jar  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/measure/  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/measure/measure-api/  
0 Wed Sep 17 17:14:08 MST 2014 repository/ddf/measure/measure-api/2.5.1/  
smk 609307 Wed Sep 17 16:54:52 MST 2014 repository/ddf/measure/measure-api/2.5.1/measure-api-2.5.1.jar  
0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/picocontainer/  
0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/picocontainer/1.2\_1/  
smk 10819 Thu Feb 13 09:32:42 MST 2014 repository/org/codice/thirdparty/picocontainer/1.2\_1/picocontainer-1.2\_1.jar  
0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/vecmath/  
0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/vecmath/1.3.2\_1/  
smk 90446 Thu Feb 13 09:32:42 MST 2014 repository/org/codice/thirdparty/vecmath/1.3.2\_1/vecmath-1.3.2\_1.jar  
0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/geotools-suite/  
0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/geotools-suite/8.4\_1/  
smk 25175516 Thu Feb 13 09:33:40 MST 2014 repository/org/codice/thirdparty/geotools-suite/8.4\_1/geotools-suite-8.4\_1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/org/codice/thirdparty/jts/  
0 Wed Sep 17 17:14:10 MST 2014 repository/org/codice/thirdparty/jts/1.12\_1/  
smk 663441 Thu Feb 13 09:33:44 MST 2014 repository/org/codice/thirdparty/jts/1.12\_1/jts-1.12\_1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-federationstrategy/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-federationstrategy/2.5.1/  
smk 155049 Wed Sep 17 17:01:02 MST 2014 repository/ddf/catalog/core/catalog-core-federationstrategy/2.5.1/catalog-core-federationstrategy-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/org/codice/thirdparty/lucene-core/  
0 Wed Sep 17 17:14:10 MST 2014 repository/org/codice/thirdparty/lucene-core/3.0.2\_1/

smk 1041824 Thu Feb 13 09:33:48 MST 2014 repository/org/codice/thirdparty/lucene-core/3.0.2\_1/lucene-core-3.0.2\_1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/ddf-pubsub/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/ddf-pubsub/2.5.1/  
smk 152993 Wed Sep 17 16:58:18 MST 2014 repository/ddf/catalog/core/ddf-pubsub/2.5.1/ddf-pubsub-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-eventcommands/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-eventcommands/2.5.1/  
smk 11132 Wed Sep 17 17:01:10 MST 2014 repository/ddf/catalog/core/catalog-core-eventcommands/2.5.1/catalog-core-eventcommands-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/ddf-pubsub-tracker/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/ddf-pubsub-tracker/2.5.1/  
smk 6130 Wed Sep 17 17:05:52 MST 2014 repository/ddf/catalog/core/ddf-pubsub-tracker/2.5.1/ddf-pubsub-tracker-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-urllresourcereader/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-urllresourcereader/2.5.1/  
smk 84648 Wed Sep 17 16:57:00 MST 2014 repository/ddf/catalog/core/catalog-core-urllresourcereader/2.5.1/catalog-core-urllresourcereader-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/filter-proxy/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/filter-proxy/2.5.1/  
smk 33497 Wed Sep 17 16:56:24 MST 2014 repository/ddf/catalog/core/filter-proxy/2.5.1/filter-proxy-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-commands/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-commands/2.5.1/  
smk 664977 Wed Sep 17 16:56:34 MST 2014 repository/ddf/catalog/core/catalog-core-commands/2.5.1/catalog-core-commands-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-metacardgroomerplugin/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-metacardgroomerplugin/2.5.1/  
smk 31421 Wed Sep 17 17:06:04 MST 2014 repository/ddf/catalog/core/catalog-core-metacardgroomerplugin/2.5.1/catalog-core-metacardgroomerplugin-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/metacard-type-registry/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/metacard-type-registry/2.5.1/  
smk 6349 Wed Sep 17 17:05:58 MST 2014 repository/ddf/catalog/core/metacard-type-registry/2.5.1/metacard-type-registry-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-standardframework/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-

standardframework/2.5.1/  
smk 4930895 Wed Sep 17 16:58:40 MST 2014 repository/ddf/catalog/core/catalog-core-standardframework/2.5.1/catalog-core-standardframework-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-resourcesizeplugin/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-resourcesizeplugin/2.5.1/  
smk 4889822 Wed Sep 17 17:06:42 MST 2014 repository/ddf/catalog/core/catalog-core-resourcesizeplugin/2.5.1/catalog-core-resourcesizeplugin-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/fanout-catalogframework/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/fanout-catalogframework/2.5.1/  
smk 9707692 Wed Sep 17 17:01:20 MST 2014 repository/ddf/catalog/core/fanout-catalogframework/2.5.1/fanout-catalogframework-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-metricsplugin/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-metricsplugin/2.5.1/  
smk 708240 Wed Sep 17 16:57:38 MST 2014 repository/ddf/catalog/core/catalog-core-metricsplugin/2.5.1/catalog-core-metricsplugin-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-sourcemetricsplugin/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/core/catalog-core-sourcemetricsplugin/2.5.1/  
smk 709297 Wed Sep 17 17:06:14 MST 2014 repository/ddf/catalog/core/catalog-core-sourcemetricsplugin/2.5.1/catalog-core-sourcemetricsplugin-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/schematron/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/schematron/catalog-schematron-plugin/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/schematron/catalog-schematron-plugin/2.5.1/  
smk 19034 Wed Sep 17 17:09:08 MST 2014 repository/ddf/catalog/schematron/catalog-schematron-plugin/2.5.1/catalog-schematron-plugin-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/rest/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/rest/catalog-rest-endpoint/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/rest/catalog-rest-endpoint/2.5.1/  
smk 151862 Wed Sep 17 17:12:54 MST 2014 repository/ddf/catalog/rest/catalog-rest-endpoint/2.5.1/catalog-rest-endpoint-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/opensearch/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/opensearch/catalog-opensearch-endpoint/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/opensearch/catalog-opensearch-endpoint/2.5.1/  
smk 465789 Wed Sep 17 17:12:26 MST 2014 repository/ddf/catalog/opensearch/catalog-opensearch-endpoint/2.5.1/catalog-opensearch-endpoint-2.5.1.jar

```
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-extensions-
opensearch/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-extensions-
opensearch/1.1.3/
smk 33785 Thu Feb 13 09:31:18 MST 2014 repository/org/apache/abdera/abdera-extensions-
opensearch/1.1.3/abdera-extensions-opensearch-1.1.3.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-server/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-
server/1.1.3/
smk 162766 Thu Feb 13 09:31:18 MST 2014 repository/org/apache/abdera/abdera-
server/1.1.3/abdera-server-1.1.3.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/opensearch/catalog-
opensearch-source/
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/opensearch/catalog-
opensearch-source/2.5.1/
smk 136957 Wed Sep 17 17:13:04 MST 2014 repository/ddf/catalog/opensearch/catalog-
opensearch-source/2.5.1/catalog-opensearch-source-2.5.1.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/commons-codec/
0 Wed Sep 17 17:14:10 MST 2014 repository/commons-codec/commons-codec/
0 Wed Sep 17 17:14:10 MST 2014 repository/commons-codec/commons-codec/1.4/
smk 58160 Thu Feb 13 09:33:48 MST 2014 repository/commons-codec/commons-
codec/1.4/commons-codec-1.4.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/servicemix/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/servicemix/bundles/
0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.axiom-impl/
0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.axiom-impl/1.2.12-
2/
smk 121899 Thu Feb 13 09:33:48 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.axiom-impl/1.2.12-
2/org.apache.servicemix.bundles.axiom-impl-1.2.12-2.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/ws/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/ws/commons/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/ws/commons/axiom/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/ws/commons/axiom/axiom-
api/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/ws/commons/axiom/axiom-
api/1.2.10/
smk 417361 Thu Feb 13 09:33:50 MST 2014 repository/org/apache/ws/commons/axiom/axiom-
api/1.2.10/axiom-api-1.2.10.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-core/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-core/1.1.3/
smk 160895 Thu Feb 13 09:24:52 MST 2014 repository/org/apache/abdera/abdera-
core/1.1.3/abdera-core-1.1.3.jar
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-client/
0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-
```

```
client/1.1.3/
smk 62059 Thu Feb 13 09:24:52 MST 2014 repository/org/apache/abdera/abdera-
client/1.1.3/abdera-client-1.1.3.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-i18n/
        0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-i18n/1.1.3/
smk 622568 Thu Feb 13 09:24:54 MST 2014 repository/org/apache/abdera/abdera-
i18n/1.1.3/abdera-i18n-1.1.3.jar
    0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.abdera-parser/
    0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.abdera-
parser/1.1.3_1/
smk 1379508 Thu Feb 13 09:33:54 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.abdera-
parser/1.1.3_1/org.apache.servicemix.bundles.abdera-parser-1.1.3_1.jar
    0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.dom4j/
    0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.dom4j/1.6.1_5/
smk 325676 Thu Feb 13 09:33:56 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.dom4j/1.6.1_5/org.a
apache.servicemix.bundles.dom4j-1.6.1_5.jar
    0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.jdom/
    0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.jdom/1.1.2_1/
smk 160101 Thu Feb 13 09:33:56 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.jdom/1.1.2_1/org.a
apache.servicemix.bundles.jdom-1.1.2_1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/org/codice/thirdparty/commons-
httpclient/
    0 Wed Sep 17 17:14:10 MST 2014 repository/org/codice/thirdparty/commons-
httpclient/3.1.0_1/
smk 306098 Thu Feb 13 09:33:56 MST 2014 repository/org/codice/thirdparty/commons-
httpclient/3.1.0_1/commons-httpclient-3.1.0_1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/plugin/
        0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/plugin/plugin-
federation-replication/
            0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/plugin/plugin-
federation-replication/2.5.1/
smk 8986 Wed Sep 17 17:12:02 MST 2014 repository/ddf/catalog/plugin/plugin-
federation-replication/2.5.1/plugin-federation-replication-2.5.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/
        0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-
transformer-metadata/
            0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-
transformer-metadata/2.5.1/
smk 32559 Wed Sep 17 17:09:44 MST 2014 repository/ddf/catalog/transformer/catalog-
```

transformer-metadata/2.5.1/catalog-transformer-metadata-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-transformer-thumbnail/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-transformer-thumbnail/2.5.1/  
smk 32578 Wed Sep 17 17:09:52 MST 2014 repository/ddf/catalog/transformer/catalog-transformer-thumbnail/2.5.1/catalog-transformer-thumbnail-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/service-xslt-transformer/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/service-xslt-transformer/2.5.1/  
smk 47227 Wed Sep 17 17:09:28 MST 2014 repository/ddf/catalog/transformer/service-xslt-transformer/2.5.1/service-xslt-transformer-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-transformer-resource/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-transformer-resource/2.5.1/  
smk 83019 Wed Sep 17 17:09:34 MST 2014 repository/ddf/catalog/transformer/catalog-transformer-resource/2.5.1/catalog-transformer-resource-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/tika-input-transformer/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/tika-input-transformer/2.5.1/  
smk 32522 Wed Sep 17 17:10:06 MST 2014 repository/ddf/catalog/transformer/tika-input-transformer/2.5.1/tika-input-transformer-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/geojson-metacard-transformer/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/geojson-metacard-transformer/2.5.1/  
smk 9004 Wed Sep 17 17:10:22 MST 2014 repository/ddf/catalog/transformer/geojson-metacard-transformer/2.5.1/geojson-metacard-transformer-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/geojson-queryresponse-transformer/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/geojson-queryresponse-transformer/2.5.1/  
smk 53446 Wed Sep 17 17:10:28 MST 2014 repository/ddf/catalog/transformer/geojson-queryresponse-transformer/2.5.1/geojson-queryresponse-transformer-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/geojson-input-transformer/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/geojson-input-transformer/2.5.1/  
smk 35487 Wed Sep 17 17:10:16 MST 2014 repository/ddf/catalog/transformer/geojson-input-transformer/2.5.1/geojson-input-transformer-2.5.1.jar  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/service-atom-transformer/  
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/service-atom-transformer/2.5.1/  
smk 38484 Wed Sep 17 17:10:40 MST 2014 repository/ddf/catalog/transformer/service-

```
atom-transformer/2.5.1/service-atom-transformer-2.5.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-extensions-
geo/
        0 Wed Sep 17 17:14:10 MST 2014 repository/org/apache/abdera/abdera-extensions-
geo/1.1.3/
smk 28410 Thu Feb 13 09:24:52 MST 2014 repository/org/apache/abdera/abdera-extensions-
geo/1.1.3/abdera-extensions-geo-1.1.3.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/common/
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/common/geo-formatter/
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/common/geo-
formatter/2.5.1/
smk 15970 Wed Sep 17 16:55:18 MST 2014 repository/ddf/catalog/common/geo-
formatter/2.5.1/geo-formatter-2.5.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/com/
    0 Wed Sep 17 17:14:10 MST 2014 repository/com/googlecode/
    0 Wed Sep 17 17:14:10 MST 2014 repository/com/googlecode/json-simple/
    0 Wed Sep 17 17:14:10 MST 2014 repository/com/googlecode/json-simple/json-
simple/
        0 Wed Sep 17 17:14:10 MST 2014 repository/com/googlecode/json-simple/json-
simple/1.1.1/
smk 23931 Thu Feb 13 09:24:52 MST 2014 repository/com/googlecode/json-simple/json-
simple/1.1.1/json-simple-1.1.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-
transformer-xml/
        0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/transformer/catalog-
transformer-xml/2.5.1/
smk 1954994 Wed Sep 17 17:11:02 MST 2014 repository/ddf/catalog/transformer/catalog-
transformer-xml/2.5.1/catalog-transformer-xml-2.5.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/commons-collections/
    0 Wed Sep 17 17:14:10 MST 2014 repository/commons-collections/commons-
collections/
        0 Wed Sep 17 17:14:10 MST 2014 repository/commons-collections/commons-
collections/3.2.1/
smk 575389 Thu Feb 13 09:24:34 MST 2014 repository/commons-collections/commons-
collections/3.2.1/commons-collections-3.2.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-filter/
        0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-filter/2.5.1/
smk 6492 Wed Sep 17 17:13:40 MST 2014 repository/ddf/catalog/security/catalog-
security-filter/2.5.1/catalog-security-filter-2.5.1.jar
    0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-plugin/
        0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-plugin/2.5.1/
smk 5463 Wed Sep 17 17:13:50 MST 2014 repository/ddf/catalog/security/catalog-
security-plugin/2.5.1/catalog-security-plugin-2.5.1.jar
```

```
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-logging/
0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-logging/2.5.1/
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.

## 3.4. Running DDF

Follow the below steps to start and stop DDF.

### 3.4.1. Starting DDF

#### \*NIX

Run the following script from a command shell to start the distribution and open a local console:

```
INSTALLATION_HOME/bin/ddf
```

#### Windows

Run the following script from a console window to start the distribution and open a local console:

```
INSTALLATION_HOME/bin/ddf.bat
```

### 3.4.2. Stop DDF

There are two options:

- 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_INSTALL/bin/stop
```

*Windows*

```
DDF_INSTALL/bin/stop.bat
```

#### *Shut Down*

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.

### **3.4.3. 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 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 **INSTALLATION\_HOME/bin/ddf-service**.

Change:

*INSTALLATION\_HOME/bin/ddf-service*

```
#RUN_AS_USER=
```

to:

*INSTALLATION\_HOME/bin/ddf-service*

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

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

- a. Add the setting for PermGen space under the JVM Parameters section.
- b. Update heap space to 2048.

*INSTALLATION\_HOME/etc/ddf-wrapper.conf*

```
#Add the following:
```

```
wrapper.java.additional.11=-Dderby.system.home="..\data\derby"  
wrapper.java.additional.12=-Dderby.storage.fileSyncTransactionLog=true  
wrapper.java.additional.13=-Dcom.sun.management.jmxremote  
wrapper.java.additional.14=-Dfile.encoding=UTF8  
wrapper.java.additional.15=-Dddf.home=%DDF_HOME%
```

```
#Update the following:
```

```
wrapper.java.maxmemory=2048
```

4. Set the **DDF\_HOME** property.

*INSTALLATION\_HOME/etc/ddf-wrapper.conf*

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

5. Install the wrapper startup/shutdown scripts.

## Windows

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

```
INSTALLATION_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_INSTALL/bin/ddf-service /etc/init.d/
root@localhost# chkconfig ddf-service --add
root@localhost# chkconfig ddf-service on
```

## Ubuntu

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

## Solaris

```
root@localhost# ln -s DDF_INSTALL/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_INSTALL/bin/ddf-service`

#### *Solaris DDF\_INSTALL/bin/ddf-service*

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

7. Change the word `command` to `comm`.

#### *Solaris DDF\_INSTALL/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](#).

### 3.4.4. 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!

### 3.4.5. Federation

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

### 3.4.6. 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.

#### Access 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. For more information on how to use the `client` scripts or how to remote into the shell console, see Using Remote Instances.

## Example Commands

### View Bundle Status

Call `bundle:list` on the console to view the status of the bundles loaded in the distribution.

### View Installed Features

Execute `feature:list` to view the features installed in the distribution.

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

### 3.4.7. Catalog Commands

Title	Namespace	Description
DDF::Catalog :: Core :: Commands	catalog	The Catalog Shell Commands are meant to be used with any <code>CatalogProvider</code> implementations. They provide general useful queries and functions against the Catalog API that can be used for debugging, printing, or scripting.

**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.

### Commands

<code>catalog:describe</code>	<code>catalog:dump</code>	<code>catalog:envlist</code>	<code>catalog:ingest</code>
<code>catalog:inspect</code>			
<code>catalog:latest</code>	<code>catalog:migrate</code>	<code>catalog:range</code>	<code>catalog:remove</code>
<code>catalog:removeall</code>			
<code>catalog:replicate</code>	<code>catalog:search</code>	<code>catalog:spatial</code>	<code>catalog:validate</code>

Table 7. Command Descriptions

Command	Description
<code>describe</code>	Provides a basic description of the Catalog implementation.
<code>dump</code>	Exports metacards from the local Catalog. Does not remove them. See below for date filtering options.

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

## 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.

## **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]]])

```

## 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.

Title	Namespace	Description
DDF :: Admin :: Application Service	app	The DDF Admin Application Service contains operations to work with applications.

## Listing Available System Console Commands

To get a list of commands, type in the namespace of the desired extension and press **<tab>**. For example, type in: **app**, then **<tab>**

```
ddf@local>app:
app:add      app:list      app:remove    app:start    app:status    app:stop
app:tree
```

## Command Descriptions

Command	Syntax	Description
<b>add</b>	<b>app:add appUri</b>	Adds an application with the given uri.
<b>remove</b>	<b>app:remove appName</b>	Removes an application with the given name.
<b>start</b>	<b>app:start appName</b>	Starts an application with the given name.
<b>stop</b>	<b>app:stop appName</b>	Stops an application with the given name.
<b>list</b>	<b>app:list</b>	Lists the applications that are in the system and gives their current state.

<b>Command</b>	<b>Syntax</b>	<b>Description</b>
<b>status</b>	<b>app:status appName</b>	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.
<b>tree</b>	<b>app:tree</b>	Creates a hierarchy tree of all of the applications.

## Command Usage Examples

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.

Getting 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 Failure State

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

```
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 failure state. Checking the status of that application will show what did not start correctly.

```
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-sourcemetricsplugin
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.

### 3.4.8. 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.

#### Usage

The Command Scheduler allows administrators to schedule Command Line Shell Commands to be run in a "platform-independent" method. 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

1. Navigate to the Admin Console (<https://localhost:8993/admin>).
2. Select **DDF Platform**
3. Select **Platform Command Scheduler**.
4. Type 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. Type in a positive integer for the **Interval In Seconds** field.
6. Select the **Save** button. Once the **Save** button is selected, the command is executed immediately. It's next scheduled execution begins after the amount of seconds specified in the **Interval In Seconds** field and repeats indefinitely until the container is shut down or the scheduled command is deleted.

**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

1. Navigate to the **Admin Console**.
2. Click on the **DDF 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`.
5. Find the desired configuration to modify and update either the **Command** text field or the **Interval In Seconds** field or both.
6. 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.

### 3.4.9. Subscriptions Commands

Title	NameSpace	Description
<a href="#">DDF :: Catalog :: Core :: PubSub Commands</a>	<a href="#">subscriptions</a>	The DDF PubSub shell commands provide functions to list the registered subscriptions in DDF and to delete subscriptions.

**WARNING**

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

## Commands

```
ddf@local>subscriptions:  
subscriptions:delete    subscriptions:list
```

### Command Descriptions

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

### List Available System Console Commands

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

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

System Console Command Help For details on any command type `help` then the `subscriptions` command. For example, `help subscriptions:list` displays the data in the following table.

## Example Help

```
ddf@local>help subscriptions:list
DESCRIPTION
    subscriptions:list
        Allows users to view registered subscriptions.
SYNTAX
    subscriptions:list [options] [search phrase or LDAP filter]
ARGUMENTS
    search phrase or LDAP filter
        Subscription ID to search for. Wildcard characters (*) can be used in the
        ID, e.g., my*name or *123. If an id is not provided, then
            all of the subscriptions are displayed.
OPTIONS
    filter, -f
        Allows user to specify any type of LDAP filter rather than searching on
        single subscription ID.
        You should enclose the LDAP filter in quotes since it will often have
        special characters in it.
        An example LDAP filter would be:
        (& (subscription-id=my*) (subscription-id=*169*))
        which searches for all subscriptions starting with "my" and having 169 in
        the ID, which can be thought of as part of an IP address.
        An example of the entire quote command would be:
        subscriptions:list -f ""(& (subscription-id=my*) (subscription-
        id=*169*))"
--help
    Display this help message
```

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

## **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
```

### List Subscriptions Using an LDAP Filter

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

```
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 Example

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.
```

### 3.4.10. Platform Commands

Title	Namespace	Description
DDF Platform Commands	platform	The DDF Platform Shell Commands provide generic platform management functions

**WARNING** | The Platform Commands are installed when the Platform application is installed.

#### Commands

##### Command Descriptions

```
ddf@local>platform:
platform:describe    platform:envlist
```

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

#### List Available System Console Commands

To view a list of commands, type the namespace of the desired extension and press the **Tab** key.

For example, type **platform** then press **Tab**.

#### System Console Command Help

For details on any command type `help` followed by the platform command.

For example, `help platform:envlist`

#### Example Help

```
ddf@local>help platform:envlist
DESCRIPTION
    platform:envlist

        Provides a list of environment variables

SYNTAX
    platform:envlist [options]

OPTIONS
    --help
        Display this help message
```

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.

### 3.4.11. Persistence Commands

<b>Title</b>	<b>NameSpace</b>	<b>Description</b>
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.

## Commands

```
store:delete    store:list
```

### Command Descriptions

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

### 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 *store*, then press **Tab**.

### System Console Command Help

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

### Example Help

```
ddf@local>help store:list
DESCRIPTION
    store:list

    Lists entries that are available in the persistent store.

SYNTAX
    store:list [options]

OPTIONS
    User ID, -u, --user
        User ID to search for notifications. If an id is not provided, then all
        of the notifications for all users are displayed.
    --help
        Display this help message
    Persistence Type, -t, --type
        Type of item to retrieve from the persistence store.
        Options: metocard, saved_query, notification, task, or workspace
    CQL, -c, --cql
        OGC CQL statement to query the persistence store. Not specifying returns
        all entries. More information on CQL is available at:
            http://docs.geoserver.org/stable/en/user/tutorials/cql/cql\_tutorial.html
```

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.

### 3.4.12. CQL Syntax

The CQL syntax used should follow the OGC CQL format. Examples and a description of the grammar is located at [CQL Tutorial](#).

#### 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
```

### 3.4.13. Ingesting Data

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

## File types 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.

### 3.4.14. Methods of Ingest

#### Easy (for fewer records or manual ingest)

##### Ingest command (console)

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

##### Usage

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 DDF 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, consult Directory Monitor.

##### Using 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`.

##### Medium

##### 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)
```

+ . 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`. This should display the catalog entry in XML within the browser. . Verify the catalog entry exists by executing a query via the OpenSearch endpoint. . Enter the following URL in a browser `/services/catalog/query?q=ddf`. A single result, in Atom format, should be returned.

## Verifying Ingest

1. Verify GeoJson file was stored using the Content REST endpoint.

a. 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 `<INSTALL_DIR>/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.

### **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.

## **3.4.15. 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.

### **Universal Expired Records Removal**

#### **Manual Removal**

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

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

When prompted, type yes to remove all expired records.

**TIP** For help on the removeall command, execute

```
help catalog:removeall
```

#### **Automated Removal**

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.

### 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. Many specific Catalog implementations might come with their own custom scripts for removing expired metadata such as the SQL scripts provided for the Oracle Catalog implementation.

### Automatic Catalog Backup

To backup local catalog records. A backup plugin is disabled by default for performance reasons. It can be enabled and configured in the:

[Admin Console](#)   [DDF Catalog](#)   [Configuration](#)   [Backup Post-Ingest Plugin](#).

### 3.4.16. 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 **DDF-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.

### 3.4.17. Monitoring DDF

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

#### 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_install_dir]/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
```

###### 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
```

## External Resources

Other appenders can be selected and configured.

For more detail on configuring the log file and what is logged to the console a handy reference is <http://karaf.apache.org/manual/latest-2.2.x/users-guide/logging-system.html>

### 3.4.18. Enabling HTTP Access Logging

#### Configuring

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

```

## External Resources

[Advanced Jetty Configuration Jetty Request Log Tutorial](#)

## 3.5. Troubleshooting DDF

*Table 8. General Troubleshooting*

Issue	Solution
Ingest 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. Depend 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.
Ingest millions of complicated data into Solr can cause Java heap space error.	Complicated data has spatial types and large text.
Ingest 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).	Install missing Windows libraries.  Some Windows platforms are missing libraries that are required by DDF. These libraries are provided by the <a href="#">Microsoft Visual C++ 2008 Redistributable Package x64</a> .
If using an unsupported terminal, <code>java.lang.NoClassDefFoundError: Could not initialize class org.fusesource.jansi.internal.Kernel32</code> is thrown.	

Issue	Solution
<p>CXF BusException</p> <p>The following exception is thrown:</p> <pre>org.apache.cxf.BusException: No conduit initiator</pre>	<p>Restart DDF.</p> <ul style="list-style-type: none"> <li>. Shut down DDF:</li> </ul> <pre>ddf@local&gt;shutdown</pre> <ul style="list-style-type: none"> <li>. Start up DDF:</li> </ul> <pre>./ddf</pre>
<p>Distribution Will Not Start</p> <p>DDF will not start when calling the start script defined during installation.</p>	<p>Complete the following procedure.</p> <ul style="list-style-type: none"> <li>. Verify that Java is correctly installed.</li> <li>+  <pre>java -version</pre> <ul style="list-style-type: none"> <li>. This should return something similar to:</li> <li>+  <pre>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)</pre> <li>. If running *nix, verify that bash is installed.</li> <li>+  <pre>echo \$SHELL</pre> <ul style="list-style-type: none"> <li>. This should return:</li> <li>+  <pre>/bin/bash</pre> </li></ul> </li> </li></ul> </li> </ul>
<p>Multiple <code>java.exe</code> processes running, indicating more than one DDF instance is running.</p> <p>This can be caused when another DDF is not properly shut down.</p>	<p>Perform one or all of the following recommended solutions, as necessary.</p> <ul style="list-style-type: none"> <li>* Wait for proper shutdown of DDF prior to starting a new instance.</li> <li>* Verify running <code>java.exe</code> are not DDF (e.g., kill/close if necessary).</li> <li>* Utilize automated start/stop scripts to run DDF as a service.</li> </ul>

# 4. Using DDF

Version: 2.9.0

The {branding} 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.

## 4.1. 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.

### 4.1.1. 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 <b>All Sources</b> or <b>Specific Sources</b> .
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.

## 4.1.2. Results

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

Item	Description
	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.
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).

## 4.1.3. 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.

Item	Description
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.

#### 4.1.4. 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 <a href="#">Details</a> tab, which displays more information about the record.
Title	The title of the record is displayed in white font.
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.

#### 4.1.5. 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.
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.

## 4.2. 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.

### 4.2.1. 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. Saved records (metacards) 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 metacards 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 {branding}, 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.

## 4.3. 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 (metacards). Workspaces are saved for each user and are loaded when the user logs in. Workspaces and their

contents are persisted, so they survive if {branding} is restarted. Currently, workspaces are private and cannot be viewed by other users.

#### 4.3.1. 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.

## 4.4. Notifications

The Standard Search UI receives all notifications from {branding}. 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.

## 4.5. 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.

## 4.6. 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.

## 4.7. 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: 3D (), 2D (), Columbus ())

# 5. Managing DDF Admin

Version: 2.9.0

The DDF Admin Application contains components that are responsible 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.

## 5.1. Installing DDF Admin

The DDF Admin application is installed by default with a standard installation.

## 5.2. Configuring DDF Admin

Configuration	ID	Description
Admin Console Configuration	<code>org.codice.admin.ui.configuration</code>	Customization options for Admin Console elements.

### 5.2.1. Defining Platform Settings

The platform settings can be configured by the `<INSTALL_HOME>/etc/system.properties` file.

The settings in this file can be changed at any time during the DDF lifecycle (before installation, after, or during run-time) and will take effect after a system restart.

## 5.3. Console Commands

The application service comes with various console commands that can be executed on the Command Console.

### 5.3.1. Application Service Interfaces

The Application service has multiple ways of interacting with it. These methods include operations that can be performed by integrators, administrators, and end users.

### 5.3.2. Installing and Uninstalling

The Admin App installs this service by default. It is recommended to NOT uninstall the application

service unless absolutely necessary.

### 5.3.3. Configuring

None.

## 5.4. Administrative User Interface

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>.

### 5.4.1. Modules

The Admin Console is a modular system that can be expanded with additional modules as necessary. DDF comes with the Configurations module and the Installation modules. However, new modules can be added, and each module is presented in its own tab of the Admin Console.

Modules are single components that implement the `org.codice.ddf.ui.admin.api.AdminModule` interface. Once they implement and expose themselves as a service, they are added in to the Admin Console as a new tab.

### 5.4.2. Included Modules

- Installer Module
- Configuration Module

#### Installer Module

The application installer module enables a user to install and remove applications. Each application includes a features file that provides a description of the application and a list of the dependencies required to successfully run that application. The installer reads the features file and presents the applications in a manner that allows the administrator to visualize these dependencies. As applications are selected or deselected, the corresponding dependent applications are selected or deselected as necessary.

#### Set Up the Installer Module

1. Install the module if it is not already pre-installed.

```
feature:install admin-modules-installer
```

2. Open a web browser and navigate to the Installation page.

```
http://DDF_HOST:DDF_PORT/admin
```

- Adding the `?dev=true` query string will auto generate the certificates

[http://DDF\\_HOST:DDF\\_PORT/admin/index.html?dev=true](http://DDF_HOST:DDF_PORT/admin/index.html?dev=true)

3. Log in with the default username of "admin" (no quotes) and the default password of "admin" (no quotes).
4. Select the Setup tab if not already selected.

#### Example Screenshots

The following are examples of what the Installation Steps/Pages look like:

Welcome Page

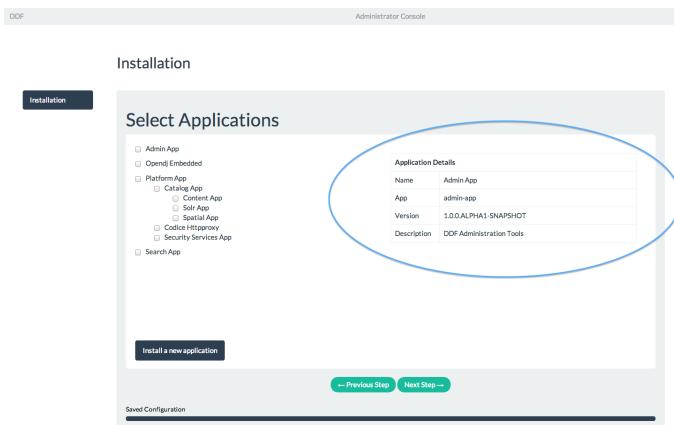
Anonymous Claims page

Installation Profile Page

**IMPORTANT** Do NOT deselect/uninstall the Platform App or the Admin App. Doing so will disable the use of this installer and the ability to install/uninstall other applications.

- Installation Profile Page
  - When a profile is selected, it will auto select applications on the Select Application Page and install them automatically.
  - If choose to customize a profile, you will be given the options to manually selected the applications on the Select Application Page.
- In the Select applications to install page, hover over each application to view additional details about the application.
- New applications can be added and existing applications can be upgraded using the Applications Module.
- When an application is selected, dependent applications will automatically be selected.
- When an application is unselected, dependent applications will automatically be unselected.

#### 5.4.3. Custom Installation



- If apps are preselected when the Select applications to install page is reached, they will be uninstalled if unselected.
- Applications can also be installed using kar deployment as stated in Application Installation.

Platform App, Admin App, and Security Services App CANNOT be selected or unselected as it is installed by default and can cause errors if removed.

#### **WARNING**

Security Services App 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.

### General Configuration Page

#### General Configuration Page (Certificates)

**NOTE** Certificate information needs to be provided if the host is changed. If the `?dev=true` query string was provided, the certificate information will be auto generated using a demo CA

### Final Page

#### Shutdown Page

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

### 5.4.4. Configuration Module

The configuration module allows administrators to change bundle and service configurations.

## Set Up the Module

1. Install the module if it is not pre-installed. [feature:install admin-modules-configuration](#)
2. Open a web browser and navigate to the Admin Console page.

[http://DDF\\_HOST:DDF\\_PORT/admin](http://DDF_HOST:DDF_PORT/admin)

1. Select the Configurations tab if not already selected.

### Configurations Tab

## 5.5. Admin Console Access Control

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.

### 5.5.1. Restricting DDF Access

1. See the documentation for your specific security infrastructure to configure users, roles, and groups.
2. On the </system/console/configMgr>, select the Web Context Policy Manager. (IMG)
  - 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 whitelist allows child contexts to be excluded from the authentication constraints of their parents.

### 5.5.2. LDAP Admin Role Configuration

The admin role will default to [system-admin](#). This can be configured to work with an external LDAP with a few minor changes.

### 5.5.3. Update the admin role in [INSTALL\\_HOME/etc/users.properties](#)

Change the value of 'system-admin' to the new admin role for any users needing the new role.

*Example `user.properties` entries:*

```
admin=admin,group,admin,manager,viewer,webconsole,system-admin  
localhost=localhost,group,admin,manager,viewer,webconsole,system-admin
```

**NOTE** A system restart is required for the changes to `users.properties` to take effect.

#### 5.5.4. Update the web context policy to point to the new admin role

1. Open DDF Security in the Admin Console
2. Select the Configuration tab and open Web Context Policy Manager
3. Update the entries under 'Required Attributes' to set the new admin role

#### Web Context Policy Manager

# 6. Managing DDF Catalog

Version: 2.9.0

## 6.1. Installing DDF Catalog

### 6.1.1. Prerequisites

Before the DDF Catalog application can be installed:

- the DDF Platform application must be installed

The DDF Catalog is pre-installed with a standard installation.

## 6.2. Configuring DDF Catalog

*Table 9. Available Configurations*

Configuration	ID	Description
[Backup Post-Ingest Plugin]	plugin.backup	Enable and configure Backup Plugin
[Catalog OpenSearch Federated Source]	OpenSearchSource	Configure settings for OpenSearch Source.
[Catalog Policy Plugin]	org.codice.ddf.catalog.security.CatalogPolicy	Configure security policy attributes for Catalog.
[Catalog Standard Framework]	ddf.catalog.CatalogFrameworkImpl	Configure settings for product retrieval through Catalog Standard Framework
[Metocard Attribute Security Policy Plugin]	org.codice.ddf.catalog.security.policy.metocard.MetocardAttributeSecurityPolicyPlugin	Configure settings for Metocard security attributes.
[Schematron Validation Services]	ddf.services.schematron.SchematronValidationService	Configure Schematron rulesets.
[XML Attribute Security Policy Plugin]	org.codice.ddf.catalog.security.policy.xml.XmlAttributeSecurityPolicyPlugin	Configure settings for locating security attributes on XML elements.
[Xml Query Transformer]	ddf.catalog.transformer.xml.XmlResponseQueueTransformer	Set threshold for running marshalling in parallel
[Catalog Duplicate Validator]	ddf.catalog.metocard.duplication.DuplicationValidator	Configure rules to check for duplicate data

Table 10. Backup Post-Ingest Plugin

Name	Property	Type	Description	Default Value	Required
Enable Backup Plugin	enableBackupPlugin	Boolean	Enable the Backup Ingest plugin which will write each result to a directory	true	No
Root backup directory path	rootBackupDir	String	Root backup directory for Metacards. A relative path is relative to ddf.home.	data/backup	Yes
Subdirectory levels	subDirLevels	Integer	Number of subdirectory levels to create. Two characters from the ID will be used to name each subdirectory level.	2	Yes

Table 11. Catalog OpenSearch Federated Source

Name	Property	Type	Description	Default Value	Required
Source Name	shortname	String		DDF-OS	Yes
OpenSearch service URL	endpointUrl	String	The OpenSearch endpoint URL or DDF's OpenSearch endpoint ( <a href="https://localhost:8993/services/catalog/query">https://localhost:8993/services/catalog/query</a> )	\${org.codice.ddf.system.protocol}\${org.codice.ddf.system.hostname}:\${org.codice.ddf.system.port}\${org.codice.ddf.system.rootContext}/catalog/query	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 OpenSearch endpoint requires basic authentication.		No

Name	Property	Type	Description	Default Value	Required
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.		No
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 ,bbox,polygo n,dtstart,dte nd,dateName ,filter,sort</code>	Yes
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 <code>src=local</code> in endpoint URL), as opposed to an enterprise search.	true	Yes
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	Yes

Table 12. Catalog Policy Plugin

Name	Property	Type	Description	Default Value	Required
Required Attributes	<code>createPermissions</code>	String	Roles/attributes required for the create operations. Example: <code>role=role1,role2</code>	<code>http://schemassoap.org/ws/2005/05/entity/claims/role=guest/</code>	Yes

Name	Property	Type	Description	Default Value	Required
Required Attributes	updatePermissions	String	Roles/attributes required for the update operation. Example: role=role1,role2	<a href="http://schemas.xmlsoap.org/ws/2005/05/entity/claims/role=guest/">http://schemas.xmlsoap.org/ws/2005/05/entity/claims/role=guest/</a>	Yes
Required Attributes	deletePermissions	String cardinality=1000	Roles/attributes required for the delete operation. Example: role=role1,role2	<a href="http://schemas.xmlsoap.org/ws/2005/05/entity/claims/role=guest/">http://schemas.xmlsoap.org/ws/2005/05/entity/claims/role=guest/</a>	Yes
Required Attributes	readPermissions	String cardinality=1000	Roles/attributes required for the read operations (query and resource). Example: role=role1,role2	<a href="http://schemas.xmlsoap.org/ws/2005/05/entity/claims/role=guest/">http://schemas.xmlsoap.org/ws/2005/05/entity/claims/role=guest/</a>	Yes

Table 13. Catalog Standard Framework

Name	Property	Type	Description	Default Value	Required
Enable Fanout Proxy	fanoutEnabled	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.	Boolean	true	No

Name	Property	Type	Description	Default Value	Required
Product Cache Directory	<code>productCacheDirectory</code>	Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Catalog Framework will attempt to create those directories. Out of the box (without configuration), the product cache directory is <code>INSTALL_DIR/data/product-cache</code> . If a relative path is provided it will be relative to the <code>INSTALL_DIR</code> . It is recommended to enter an absolute directory path such as <code>/opt/product-cache</code> in Linux or <code>C:/product-cache</code> in Windows.	String		No
Enable Product Caching	<code>cacheEnabled</code>	Check to enable caching of retrieved products.	Boolean	true	No
Max Cache Directory Size in Megabytes	<code>cacheDirMaxSizeMegabytes</code>	Configure maximum directory size for product caching. Oldest product cached will be evicted when a new product pushes the size over the specified limit. Don't set this value to the available disk space because the cache will allow a new product to get cached and then check to see if the cache exceeds the maximum allowable size. A value of 0 disables the max limit.	Long	10240	No

Name	Property	Type	Description	Default Value	Required
Delay (in seconds) between product retrieval retry attempts	delayBetweenRetryAttempts	The time to wait (in seconds) between attempting to retry retrieving a product.	Integer	10	No
Max product retrieval retry attempts	maxRetryAttempts	The maximum number of attempts to retry retrieving a product.	Integer	3	No
Product Retrieval Monitor Period	retrievalMonitorPeriod	How many seconds to wait and not receive product data before retrying to retrieve a product.	Integer	5	No
Always Cache Product	cacheWhenCancelled	Check to enable caching of retrieved products even if client cancels the download.	Boolean	false	No
Enable Notifications	notificationEnabled	Check to enable notifications.	Boolean	true	No

Table 14. Metocard Attribute Security Policy Plugin

Name	Property	Type	Description	Default Value	Required
Metocard Attributes: 2	metocardAttributes 3	String 4	Attributes within the metocard that will be collected for security information. 5	6	Nottrue

Table 15. Schematron Validation Services

Name	Property	Type	Description	Default Value	Required
Ruleset Name	id	String	Give this ruleset a name		Yes
Root Namespace	namespace	String	The root namespace of the XML	Yes	Schematron Files

Table 16. XML Attribute Security Policy Plugin

Name	Property	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

Name	Property	Type	Description	Default Value	Required
Security Attributes (intersection) :	<code>securityAttributeIntersection</code>	String 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.	Security Attributes. These attributes, if they exist on any of the XML elements listed above, will have their values extracted		false

Table 17. Xml Query Transformer

Name	Property	Type	Description	Default Value	Required
Parallel Marshalling Threshold	<code>threshold</code>	Integer	Response size threshold above which marshalling is run in parallel	50	true

Table 18. Catalog Duplicate Validator

Name	Property	Type	Description	Default Value	Required
Metocard attributes (duplicates cause a validation error)	errorOnDuplicateAttributes	String cardinality=1000	A list of metocard attributes used in the duplication check against the local catalog. If a duplicate is found, the ingest will cause a metocard validation ERROR, but the ingest will succeed.		No
Metocard attributes (duplicates cause a validation warning)	warnOnDuplicateAttributes	String cardinality=1000	A list of metocard attributes used in the duplication check against the local catalog. If a duplicate is found, the ingest will cause a metocard validation WARNING, but the ingest will succeed.	checksum	No

# 7. Managing DDF Platform

Version: 2.9.0

The DDF 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](#), and
- [Camel](#).

A Command Scheduler is also included as part of the Platform application. The Command Scheduler allows users to schedule Command Line Shell Commands to run at certain specified intervals.

## 7.1. Using DDF Platform Application

The Platform application is a core building block for any application and should be referenced for its core component versions so that developers can ensure compatibility with their own applications. The Command Scheduler included in the Platform application should be for the convenience of a "platform independent" method of running certain commands, such as backing up data or logging settings.

## 7.2. Installing and Uninstalling Platform

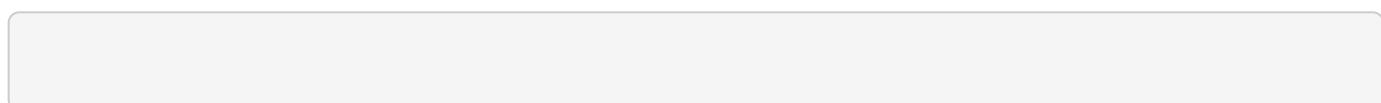
### 7.2.1. Prerequisites

None.

### 7.2.2. Installing DDF Platform

The Platform application is installed by default on a standard installation.

### 7.2.3. Configuring DDF Platform



Configuration	ID	Description
Landing Page	<a href="#">org.codice.ddf.distribution.landing-page.properties</a>	Customize elements seen on landing page.

<b>Configuration</b>	<b>ID</b>	<b>Description</b>
MIME Custom Types	<code>DDF_Custom_Mime_Type_Resolver</code>	Add or configure resolvers to identify files by type.
Metrics Reporting	<code>MetricsReporting</code>	Allocate resources for metrics collection.
Persistent Store	<code>org.codice.ddf.persistence.internal.PersistentStoreImpl</code>	Set URL for Solr server used as persistant store.
Platform Command Scheduler	<code>ddf.platform.scheduler.Command</code>	Schedule shell commands to run periodically.
Platform UI Configuration	<code>ddf.platform.ui.config</code>	

# 8. Managing DDF Security

Version: 2.9.0

The Security application provides authentication, authorization, and auditing services for the DDF. They comprise both a framework that developers and integrators can extend and a reference implementation that meets security requirements.

This section documents the installation, maintenance, and support of this application.

## 8.1. Installing DDF Security

### 8.1.1. Prerequisites

Before the DDF Security application can be installed:

- the DDF must be running
- the DDF Platform Application must be installed

### 8.1.2. Installing DDF Security

DDF Security is included with a standard installation.

### 8.1.3. Configuring DDF Security

From the Web Admin Console, the following configurations are available from a standard installation.

Configuration	Configuration ID	Description
Login Page	<code>org.codice.ddfsecurity.handler.guest.configuration</code>	Options for customizing the Login page, such as header, footer, text style
SAML NameID Policy	<code>ddfsecurity.service.SecurityManager</code>	Customize attributes to replace username of logged-in user.
STS Server Token Endpoint	<code>ddfsecurity.sts.StsStaticService</code>	Add or update addresses to use with STS service.
Security SOAP Guest Interceptor	<code>org.codice.ddfsecurity.interceptor.GuestInterceptor</code>	Settings for allowing Guest access.

Configuration	Configuration ID	Description
Security STS Address Provider	<code>ddfsecurity.sts.address.provider</code>	Configure use of alternate STS address provider
Security STS Client	<code>ddfsecurity.sts.client.configuration</code>	Settings for STS client
Security STS Guest Claims Handler	<code>ddfsecurity.sts.guestclaims</code>	Add or remove attributes to be attached to claims for guest users.
Security STS Guest Validator	<code>ddfsecurity.sts.guestvalidator</code>	Configure realms to use with Guest Validator
Security STS PKI Token Validator	<code>org.codice.ddfsecurity.validator.pki</code>	Configure realms to use with PKI Token Validator
Security STS Property File Claims Handler	<code>org.codice.ddfsecurity.sts.claims.property.PropertyFileClaimsHandler</code>	Settings for retrieving claims from properties file
Security STS Server	<code>ddfsecurity.sts</code>	Settings for STS Server
Security STS WSS	<code>ddfsecurity.sts.wss.configuration</code>	WSS-enabled version of STS
Security AuthZ Realm	<code>ddfsecurity.pdp.realm.AuthzRealm</code>	Configuration of Match-One and/or Match-All mappings in SimpleAuthz realm
Session	<code>org.codice.ddfsecurity.filter.login.Session</code>	Set session timeout
Web Context Policy Manager	<code>org.codice.ddfsecurity.policy.context.impl.PolicyManager</code>	Configure Realms, Auth types, and Attributes for different contexts.

#### 8.1.4. Applications Included in DDF Security

- Security CAS
- Security Core
- Security Encryption
- Security IdP

- Security PEP
- Security PDP
- Security STS

# 9. Managing DDF Solr

Version: 2.9.0

The Solr Catalog Provider (SCP) is an implementation of the [CatalogProvider](#) interface using [Apache Solr](#) as a data store.

## 9.1. DDF Catalog Solr External Provider

### 9.1.1. Using

The Solr Catalog Provider is used in conjunction with an Apache Solr Server data store and acts as the client for an external Solr Server. It is meant to be used only with the Standalone Solr Server ([catalog-solr-server](#)).

### 9.1.2. Installing and Uninstalling

#### Prerequisites

Before the DDF Solr Application can be installed,

- the DDF Platform Application and
- the DDF Catalog Application must be installed.

#### Installing

By default, the DDF Solr application installs the External Solr Provider.

#### Configuring

In order for the external Solr Catalog Provider to work, it must be pointed at the external Solr Server. When the [catalog-solr-external-provider](#) feature is installed, it is in an unconfigured state until the user provides an HTTP URL to the external Solr Server. The configurable properties for this SCP are accessed from the Catalog External Solr Catalog Provider configurations in the Web Console.

#### Configurable Properties

Title	Property	Type	Description	Default Value	Required
HTTP URL	<a href="#">url</a>	String	HTTP URL of the standalone, preconfigured Solr Server.	<a href="https://localhost:8993/solr">https://localhost:8993/solr</a>	Yes

Title	Property	Type	Description	Default Value	Required
Force AutoCommit	<code>forceAutoCommit</code>	Boolean / Checkbox	<p>[IMPORTANT] =====</p> <p><b>Performance Impact</b></p> <p>Only in special cases should auto-commit be forced.</p> <p>Forcing auto-commit makes the search results visible immediately.</p> <p>=====</p>	Unchecked	No

## 9.2. DDF Catalog Solr Embedded Provider

### 9.2.1. Using

The Solr Catalog Embedded Provider is an embedded, local Solr Server instance used in conjunction with an Apache Solr Server data store. It is a local instance that is a lightweight Catalog provider solution. However, it does not provide a Solr Admin GUI or a "REST-like HTTP/XML and JSON API." If that is necessary, see Standalone Solr Server.

### 9.2.2. Installing and Uninstalling

#### Prerequisites

Before the DDF Solr Application can be installed:

- the DDF Kernel must be running,
- the DDF Platform Application must be installed, and
- the DDF Catalog Application must be installed

#### Installing

1. Navigate to the DDF Solr application in the Admin console.
2. Under the **Features** tab, uninstall the `catalog-solr-external-provider` feature.
3. Install the `catalog-solr-embedded-provider`

### 9.2.3. Configuring Embedded Solr Server and Solr Catalog Provider

No configuration is necessary for the embedded Solr Server and the Solr Catalog Provider. The standard installation described above is sufficient. When the `catalog-solr-embedded-provider` feature is installed, it stores the Solr index files to `<INSTALLATION DIRECTORY>/data/solr` by default. A

user does not have to specify any parameters and the **catalog-solr-embedded-provider** feature contains all files necessary for Solr to start the server.

However, this component *can* be configured to specify the directory to use for data storage.

The configurable properties for the SCP are accessed from the **Catalog Embedded Solr Catalog Provider** configurations in the Web Console.

**TIP** The Embedded (Local) Solr Catalog Provider works on startup without any configuration because a local embedded Solr Server is automatically started and pre-configured.

## Configurable Properties

Title	Property	Type	Description	Default Value	Required
Data Directory File Path	<code>dataDirectoryPath</code>	String	<p>Specifies the directory to use for data storage. The server must be shutdown for this property to take effect. If a filepath is provided with directories that don't exist, SCP will attempt to create those directories. Out of the box (without configuration), the SCP writes to <code>&lt;INSTALLATION DIRECTORY&gt;/data/solr</code>.</p> <p>If <code>dataDirectoryPath</code> is left blank (empty string), it will default to <code>&lt;INSTALLATION DIRECTORY&gt;/data/solr</code>.</p> <p>If data directory file path is a relative string, the SCP will write the data files starting at the installation directory. For instance, if the string <code>scp/solr_data</code> is provided, the data directory will be at <code>&lt;INSTALLATION DIRECTORY&gt;/scp/solr_data</code>.</p> <p>If data directory file path is <code>/solr_data</code> in Windows, the Solr Catalog Provider will write the data files starting at the beginning of the drive, e.g., <code>C:\solr_data</code>.</p> <p>It is recommended that an absolute filepath be used to minimize confusion, e.g., <code>/opt/solr_data</code> in Linux or <code>C:\solr_data</code> in Windows. Permissions are necessary to write to the directory.</p>		No

Title	Property	Type	Description	Default Value	Required
Force Auto Commit	forceAutoCommit	Boolean / Checkbox	[IMPORTANT] ===== <b>Performance Impact</b>  Only in special cases should auto-commit be forced. Forcing auto-commit makes the search results visible immediately. =====		No

## 9.2.4. Solr Configuration Files

The Apache Solr product has Configuration files to customize behavior for the Solr Server. These files can be found at <DISTRIBUTION\_INSTALLATION\_DIRECTORY>/etc/solr. Care must be taken in editing these files because they will directly affect functionality and performance of the Solr Catalog Provider. A restart of the distribution is necessary for changes to take effect.

### Solr Configuration File Changes

#### WARNING

Solr Configuration files should not be changed in most cases. Changes to the `schema.xml` will most likely need code changes within the Solr Catalog Provider.

## 9.2.5. Move Solr Data to a New Location

If SCP has been installed for the first time, changing the Data Directory File Path property and restarting the distribution is all that is necessary because no data had been written into Solr previously. Nonetheless, if a user needs to change the location after the user has already ingested data in a previous location, complete the following procedure:

1. Change the data directory file path property within the **Catalog Embedded Solr Catalog Provider** configuration in the Admin Console to the desired future location of the Solr data files.
2. Shut down the distribution.
3. Find the future location on the drive. If the current location does not exist, create the directories.
4. Find the location of where the current Solr data files exist and copy all the directories in that location to the future the location. For instance, if the previous Solr data files existed at C:\solr\_data and it is necessary to move it to C:\solr\_data\_new, copy all directories within `C:\solr_data` into `C:\solr_data_new`. Usually this consists of copying the index and tlog directories into the new data directory.
5. Start the distribution. SCP should recognize the index files and be able to query them again.

	<b>Changes Require a Distribution Restart</b>
<b>WARNING</b>	If the Data Directory File Path property is changed, no changes will occur to the SCP until the distribution has been restarted.
<b>NOTE</b>	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.

## 9.3. Standalone Solr Server

The Standalone Solr Server gives the user an ability to run an Apache Solr instance as a Catalog data store within the distribution. The Standalone Solr Server contains a Solr Web Application Bundle and pre-configured Solr configuration files. A Solr Web Application Bundle is essentially the Apache Solr war repackaged as a bundle and configured for use within this distribution.

### 9.3.1. Using

Users can use this feature to create a data store. Users would use this style of deployment over an embedded Java Solr Server when the user wants to install a Solr Server on a separate, dedicated machine for the purpose of isolated data storage or ease of maintenance. The Standalone Solr Server can now run in its own JVM (separate from endpoints and other frameworks) and accept calls with its "REST-like HTTP/XML and JSON API."

This Standalone Solr Server is meant to be used in conjunction with the Solr Catalog Provider for External Solr. The Solr Catalog Provider acts as a client to the Solr Server.

### 9.3.2. Installing and Uninstalling

#### Prerequisites

Before the DDF Solr Application can be installed for configuration as the Standalone Solr Server, the DDF Kernel must be running.

In production environments, it is recommended that Standalone Solr Server be run in isolation on a separate machine in order to maximize the Solr Server performance and use of resources such as RAM and CPU cores. The Standalone Solr Server, as its name suggests, does not require or depend on other apps, such as the Catalog API, nor does it require their dependencies, such as Camel, CXF, etc. Therefore, it is recommended to have the Solr Server app run on a lightweight DDF distribution, such as the DDF Distribution Kernel. If clustering is necessary, the Solr Server application can run alongside the Platform application for clustering support.

### 9.3.3. Installing

By default, the features for the Standalone Solr Server and External Solr Catalog Provider are installed.

## Remove Data from Solr Core

It is possible to remove data in the Solr index of a Solr core. Replace <CORE\_NAME> in the following command with a valid Solr core to delete all data in that Solr core:

*How to delete Solr Core data with curl*

```
curl 'https://localhost:8993/solr/<CORE_NAME>/update?commit=true' -H 'Content-type: text/xml' -d '<delete><query>*:*</query></delete>'
```

Use the core selector in the Solr administration page to get a list of available Solr cores.

*Solr administration page*

```
https://localhost:8993/solr
```

### 9.3.4. Configuring

The Standalone Solr Server comes pre-configured to work with Solr Catalog External Provider implementations. For most use cases, no other configuration to the Solr Server is necessary with the standard distribution.

### 9.3.5. Known Issues

The standalone Solr Server fails to install if it has been previously uninstalled prior to the distribution being restarted.

### 9.3.6. Solr Standalone Server Meta Catalog Backup

Prior to setting up backup for the Solr Metadata catalog, it is important to plan how backup and recovery will be executed. The amount and velocity of data entering the catalog differ depending on the use of the system. As such, there will be varying plans depending on the need. It is important to get a sense of how often the data changes in the catalog in order to determine how often the data should be backed up. When something goes wrong with the system and data is corrupted, how much time is there to recover? A plan must be put in place to remove corrupted data from the catalog and replace it with backed up data in a time span that fits deadlines. Equipment must also be purchased to maintain backups, and this equipment may be co-located with local production systems or remotely located at a different site. A backup schedule will also have to be determined so that it does not affect end users interacting with the production system.

#### Back Up Data from the Solr Server Standalone Metadata Catalog

The Solr server contains a built-in backup system capable of saving full snapshot backups of the catalog data upon request. Backups are created by using a web based service. Through making a web based service call utilizing the web browser, a time-stamped backup can be generated and saved to a local drive, or location where the backup device has been mounted.

The URL for the web call contains three parameters that allow for the customization of the backup:

*command*

allows for the command 'backup' to backup the catalog.

*location*

allows for a file system location to place the backup to be specified.

*numberToKeep*

allows the user to specify how many backups should be maintained. If the number of backups exceed the "numberToKeep" value, the system will replace the oldest backup with the newest one.

An example URL would look like  
`http://127.0.0.1:8181/solr/replication?command=backup&location=d:/solr_data&numberToKeep=5.`

The IP address and port in the URL should be replaced with the IP address and port of the Solr Server. The above URL would run a backup, save the backup file in `D:/solr_data`, and it would keep up to five backup files at any time. To execute this backup, first ensure that the Solr server is running. Once the server is running, create the URL and copy it into a web browser window. Once the URL is executed, the following information is returned to the browser:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">15</int>
  </lst>
  <str name="status">OK</str>
</response>
```

If the status equals 0, there was success. Qtime shows the time it took to execute the backup (in milliseconds). Backup files are saved in directories which are given the name `snapshot` along with a timestamp. Within the directory are all of the files that contain the data from the catalog.

## Restore Data to the Solr Server Standalone Metadata Catalog

Under certain circumstances, such as when data has been corrupted, information has accidentally been deleted, or a system upgrade is occurring, the catalog must be restored. The backup files acquired from the previous section will be used to restore data into the catalog.

1. The first step in the process is to choose which data backup will be used for restoring the catalog. A most recent backup maybe the correct choice, or the last stable backup may be a better option.
2. At this point, one more backup may be executed to save the corrupted data just in case it needs to be revisited.

3. Shut down the Solr server. The catalog cannot be restored while the server is running.
4. Locate the index that contains all of the Solr data. This index is found at `DDF_INSTALL/solr/collection1/data/index`
5. All files within the index directory should be deleted.
6. Copy the files from the chosen backup directory into the index directory.
7. Restart the Solr server. The data should now be restored.

### Suggestions for Managing Backup and Recovery

Here are some helpful suggestions for setting up data backups and recoveries:

- Acquire a backup drive that is separate from the media that runs the server. Mount this drive as a directory and save backups to that location.
- Ensure that the backup media has enough space to support the number of backups that need to be saved.
- Run a scheduler program that calls the backup URL on a timed basis.
- Put indicators in place that can detect when data corruption may have occurred.
- Testing a backup before recovery is possible. A replicated "staging" Solr server instance can be stood up, and the backup can be copied to that system for testing before moving it to the "production" system.

# 10. Managing DDF Spatial

Version: 2.9.0

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

This page describes:

- which applications must be installed prior to installing this application.
- how to install the DDF Spatial Application.
- how to verify if the application was successfully installed.
- how to uninstall the application.
- how to upgrade the application.
- the optional features available in the application.
- the console commands that come with the application.

## 10.1. Prerequisites

Before the DDF Spatial Application can be installed:

- the DDF Platform Application and
- the DDF Catalog Application must be installed

## 10.2. Installing

1. Before installing a DDF application, verify that its prerequisites have been met.
2. Copy the DDF application's KAR file to the <INSTALL\_DIRECTORY>/deploy directory.

**NOTE**

These Installation steps are the same whether DDF was installed from a distribution zip or a custom installation using the DDF Kernel zip.

### 10.2.1. Verifying

1. Verify the appropriate features for the DDF application have been installed using the `features:list` command to view the KAR file's features.
2. Verify that the bundles within the installed features are in an active state.

## 10.2.2. Uninstalling

**WARNING**

It is very important to save the KAR file or the feature repository URL for the application prior to an uninstall so that the uninstall can be reverted if necessary.

If the DDF application is deployed on the DDF Kernel in a custom installation (or the application has been upgraded previously), i.e., its KAR file is in the <INSTALL\_DIRECTORY>/deploy directory, uninstall it by deleting this KAR file.

Otherwise, if the DDF application is running as part of the DDF distribution zip, it is uninstalled the first time and only the first time using the **features:removeurl** command:

*Uninstall DDF application from DDF distribution*

```
feature:repo-remove -u <Application's feature repository URL>
```

Example: feature:repo-remove -u mvn:org.codice.ddf.spatial/spatial-app/2.5.0/xml/features

The uninstall of the application can be verified by the absence of any of the DDF application's features in the **feature:list** command output.

**NOTE**

The repository URLs for installed applications can be obtained by entering:  
**feature:repo-list -u**

## 10.2.3. Reverting the Uninstall

If the uninstall of the DDF application needs to be reverted, this is accomplished by either:

- copying the application's KAR file previously in the <INSTALL\_DIRECTORY>/deploy directory, OR
- adding the application's feature repository back into DDF and installing its main feature, which typically is of the form <applicationName>-app, e.g., **catalog-app**.

*Reverting DDF application's uninstall*

```
feature:repo-add <Application's feature repository URL>
feature:install <Application's main feature>
```

*Example*

```
ddf@local>feature:repo-add mvn:ddf.catalog/catalog-app/2.9.0/xml/features
ddf@local>feature:install catalog-app
```

## 10.2.4. Upgrading

To upgrade an application, complete the following procedure.

1. Uninstall the application by following the Uninstall Applications instructions above.
2. Install the new application KAR file by copying the `admin-app-X.Y.kar` file to the `<INSTALL_DIRECTORY>/deploy` directory.
3. Start the application.
4. Complete the steps in the Verify section above to determine if the upgrade was successful.

## 10.3. Optional Features

### 10.3.1. 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. To do so, you'll need to use the `geonames:update` command which is explained in the next section.

## 10.4. Console Commands

Table 19. *GeoNames Commands*

Title	Namespace	Description
DDF :: Spatial :: Commands	<code>geonames</code>	The <code>geonames</code> commands provide the ability to interact with a local GeoNames index. The local GeoNames index is used by the <code>offline-gazetteer</code> feature, which is an optional feature available in this application and is explained above. Note that these commands are only available if the <code>offline-gazetteer</code> feature is installed.

### 10.4.1. Commands

```
geonames:update
```

### 10.4.2. Command Descriptions

Command	Description
<code>update</code>	<p>Adds new entries to an existing local GeoNames index. Entries can be manually downloaded from <a href="http://download.geonames.org/export/dump">http://download.geonames.org/export/dump</a>, where the absolute path of the file would be passed as an argument to the command (ex. /Users/john doe/Downloads/AU.zip). Currently .txt and .zip files are supported for manual entries. Entries can also be automatically downloaded from <a href="http://download.geonames.org/export/dump">http://download.geonames.org/export/dump</a> 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 <a href="http://download.geonames.org/export/dump/countryInfo.txt">http://download.geonames.org/export/dump/countryInfo.txt</a>. 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.</p> <p>The index location can be configured via the Admin UI or the Felix Web 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.</p> <p>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 UI or Felix Web 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>

# 11. Managing DDF Search UI

Version: 2.9.0

The Standard Search UI is a user interface that enables users to search a catalog and associated sites for content and metadata.

This section describes:

- Which applications must be installed prior to installing this application.
- How to install the DDF Search UI.
- How to verify if the DDF Search UI was successfully installed.
- How to uninstall the DDF Search UI.
- How to upgrade the DDF Search UI.

## 11.1. Prerequisites

Before the DDF Search UI application can be installed:

- the DDF Platform Application and
- the DDF Catalog Application must be installed.

## 11.2. Installing DDF Search UI

The Search UI application is installed by default.

## 11.3. Configuring DDF Search UI

Configure individual features within the application with the Admin Console.

### 11.3.1. Configurable Properties

#### Configurations

Configuration	ID	Description
Search UI Endpoint	<code>org.codice.ddf.ui.search.standard.endpoint</code>	Setting for cache and normalization of Search UI endpoint
Search UI Redirect	<code>org.codice.ddf.ui.searchui.filter.RedirectServlet</code>	URI to use with Search UI redirect

Configuration	ID	Description
Simple Search UI	<code>org.codice.ddf.ui.search.simple.properties</code>	Basic Display options for using Simple Search UI
Standard Search UI	<code>org.codice.ddf.ui.search.standard.properties</code>	Display options for using Standard Search UI

Table 20. Search UI Endpoint

Title	Property	Type	Description	Required
Disable Cache	<code>cacheDisabled</code>	Boolean	Disables use of cache.	no
Disable Normalization	<code>normalizationDisabled</code>	Boolean	Disables relevance and distance normalization.	no

Table 21. Search UI Redirect

Title	Property	Type	Description	Required
Redirect URI	<code>defaultUri</code>	String	Specifies the redirect URI to use when accessing the /search URI.	Yes

Table 22. Simple Search UI

Title	Property	Type	Description	Required
Header	<code>header</code>	String	Specifies the header text to be rendered on the generated Query Page	Yes
Footer	<code>footer</code>	String	Specifies the footer text to be rendered on the generated Query Page	Yes
Text Color	<code>color</code>	Specifies the Text Color of the Header and Footer. Use html css colors or \#rrggbb.	String	Yes
Background Color	<code>background</code>	String	Specifies the Background Color of the Header and Footer. Use html css colors or \#rrggbb.	Yes

Table 23. Standard Search UI

Title	Property	Type	Description	Required
Header	header	String	The header text to be rendered on the Search UI.	no
Footer	footer	String	The footer text to be rendered on the Search UI.	no
Style	style	String	The style name (background color) of the Header and Footer.	yes
Text Color	textColor	String	The text color of the Heater and Footer.	yes
Result count	resultCount	Integer	The max number of results to display.	yes
Imagery Providers	imageryProviders	String	<p>List of imagery providers to use. Valid types are:</p> <ul style="list-style-type: none"> <li>-OSM (OpenStreetMap),</li> <li>-AGM (ArcGisMap),</li> <li>-BM (BingMap),</li> <li>-WMS (WebMapService),</li> <li>-WMT (WebMapTile),</li> <li>-TMS (TileMapService),</li> <li>-GE (GoogleEarth).</li> <li>-</li> <li>-Example: TYPE={url=</li> <li>-{"type" "WMS" "url" "http://example.com" "layers" ["layer1" "layer2"] "parameters" {"FORMAT" "image/png" "VERSION" "1.1.1"} "alpha" 0.5}}</li> </ul>	no
Terrain Providers	terrainProvider	String	<p>Terrain provider to use for height data. Valid types are:</p> <ul style="list-style-type: none"> <li>-CT (CesiumTerrain),</li> <li>-AGS (ArcGisImageServer),</li> <li>-VRW (VRTheWorld).</li> <li>-</li> <li>-Example:</li> <li>-{"type" "CT" "url" "http://example.com"}</li> </ul>	no
Map Projection	projection	String	Projection of imagery providers	no
Connection timeout	timeout	Integer	The WMS connection timeout.	yes

Title	Property	Type	Description	Required
Show sign in	<code>signIn</code>	Boolean	Whether or not to authenticate users.	no
Show tasks	<code>task</code>	Boolean	Whether or not to display progress of background tasks.	no
Show Gazetteer	<code>gazetteer</code>	Boolean	Whether or not to show gazetteer for searching place names.	no
Show Uploader	<code>ingest</code>	Boolean	Whether or not to show upload menu for adding new metadata.	no
Type Name Mapping	<code>typeNameMapping</code>	String[]	The mapping of content types to displayed names.	no

### 11.3.2. Upgrading

Upgrading to a newer version of the app can be performed by the Admin Console.

## 11.4. Troubleshooting DDF Search UI

### 11.4.1. Deleted Records Are Being Displayed In The Standard Search UI's Search Results

When queries are issued by the Standard Search UI, the query results that are returned are also cached in an internal Solr database for faster retrieval when the same query may be issued in the future. As records are deleted from the catalog provider, this Solr cache is kept in sync by also deleting the same records from the cache if they exist.

Sometimes the cache may get out of sync with the catalog provider such that records that should have been deleted are not. When this occurs, users of the Standard Search UI may see stale results since these records that should have been deleted are being returned from the cache. Records in the cache can be manually deleted using the URL commands listed below from a browser. In these command URLs, `metacard_cache` is the name of the Solr query cache.

- To delete all of the records in the Solr cache:

*Deletion of all records in Solr query cache*

```
http://localhost:8181/solr/metacard_cache/update?stream.body=<delete><query>*:*</query></delete>&commit=true
```

- To delete a specific record in the Solr cache by ID (specified by the `original_id_txt` field):

*Deletion of record in Solr query cache by ID*

```
http://localhost:8181/solr/metocard_cache/update?stream.body=<delete><query>original_id_t  
xt:50ffd32b21254c8a90c15fccfb98f139</query></delete>&commit=true
```

- To delete record(s) in the Solr cache using a query on a field in the record(s) - in this example, the **title\_txt** field is being used with wildcards to search for any records with word remote in the title:

*Deletion of records in Solr query cache using search criteria*

```
http://localhost:8181/solr/metocard_cache/update?stream.body=<delete><query>title_txt:*re  
mote*</query></delete>&commit=true
```

# 12. Integrating DDF

Version: 2.9.0

This section supports integrating DDF with existing applications or frameworks.

## 12.1. Understanding Metadata and Metacards

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

## 12.2. Populating Metacards (during ingest)

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

## 12.3. Searching Metadata

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

### 12.3.1. Search Types

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

#### Contextual Search

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

#### Spatial Search

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

## Temporal Search

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

## Datatype

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

## Compound Search

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

### 12.3.2. Search Interfaces

#### DDF Search UI Application

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

#### SSH

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

## 12.4. Catalog Search Result Objects

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

- Distance: Populated when a point radius spatial search occurs. Numerical value that indicates the result's distance from the center point of the search.
- Units: Populated when a point radius spatial search occurs. Indicates the units (kilometer, mile, etc.) for the distance field.
- Relevance: Populated when a contextual search occurs. Numerical value that indicates how

relevant the text in the result is to the text originally searched for.

### 12.4.1. Search Programmatic Flow

Searching the catalog involves three basic steps:

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

#### Sort Policies

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

There are three policies available.

*Table 24. 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.

**WARNING**

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

## 12.5. Asynchronous Search & Retrieval

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

*Table 25. Important Terms for Asynchronous Search*

Capability	Description	Endpoint Integration
Asynchronous Search	Search multiple sources simultaneously	Search UI
Product caching	Allows quick retrieval of already downloaded products	DDF Catalog
Canceling Product Downloads	The ability to cancel a download in progress	DDF Catalog
Activities	Activities <ul style="list-style-type: none"><li>* <a href="#">download</a></li><li>* <a href="#">retry</a></li><li>* <a href="#">cancel</a></li><li>* <a href="#">pause</a></li><li>* <a href="#">remove</a></li><li>* <a href="#">resume</a></li></ul>	DDF Catalog, CometD endpoint
Notifications	Time-stamped messages of an action	DDF Catalog, DDF UI/CometD endpoint
Workspaces	Ability to save and manage queries and save metacards	DDF Platform, DDF UI/CometD endpoint
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.

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

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

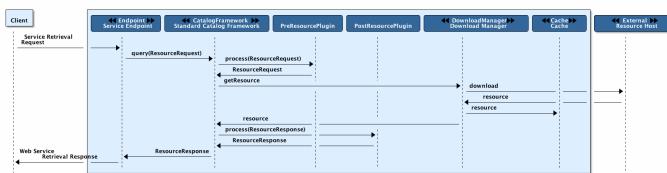
### *Product Caching*

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.

#### **NOTE**

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

Product Caching is disabled by default.



*Figure 2. Product Retrieval Request*

The Catalog Framework optionally provides caching of products, so future requests to retrieve the same product will be serviced much quicker. If caching is enabled, each time a retrieve product request is received, the Catalog Framework will look in its cache (default location: `<INSTALL_DIR>/data/productcache`) to see if the product has been cached locally. If it has, the product is retrieved from the local site and returned to the client, providing a much quicker turnaround because

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

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

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

## Product Download Status

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

## Notifications and Activities

### Notifications

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

### Activities

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

### 12.5.4. Integrating with the Asynchronous Capabilities Endpoint

The following section shows examples of integration. The channels are used for clients to subscribe,

followed by examples of request and responses. There is only one endpoint hosted at <DDF\_IP>:<DDF\_PORT\_NUMBER>/cometd

## Subscribing to Notifications

### Notifications Overview

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

### Usage

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

Notifications are currently being utilized in the Catalog application for resource retrieval. When a user initiates a resource retrieval, the channel `/ddf/notification/catalog/downloads` is opened, where notifications indicating the progress of that resource download are sent. Any client interested in receiving these progress notifications must subscribe to that channel. DDF starts downloading the resource to the client that requested it, a notification with a status of "Started" will be broadcast. If the resource download fails, a notification with a status of "Failed" will be broadcast. Or, if the resource download is being attempted again after a failure, "Retry" will be broadcast. When a notification is received, DDF Search UI displays a popup containing the contents of the notification, so a user is made aware of how their downloads are proceeding. Behind the scenes, the DDF Search UI invokes the REST endpoint to retrieve a resource. In this request, it adds the query parameter "user" with the CometD session ID or the unique User ID as the value. This allows the CometD server to know which subscriber is interested in the notification. For example, [http://DDF\\_HOST:8181/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1w1qlo79j6tscii19jszwp9s2i55](http://DDF_HOST:8181/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1w1qlo79j6tscii19jszwp9s2i55) notifications contain the following information:

Parameter Name	Description	Required by DDF Search UI
<code>application</code>	"Downloads" for resource retrieval. This is used as a "type" or category of messages.	Yes
<code>title</code>	Resource/file name for resource retrieval.	Yes
<code>message</code>	Human-readable message containing status and a more detailed message.	Yes
<code>timestamp</code>	Timestamp in milliseconds of when event occurs.	Yes
<code>user</code>	CometD Session ID or unique User ID.	Yes
<code>status</code>	Status of event.	No
<code>option</code>	Resource retrieval option.	No
<code>bytes</code>	Number of bytes transmitted.	No

## Receive Notifications

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

## Notification Events

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

### Notifications Channel

[/TODO: fix subscribe link](#) To receive all notifications, subscribe to [/ddf/notifications/\\*\\\*](#)

### Notification Message Format

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

Map Key	Description	Value Type	Possible Values	Example Values
<a href="#">application</a>	Name of the application that caused the notification to be sent	String	Any (Downloads is the only application currently implemented)	"Downloads"
<a href="#">id</a>	ID of the notification "thread" – Notifications about the same event should use the same id to allow clients to filter out notifications that may be outdated.	String	Any	"27ec3222af1144ff827a351b1962a236"

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

*Example: Notification Message*

```

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

```

## 12.5.5. Notification Operations

### Notification Operations Channel

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

*Table 26. Operation Format*

Map Key	Description	Value Type	Possible Values	Example Values	Comments
action	Type of action to request	String	Any	"remove" (Currently only used action)	If a client publishes with the <code>remove</code> action, it removes the notification with the given id from the persistence store.

Map Key	Description	Value Type	Possible Values	Example Values	Comments
<code>id</code>	ID of the notification to which the action relates	String	Any	"27ec3222af1144ff827a351b1962a236"	This is the id of the notification

*Example: Notification Operation Request*

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

## 12.5.6. Activity Events

### Activity Events Channel

To receive all activity updates, follow the instructions on the Asynchronous Search and Retrieval page and subscribe to `/ddf/activities/*\*`

### Activity Format

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

Map Key	Description	Value Type	Possible Values	Example Values
<code>category</code>	Category of the activity	Any	String	Any String
<code>"Product Retrieval"</code>	event.topics		String	
	<code>id</code>	ID that uniquely identifies the activity that sent out the update. Not required to be unique per update.	String	Any String

Map Key	Description	Value Type	Possible Values	Example Values
<code>"b72ccdf6-8ca7-4f53-a0f6-b0ad264393b0"</code>	message	User-readable message that explains the current activity status	String	Any String
<code>"Resource retrieval downloading."</code>	operations	Map of operations that can be performed on this activity	JSON Map	<p>A map of keys with populated values (that evaluate to 'true' rather than 'null' 'undefined' or 'false') These operations and their values can be used by clients to communicate back to the server by sending a message back on the same channel.</p> <p>If the value is a URL, the client should invoke the URL as a result of the user invoking the activity operation.</p>

Map Key	Description	Value Type	Possible Values	Example Values
<pre>[source,json,l inenums] ---- "operations" :{ "download" : "http://exam ple.com/prod uct" } ----</pre> <p>If the value is not a URL, the client should send a message back to the server on the same topic with the operation name.</p> <p>Note: the DDF UI will interpret several values with special icons:</p> <ul style="list-style-type: none"> <li>* <code>download</code></li> <li>* <code>retry</code></li> <li>* <code>cancel</code></li> <li>* <code>pause</code></li> <li>* <code>remove</code></li> <li>* <code>resume</code></li> </ul>	progress	Percentage value of activity completion	String	Integer between 0 - 100 followed by a %
<code>"45%"</code>	status	Enumerated value that displays the current state of the activity	String	<ul style="list-style-type: none"> <li>* <code>STARTED</code>,</li> <li>* <code>RUNNING</code>,</li> <li>* <code>FINISHED</code>,</li> <li>* <code>STOPPED</code>,</li> <li>* <code>PAUSED</code>, or</li> <li>* <code>FAILED</code></li> <li>* <code>RUNNING</code></li> </ul>

Map Key	Description	Value Type	Possible Values	Example Values
timestamp	Time that the activity update was sent	String	Positive long value (seconds since unix epoch)	1403801920875
title	User-readable title for the activity update	String	Any String	"Download Complete"
user	User who started the activity	String	Any String	"admin"
Custom Value	Additional keys can be inserted by the component sending the activity notification	Any JSON Type		

*Example: Activity update with custom 'bytes' field*

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

## 12.5.7. Activity Operations

### Channel

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

Refer to [\[Asynchronous Search and Retrieval\]](#) for instructions on how to publish to a CometD channel.

*Table 27. Activity Format*

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

*Example: Activity Operation Request Message*

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

## 12.5.8. Query Service

### Query Service Channel

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

### Query Request Format

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

Map Key	Description	Value Type	Possible Search UI Values	Example Values	Comments
<b>count</b>	Number of entries to return in the response	Number	Positive integer	250	
<b>format</b>	Format that the results should be displayed in	String	"geojson"	"geojson"	"geojson" is the recommended format to use

Map Key	Description	Value Type	Possible Search UI Values	Example Values	Comments
id		String		"4303ba5d-21af-4878-9a4c-808e80052e6c"	
src	Comma-delimited list of federated sites to search over	String	Any	list of site names. "DDF-OS,ddf.distribution"	
start	Specifies the number of the first result that should be returned	Number	Positive integer	1 10 would mean the 10th result from the query would be returned as the first one in the response.	cql

## Query Request Examples

### Enterprise Contextual

```

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

```

## *Multiple Site Temporal Absolute*

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

## *Enterprise Spatial Bounding Box*

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

## 12.5.9. Query Responses

### Query Response Channel

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

### Query Response Message Format

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

Map Key	Description	Value Type	Possible Values	Example Values
<code>id</code>	ID that corresponds to the request	String	ID value	

Map Key	Description	Value Type	Possible Values	Example Values
hits	Total number of query hits that were found by the server	Number	Integer >= 0	This contains the total amount of items that were found by the query. Depending on the 'count' in the request, not all of the results may be returned.
results	Array of metocard results	Array of Maps	GeoJson-formatted value	This format is defined by the GeoJSON Metocard Transformer.
results/meta card/actions	An array of actions that applies to each metocard, injected into each metocard	Array of Maps	Array of objects, possibly empty if no actions are available	Each Action will contain an id, title, description, and url
status	Array of status for each source queried	Array		
status.state	Specifies the state of the query	String	SUCCEEDED, FAILED, ACTIVE	status.elapsed
Time in milliseconds that it took for the source to complete the request	Number	Integer >= 0		status.hits
Number of records that were found on the source that matched the query	Number	Integer >= 0		status.id
ID of the federated source	String	Any string value		status.results

Map Key	Description	Value Type	Possible Values	Example Values
Number of results that were returned in this response from the source	Number	Integer $\geq 0$		types

### Query Response Examples

## Example Query Response

```
"data": {
  "id": "4303ba5d-21af-4878-9a4c-808e80052e6c",
  "hits": 20,
  "results": [
    {
      "metacard": {
        "geometry": {
          "coordinates": [ 174.77557, -41.28664 ],
        },
        "type": "Point"
      },
      "properties": {
        "created": "2014-07-02T18:53:59.496 +0000",
        "id": "126e16d340dd47b7ad823b70662ed8ca",
        "metacard-type": "ddf.metacard",
        "metadata": "...xml...",
        "modified": "2014-07-02T18:53:59.496+0000",
        "source-id": "ddf.distribution",
        "title": "NORMAL WORK RESUMES AT NEW ZEALAND PORTS"
      },
      "type": "Feature",
    },
    "actions": [
      {
        "id": "catalog.data.metacard.resource",
        "title": "Get resource",
        "description": "Gets the Metacard resource",
        "url":
        "http://ddf.codice.org:8181/services/catalog/sources/ddf.distribution/85227c45d9c34fe1ad3
        e725a72d6b44a?transform=resource"
      },
      {
        "id": "catalog.data.metacard.html",
        "title": "Get html",
        "description": "Gets the Metacard html",
        "url":
        "http://ddf.codice.org:8181/services/catalog/sources/ddf.distribution/85227c45d9c34fe1ad3
        e725a72d6b44a?transform=html"
      }
    ],
    ...
    ...other metacards...
  ],
  "status": [
    {
      "elapsed": 539,
    },
    "hits": 10,
    "id": "DDF-OS",
    "results": 10,
  ]
}
```

```
"state": "SUCCEEDED"
},
{
    "elapsed": 11,
"hits": 10,
"id": "ddf.distribution",
"results": 10,
"state": "SUCCEEDED"
}],
"types": {
    "ddf.metacard" : {
        "resource-uri" : "STRING",
"location" : "GEOMETRY",
"expiration" : "DATE",
"metadata-target-namespace" : "STRING",
"metadata-content-type" : "STRING",
"effective" : "DATE",
"modified" : "DATE",
"id" : "STRING",
"title" : "STRING",
"thumbnail" : "BINARY",
"created" : "DATE",
"metadata-content-type-version" : "STRING",
"resource-size" : "STRING",
"metadata" : "XML"
    },
    ...other metacard types...
}
}
}
```

# 13. Integrating DDF Admin

Version: 2.9.0

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

## 13.1. API

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

### 13.1.1. JMX Managed Bean

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

*getApplicationTree*

Creates an application hierarchy tree that shows relationships between applications.

*startApplication*

Starts an application with the given name.

*stopApplication*

Stops an application with the given name.

*addApplications*

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

### 13.1.2. Configuration Files

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

### 13.1.3. Initial Application Installation

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

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

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

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

*Examples:*

```
# Local application:  
opendj-embedded  
  
# Application installed into a local maven repository:  
opendj-embedded=mvn:org.codice.opendj.embedded/opendj-embedded-app/1.0.1-  
SNAPSHOT/xml/features  
  
# Application located on the file system:  
opendj-embedded=file:/location/to/opendj-embedded-app-1.0.1-SNAPSHOT.kar
```

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

### 13.1.4. Interface Details

The Application Service comes with several interfaces to use.

#### ApplicationService Interface

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

##### getApplications

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

##### getApplication

Returns the application that has the given name.

##### startApplication

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

##### stopApplication

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

##### addApplication

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

##### removeApplication

Removes an application that has the given URI.

### `isApplicationStarted`

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

### `getApplicationStatus`

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

### `getApplicationTree`

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

### `findFeature`

Determine which application contains a certain feature.

## **Application Interface**

### `getName`

Name of the application. Should be unique among applications.

### `getFeatures`

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

### `getBundles`

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

## **ApplicationStatus**

### `getApplication`

Sends back the application that is associated with this status.

### `getState`

Returns the application's state as defined by `ApplicationState`.

### `getErrorFeatures`

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

### `getErrorBundles`

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

## **ApplicationNode Interface**

### `getApplication`

Returns the application this node is referencing.

### `getStatus`

Returns the status for the application this node is referencing.

## getParent

Returns the parent of the application.

## getChildren

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

## 13.2. Implementation Details

### NOTE

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

### 13.2.1. 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.

### 13.2.2. 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>	

# 14. Integrating DDF Catalog

Version: 2.9.0

## 14.1. Design

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

### 14.1.1. Ensuring Compatibility

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

This section supports integration of the Catalog Application.

## 14.2. Integrating Endpoints

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

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

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

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

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

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

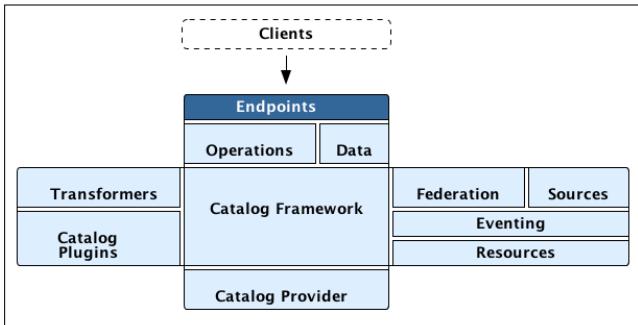


Figure 3. Endpoint Architecture

#### 14.2.1. Existing Endpoints

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

##### DDF Catalog RESTful CRUD Endpoint

The Catalog REST Endpoint allows clients to perform CRUD operations on the Catalog using REST, a simple architectural style that performs communication using HTTP. The URL exposing the REST functionality is located at `http://<HOST>:<PORT>/services/catalog`, where **HOST** is the IP address of where the distribution is installed and **PORT** is the port number on which the distribution is listening.

##### Installing and Uninstalling

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

##### Configuring

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

##### Using the REST CRUD Endpoint

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

Operation	HTTP Request	Details	Example URL
<b>create</b>	HTTP POST	HTTP request body contains the input to be ingested.	<code>http://&lt;DISTRIBUTION_HOST&gt;:&lt;DISTRIBUTION_PORT&gt;/services/catalog</code>

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

### Sources Operation Example

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

## Sources Response Example

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

Note that for all RESTful CRUD commands only one 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.)

## 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:

```
http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/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>`

## Metocard Transforms Available in DDF

DDF includes the following Metocard Transformers:

### *HTML Metocard Transformer*

transforms a Metocard into an HTML formatted document

### *XML Metocard Transformer*

transforms a Metocard into an XML formatted document

### *GeoJSON Metocard Transformer*

transforms a Metocard into GeoJSON text

### *Thumbnail Metocard Transformer*

retrieves the thumbnail bytes of a Metocard

### *Metadata Metocard Transformer*

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

### *Resource Metocard Transformer*

retrieves the resource bytes of a Metocard product

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

## HTML Metocard Transformer

### *Description*

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

### *Usage*

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
```

## Installing and Uninstalling

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

## Configuration

None

## Implementation Details

Registered Interface	Service Property	Value
<code>ddf.catalog.transform.MetocardTransformer</code>	title	View as html...
	description	Transforms query results into html
	shortname (for backwards compatibility)	html

## Known Issues

None

### XML Metocard Transformer

#### Description

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

#### Usage

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

```
https://localhost:8993/services/catalog/ac0c6917d5ee45fb3c2bf8cd2eba67?transform=xml
```

## Installation and Uninstallation

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

## Configuration

None

## Implementation Details

*Table 28. 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/&lt;AttributeType&gt;[name='&lt;AttributeName&gt;']/value</code> For instance, the value for the Metocard Attribute named “title” would be found at: <code>metocard/string[@name='title']/value</code>

Table 29. *AttributeTypes*

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

## Known Issues

None

## GeoJSON Metocard Transformer

### Description

The GeoJSON Metocard Transformer translates a Metocard into GeoJSON.

### Usage

The GeoJSON Metocard Transformer can be used programmatically by requesting a

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

## Example

*REST GET method with the GeoJSON MetacardTransformer*

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

## Installation and Uninstallation

Install the [catalog-transformer-json](#) feature using the Admin Console.

## Configuration

None

## Implementation Details

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

## Known Issues

None.

## Thumbnail Metacard Transformer

### Description

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

### Usage

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

## Sample Blueprint Reference Snippet

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

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

## Installation and Uninstallation

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

## Configuration

None

## Implementation Details

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

## Known Issues

None

## Metadata Metocard Transformer

### Description

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

### Usage

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

### Example

*REST GET method with the Metadata MetacardTransformer*

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

## Installation and Uninstallation

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

## Configuration

None

## Implementation Details

Registered Interface	Service Property	Value
<a href="#">ddf.catalog.transform.MetacardTransformer</a>	mime-type	<a href="#">text/xml</a>
	id	<a href="#">metadata</a>
	shortname (for backwards compatibility)	<a href="#">metadata</a>

## Known Issues

None.

## Resource Metacard Transformer

### Description

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

### Usage

Endpoints or other components can retrieve an instance of the Resource Metacard Transformer using its id [resource](#).

#### *Sample Blueprint Reference Snippet*

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

## Installation and Uninstallation

This transformer is installed by installing the feature associated with the transformer [catalog-](#)

[transformer-resource](#). To uninstall the transformer, you must uninstall the feature [catalog-transformer-resource](#).

## Configuration

None

## Implementation Details

Service Property	Value
id	<a href="#">resource</a>
shortname	<a href="#">resource</a>
mime-type	<a href="#">application/octet-stream</a>
title	Get Resource ...

## Known Issues

None

## [InputTransformers](#)

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

**NOTE** | [InputTransformers](#) can be added to the system at any time.

## Resources (Content)

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

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

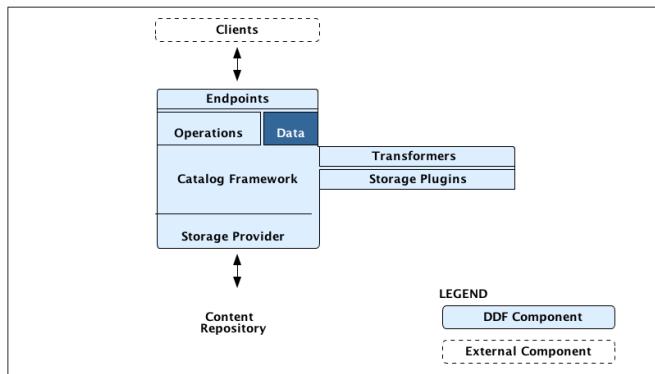


Figure 4. Content Data Component Architecture

### Content Item

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

### Storage Plugins

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

Types of Storage Plugins available out of the box:

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

### Video Thumbnail Plugin

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

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

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

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

## Directory Monitor

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

The typical execution flow of the Directory Monitor is:

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

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

## Using

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

### Sample Usage Scenarios

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

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

- The **Copy Ingested Files** option is not checked.
- As files are placed in the monitored directory <DDF\_INSTALL\_DIR>/inbox, the files are ingested into the Catalog Framework.
  - The Catalog Framework generates a GUID for the create request for this ingested file.
  - Since the Store and Process directive was configured the ingested file is passed on to the Content File System Storage Provider, which creates a sub-directory in the Content Repository using the GUID and places the ingested file into this GUID sub-directory using the file name provided in the request.
  - The Catalog Framework then invokes the Catalog Content Plugin, which looks up the Input Transformer associated with the ingested file's mime type and invokes the Catalog Framework, which inserts the metocard into the MDC. This Input Transformer creates a metocard based on the contents of the ingested file.
  - The Catalog Framework sends back a successful status to the Camel route that was monitoring the directory.
  - Camel route completes and deletes the file from the monitored directory.

#### **Scenario 2: Monitor single directory for storage with file backup**

- The Content Directory Monitor has the following configurations.
  - The **absolute** path of /usr/my/home/dir/inbox for the directory path.
  - The Processing Directive is set to store only.
  - The **Copy Ingested Files** option is checked.
- As files are placed in the monitored directory /usr/my/home/dir/inbox, the files are ingested into the Catalog Framework.
  - The Catalog Framework generates a GUID for the create request for this ingested file.
  - Since the Store directive was configured, the ingested file is passed on to the Content File System Storage Provider, which creates a sub-directory in the Content Repository using the GUID and places the ingested file into this GUID sub-directory using the file name provided in the request.
  - The Catalog Framework sends back a successful status to the Camel route that was monitoring the directory.
  - The Camel route completes and moves the file from the monitored directory to its sub-directory /usr/my/home/dir/inbox/.ingested.

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

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

### Configuring

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

## Configuring Content Directory Monitors

Managed Service Factory PID: [org.codice.ddf.catalog.content.monitor.ContentDirectoryMonitor](#)

### Configurable Properties

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

### Implementation Details

#### Imported Services

Registered Interface	Availability	Multiple
<a href="#">ddf.mime.MimeTypeToTransformerMapper</a>	required	false
<a href="#">ddf.catalog.CatalogFramework</a>	required	false
<a href="#">ddf.catalog.filter.FilterBuilder</a>	required	false

#### Exported Services

Registered Interface	Service Property	Value
<a href="#">ddf.action.ActionProvider</a>	id	catalog.data.metacard.view
<a href="#">ddf.catalog.util.DdfConfigurationWatcher</a>		

## Known

None.

### 14.2.2. OpenSearch Endpoint

The OpenSearch Endpoint provides a CDR REST Search v3.0 and CDR REST Brokered Search 1.1 compliant DDF endpoint that a client accesses to send query parameters and receive search results.

This endpoint uses the input query parameters to create an OpenSearch query. The client does not need to specify all of the query parameters, only the query parameters of interest.

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

#### Installing and Uninstalling

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

#### Configuring

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

#### Using the OpenSearch Endpoint

Once installed, the OpenSearch endpoint is accessible from [http://<DDF\\_HOST>:<DDF\\_PORT>/services/catalog/query](http://<DDF_HOST>:<DDF_PORT>/services/catalog/query).

##### Using the endpoint

###### From Code:

The OpenSearch specification defines a file format to describe an OpenSearch endpoint. This file is XML-based and is used to 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://<ddf_host>:8181/services/catalog/query?q=Predator
```

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

## Parameter List

### Main OpenSearch Standard

OS Element	HTTP Parameter	Possible Values	Comments
searchTerms	q	URL-encoded string	Complex contextual search string.
count	count	integer >= 0	Maximum # of results to retrieve default: 10
startIndex	start	integer >= 1	Index of first result to return. default: 1 This value uses a one based index for the results.
format	format	requires a transformer shortname as a string, possible values include, when available  atom  html  kml  see Included Query Response Transformers for more possible values.	default: atom

### Temporal Extension

OS Element	HTTP Parameter	Possible Values	Comments
start	dtstart	RFC-3399-defined value	yyyy-MM-dd T HH:mm:ss.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** = geometries that are not **DISJOINT** of the given geospatial parameter.

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

OS Element	HTTP Parameter	Possible Values	Comments
geometry	geometry	WKT Geometries: <code>POINT</code> , <code>POLYGON</code> , <code>MULTIPOINT</code> , <code>MULTIPOLYGON</code>	Examples:  <code>POINT(10 20)</code> where 10 is the longitude and 20 is the latitude.  <code>POLYGON ( ( 30 10, 10 20, 20 40, 40 40, 30 10 ) )</code> . 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 30. Extensions

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

Table 31. 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.

## DDF Extensions

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

## Supported Complex Contextual Query Format

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

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

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

## OpenSearch Complex Contextual Query EBNF

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

The OpenSearch Description Document is an XML file is found inside of the OpenSearch Endpoint bundle and is named [ddf-os.xml](#).

## Implementation Details

### Imported Services

Registered Interface	Availability	Multiple
<a href="#">ddf.catalog.CatalogFramework</a>	required	false
<a href="#">ddf.catalog.filter.FilterBuilder</a>	required	false

### Exported Services

Registered Interface	Service Property	Value
<a href="#">ddf.catalog.util.DdfConfigurationWatcher</a>		

### Known Issues

None

## 14.3. Developing a New Endpoint

Complete the following procedure to create an endpoint.

1. Create a Java class that implements the endpoint's business logic. Example: Creating a web service

- that external clients can invoke.
2. Add the endpoint's business logic, invoking [CatalogFramework](#) calls as needed.
  3. Import the DDF packages to the bundle's manifest for run-time (in addition to any other required packages):  
**Import-Package: ddf.catalog, ddf.catalog.\***
  4. Retrieve an instance of [CatalogFramework](#) from the OSGi registry. (Refer to the Working with OSGi - Service Registry section for examples.)

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

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

**No implementation of an interface is required**

**TIP** 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.

#### 14.3.1. Common Endpoint Business Logic

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

## 14.4. DDF Data Migration

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

## 14.4.1. Set Up

Set up DDF as instructed in Starting DDF section.

## 14.4.2. Move Metadata from One Catalog Provider to Another

### Export Metadata Out of Catalog Provider

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

*ddf@local*

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

### Ingest Exported Metadata into Catalog Provider

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

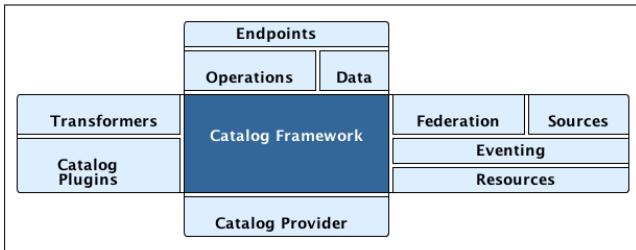
*ddf@local*

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

### Translate Metadata from One Format to Another

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

## 14.5. Integrating Catalog Framework



## 14.5.1. Catalog Framework

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

### Example Catalog Frameworks

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

- Catalog Framework
- Catalog Fanout Framework

### Catalog Framework Sequence Diagrams

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

#### Ingest

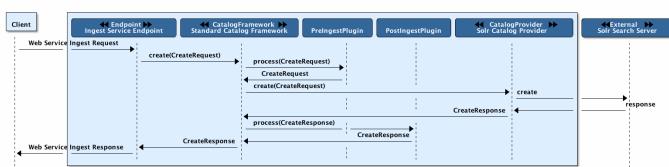


Figure 5. Ingest Request

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

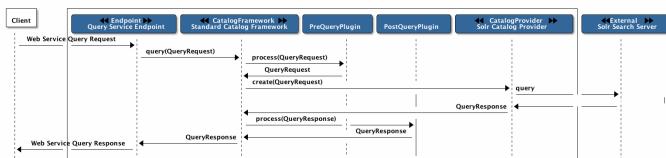


Figure 6. Ingest Request

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

### 14.5.2. Product Retrieval

The Query Service Endpoint, the Catalog Framework, and the `CatalogProvider` are key components for

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

## Product Caching

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

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

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

## Product Download Status

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

## 14.6. DDF Schematron

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.

### 14.6.1. Configuring

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

## 14.7. Adding New Attribute Types, Metocard Types, and Validators Using JSON Files

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

### 14.7.1. Definition File Format

Metocard Types, Attribute Types, and global Attribute Validators can be defined within a definition file. A definition file follows the JSON format as specified in ECMA-404. All definition files must be valid JSON in order to be parsed. There are three main types that can be defined in a definition file.

- Metocard Types
- Attribute Types
- Global Attribute Validators

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

### 14.7.2. Deploying

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

### 14.7.3. Attribute Type Definition

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

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

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

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

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

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

- `SHORT_TYPE`

An example with multiple attributes defined:

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

#### 14.7.4. Metocard Type Definition

To define Metocard Types, your 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  
        },  
        "point-of-contact": {  
          "required": true  
        }  
      }  
    }  
  ]  
}
```

**NOTE**

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

You can define multiple metocard types in a single file:

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

#### 14.7.5. Validator Definition

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

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

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

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

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

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

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

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

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

Examples:

```
{  
  "validators": {  
    "title": [  
      {  
        "validator": "size",  
        "arguments": ["1", "50"]  
      },  
      {  
        "validator": "pattern",  
        "arguments": ["\\D+"]  
      }  
    ],  
    "created": [  
      {  
        "validator": "pastdate",  
        "arguments": []  
      }  
    ],  
    "expiration": [  
      {  
        "validator": "futurereadate",  
        "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"]  
      }  
    ]  
  }  
}
```

# 15. Integrating DDF Platform

Version: 2.9.0

This section supports integration of this application with external frameworks.

## 15.1. System Properties

The Platform Global Settings are the system-wide configuration settings used throughout DDF to specify the information about the machine hosting DDF.

### 15.1.1. Configuration

The configurable properties for the platform-wide configuration are accessed from **Configuration Platform Global Configuration** in the Web Console.

### 15.1.2. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Protocol	<code>protocol</code>	String	Default protocol that should be used to connect to this machine.	http	yes
Host	<code>host</code>	String	The host name or IP address of the machine that DDF is running on. Do not enter localhost.		yes
Port	<code>port</code>	String	The port that DDF is running on.		yes
Site Name	<code>id</code>	String	The site name for this DDF instance.	<code>ddf.distribution</code>	yes
Version	<code>version</code>	String	The version of DDF that is running. This value should not be changed from the factory default.	DDF 2.3.0	yes
Organization	<code>organization</code>	String	The organization responsible for this installation of DDF	Codice Foundation	yes

## 15.2. Platform UI Settings

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

## 15.2.1. Configuration

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

## 15.2.2. Configurable Properties

Title	Property	Type	Description	Default Value	Required
Enable System Usage Message	systemUsageEnabled	Boolean	Turns on a system usage message, which is shown when the Search Application is opened		yes
System Usage Message Title	systemUsageTitle	String	A title for the system usage Message when the application is opened		yes
System Usage Message	systemUsageMessage	String	A system usage message to be displayed to the user each time the user opens the application		yes
Show System Usage Message once per session	systemUsageOncePerSession	Boolean	With this selected, the system usage message will be shown once for each browser session. Uncheck this to have the usage message appear every time the search window is opened or refreshed.	true	yes

Title	Property	Type	Description	Default Value	Required
Header	header	String	Specifies the header text to be rendered on all pages.		yes
Footer	footer	String	Specifies the footer text to be rendered on all pages.		yes
Text Color	color	String	Specifies the Text Color of the Header and Footer. Use html css colors or #rrggb.		yes
Background Color	background	String	Specifies the Background Color of the Header and Footer. Use html css colors or #rrggb.		yes

## 15.3. Landing Page

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

### 15.3.1. Configuration

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

### 15.3.2. Configurable Properties

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

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

## 15.4. DDF Mime Framework

### 15.4.1. Mime Type Mapper

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

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

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

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

## 15.4.2. Included Mime Type Mappers

### DDF Mime Type Mapper

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

#### Installing and Uninstalling

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

#### Configuring

There is no configuration needed for this feature.

## 15.4.3. Mime Type Resolver

A [MimeTypeResolver](#) is a DDF service that can map a file extension to its corresponding mime type and, conversely, can map a mime type to its file extension.

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

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

### Tika Mime Type Resolver

The [TikaMimeTypeResolver](#) is a [MimeTypeResolver](#) that is implemented using the Apache Tika open source product.

Using the Apache Tika content analysis toolkit, the [TikaMimeTypeResolver](#) provides support for resolving over 1300 mime types.

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

#### Using

The [TikaMimeTypeResolver](#) provides the bulk of the default mime type support for DDF.

## Installing and Uninstalling

The [TikaMimeTypeResolver](#) is bundled as the `mime-tika-resolver` feature in the `mime-tika-app` application.

This feature is installed by default.

## Configuring

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

## Implementation Details

### Exported Services

Registered Interface	Service Property	Value
<code>ddf.mime.MimeTypeResolver</code>		<code>tika-mimetypes.xml</code>

## Custom Mime Type Resolver

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

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

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

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

As a [MimeTypeResolver](#), the Custom Mime Type Resolver will provide methods to map the file extension to the corresponding mime type, and vice versa.

## Using

The Custom Mime Type Resolver is used when mime types need to be added that are not supported by DDF out of the box. By adding custom mime type resolvers to DDF, new content with that mime type can be processed by DDF.

## Installing and Uninstalling

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

core feature.

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

## Configuring

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

## Managed Service Factory PID

- `DDF_Custom_Mime_Type_Resolver`

*Table 32. Configurable Properties*

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

## Implementation Details

*Table 33. Imported Services*

Registered Interface	Availability	Multiple
<code>ddf.catalog.transform.InputTransformer</code>	optional	true
<code>ddf.catalog.transform.QueryResponseTransformer</code>	optional	true
<code>ddf.mime.MimeTypeResolver</code>	optional	true

*Table 34. Exported Services*

Registered Interface	Service Property	Value
<code>ddf.mime.MimeTypeToTransformerMapper</code>		
<code>ddf.mime.MimeTypeMapper</code>		

## 15.5. Metrics Collection

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

**WARNING**

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

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

The metrics currently being collected by DDF are:

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Exceptions	<code>ddf.metrics.catalog:name=Exceptions</code>	Count	A count of the total number of exceptions, of all types, thrown across all catalog queries executed.
Catalog Exceptions Federation	<code>ddf.metrics.catalog:name=Exceptions.Federation</code>	Count	A count of the total number of Federation exceptions thrown across all catalog queries executed.
Catalog Exceptions Source Unavailable	<code>ddf.metrics.catalog:name=Exceptions.SourceUnavailable</code>	Count	A count of 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	<code>ddf.metrics.catalog:name=Exceptions.UnsupportedQuery</code>	Count	A count of the 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.
Catalog Ingest Created	<code>ddf.metrics.catalog:name=Ingest.Created</code>	Count	A count of the number of catalog entries created in the Metadata Catalog.
Catalog Ingest Deleted	<code>ddf.metrics.catalog:name=Ingest.Deleted</code>	Count	A count of the number of catalog entries updated in the Metadata Catalog.

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Ingest Updated	<code>ddf.metrics.catalog:name=Ingest.Updated</code>	Count	A count of the number of catalog entries deleted from the Metadata Catalog.
Catalog Queries	<code>ddf.metrics.catalog:name=Queries</code>	Count	A count of the number of queries attempted.
Catalog Queries Comparison	<code>ddf.metrics.catalog:name=Queries.Comparison</code>	Count	A count of 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	<code>ddf.metrics.catalog:name=Queries.Federated</code>	Count	A count of the number of federated queries attempted.
Catalog Queries Fuzzy	<code>ddf.metrics.catalog:name=Queries.Fuzzy</code>	Count	A count of the number of queries attempted that included a string comparison criteria with fuzzy searching enabled as part of the search criteria.
Catalog Queries Spatial	<code>ddf.metrics.catalog:name=Queries.Spatial</code>	Count	A count of the number of queries attempted that included a spatial criteria as part of the search criteria.
Catalog Queries Temporal	<code>ddf.metrics.catalog:name=Queries.Temporal</code>	Count	A count of the number of queries attempted that included a temporal criteria as part of the search criteria.
Catalog Queries Total Results	<code>ddf.metrics.catalog:name=Queries.TotalResults</code>	Mean	An average of the total number of results returned from executed queries. This total results data is averaged over the metric's sample rate.
Catalog Queries Xpath	<code>ddf.metrics.catalog:name=Queries.Xpath</code>	Count	A count of the number of queries attempted that included a Xpath criteria as part of the search criteria.
Catalog Resource Retrieval	<code>ddf.metrics.catalog:name=Resource</code>	Count	A count of the number of products retrieved.

Metric	JMX MBean Name	MBean Attribute Name	Description
Services Latency	<code>ddf.metrics.services:name=Latency</code>	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.

### 15.5.1. Source Metrics

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

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

Metric	JMX MBean Name	MBean Attribute Name	Description
Source <sourceId> Exceptions	<code>ddf.metrics.catalog.source:name=&lt;sourceId&gt;.Exceptions</code>	Count	A count of the total number of exceptions, of all types, thrown from catalog queries executed on this source.
Source <sourceId> Queries	<code>ddf.metrics.catalog.source:name=&lt;sourceId&gt;.Queries</code>	Count	A count of the number of queries attempted on this source.
Source <sourceId> Queries Total Results	<code>ddf.metrics.catalog.source:name=&lt;sourceId&gt;.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_INSTALL_DIR>/data/metrics` directory. Also note that the source metrics' names are modified to remove all non-alphanumeric characters and renamed in camelCase.

### 15.5.2. Usage

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

### 15.5.3. Install and Uninstall

The Metrics Collecting application is installed by default.

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

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

### 15.5.4. Configuration

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

### 15.5.5. Known Issues

None

## 15.6. Metrics Reporting Application

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

PowerPoint formats.

### 15.6.1. Usage

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

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

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

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

**WARNING**

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

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

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

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

The user can also specify custom time ranges by adjusting the URL used to access the metric's graph. The Catalog Queries metric data may also be graphed for a specific time range by specifying the `startDate` and `endDate` query parameters in the URL.

**WARNING**

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

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

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

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

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

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

Parameter	Description	Example
<b>startDate</b>	Specifies the start of the time range of the search on the metric's data (RFC-3339 - Date and Time format, i.e. <b>YYYY-MM-DDTHH:mm:ssZ</b> ). Date/time must be earlier than the endDate.  <i>This parameter cannot be used with the dateOffset parameter.</i>	<b>startDate=2013-03-31T06:00:00-07:00</b>
<b>endDate</b>	Specifies the end of the time range of the search on the metric's data (RFC-3339 - Date and Time format, i.e. <b>YYYY-MM-DDTHH:mm:ssZ</b> ). Date/time must be later than the startDate.  <i>This parameter cannot be used with the dateOffset parameter.</i>	<b>endDate=2013-04-01T11:00:00-07:00</b>
<b>dateOffset</b>	Specifies an offset, backwards from the current time, to search on the modified time field for entries. Defined in seconds and must be a positive Integer.  <i>This parameter cannot be used with the startDate or endDate parameters.</i>	<b>dateOffset=1800</b>

Parameter	Description	Example
<code>yAxisLabel</code>	(optional) the label to apply to the graph's y-axis. Will default to the metric's name, e.g., Catalog Queries.  This parameter is only applicable for the metric's graph display format.	Catalog Query Count
<code>title</code>	(optional) 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

## 15.6.2. Metric Data Supported Formats

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

The table below describes each of the supported formats, how to access them, and an example where applicable. (NOTE: all example URLs begin with `http://<DDF_HOST>:<DDF_PORT>` which is omitted in the table for brevity.)

Display Format	Description	How To Access	Example URL
PNG	Displays the metric's data as a PNG-formatted graph, where the x-axis is time and the y-axis is the metric's sampled data values.	Via hyperlink on the Metrics tab or directly via custom URL.	<p>Accessing Catalog Queries metric data for last 8 hours (<math>8 * 60 * 60 = 28800</math> seconds):</p> <pre>/services/internal/metrics/catalogQueries.png?dateOffset=28800&amp;yAxisLabel=my%20label&amp;title=my%20graph%20title</pre> <p>Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:</p> <pre>/services/internal/metrics/catalogQueries.png?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00&amp;yAxisLabel=my%20label&amp;title=my%20graph%20title</pre> <p><i>Note that the <code>yAxisLabel</code> and <code>title</code> parameters are optional.</i></p>

Display Format	Description	How To Access	Example URL
CSV	<p>Displays the metric's data as a Comma-Separated Value (CSV) file, which can be auto-displayed in Excel based on browser settings.</p> <p>The generated CSV file will consist of two columns of data: Timestamp and Value, where the first row are the column headers and the remaining rows are the metric's sampled data over the specified time range.</p>	Via hyperlink on the Metrics tab or directly via custom URL.	<p>Accessing Catalog Queries metric data for last 8 hours (<math>8 * 60 * 60 = 28800</math> seconds):</p> <p><a href="/services/internal/metrics/catalogQueries.csv?dateOffset=28800">/services/internal/metrics/catalogQueries.csv?dateOffset=28800</a></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><a href="/services/internal/metrics/catalogQueries.csv?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00">/services/internal/metrics/catalogQueries.csv?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00</a></p>
XLS	<p>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</p>	Via hyperlink on the Metrics tab or directly via custom URL.	<p>Accessing Catalog Queries metric data for last 8 hours (<math>8 * 60 * 60 = 28800</math> seconds):</p> <p><a href="/services/internal/metrics/catalogQueries.xls?dateOffset=28800">/services/internal/metrics/catalogQueries.xls?dateOffset=28800</a></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><a href="/services/internal/metrics/catalogQueries.xls?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00">/services/internal/metrics/catalogQueries.xls?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00</a></p>

<b>Display Format</b>	<b>Description</b>	<b>How To Access</b>	<b>Example URL</b>
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 (<math>8 * 60 * 60 = 28800</math> seconds):</p> <p><a href="/services/internal/metrics/catalogQueries.ppt?dateOffset=28800">/services/internal/metrics/catalogQueries.ppt?dateOffset=28800</a></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><a href="/services/internal/metrics/catalogQueries.ppt?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00">/services/internal/metrics/catalogQueries.ppt?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00</a></p>

Display Format	Description	How To Access	Example URL
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 (<math>8 * 60 * 60 = 28800</math> 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&amp;endDate=2013-04-02T10:00:00-07:00</code></p> <p>Sample XML-formatted output would look like:</p> <pre>[source,xml,linenums] ----- &lt;catalogQueries&gt; &lt;title&gt;Catalog Queries for Apr 15 2013 08:45:53 to Apr 15 2013 09:00:53&lt;/title&gt; &lt;data&gt; &lt;sample&gt; &lt;timestamp&gt;Apr 15 2013 08:45:00&lt;/timestamp&gt; &lt;value&gt;361&lt;/value&gt; &lt;/sample&gt; &lt;sample&gt; &lt;timestamp&gt;Apr 15 2013 09:00:00&lt;/timestamp&gt; &lt;value&gt;353&lt;/value&gt; &lt;/sample&gt; &lt;totalCount&gt;5721&lt;/totalCount&gt; &lt;/data&gt; &lt;/catalogQueries&gt; -----</pre>

Display Format	Description	How To Access	Example URL
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 (<math>8 * 60 * 60 = 28800</math> seconds):</p> <pre>/services/internal/metrics/catalogQueries.json?dateOffset=28800</pre> <p>Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013:</p> <pre>/services/internal/metrics/catalogQueries.json?startDate=2013-03-10T06:00:00-07:00&amp;endDate=2013-04-02T10:00:00-07:00</pre> <p>.Sample JSON-formatted Output [source,json,linenums]</p> <pre>---- {   "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     }   ] }----</pre>

### 15.6.3. Metrics Aggregate Reports

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

The aggregate reports provided are:

- Weekly reports for each week up to the past four **complete** weeks from current time. A complete week is defined as a week from Monday through Sunday. For example, if current time is Thursday, April 11, 2013, the past complete week would be from April 1 through April 7.
- Monthly reports for each month up to the past 12 **complete** months from current time. A complete month is defined as the full month(s) preceding current time. For example, if current time is

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

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

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

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

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

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

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

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

where **<format>** is either **xls** or **ppt** and the **<start\_date\_value>** and **<end\_date\_value>** specify the custom time range for the report.

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

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

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

#### 15.6.4. Add Custom Metrics to the Metrics Tab

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

**WARNING**

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

#### 15.6.5. Install and Uninstall

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

#### 15.6.6. Configuration

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

The [metrics-reporting](#) feature can only be installed and uninstalled. It is installed by default.

#### 15.6.7. Known Issues

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

### 15.7. Security Core API

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

### 15.7.1. Install and Uninstall

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

### 15.7.2. Configuration

None

### 15.7.3. Implementation Details

#### Imported Services

None

#### Exported Services

None

## 15.8. Compression Services

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

### 15.8.1. Install and Uninstall

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

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

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

Compression Type	Description
<code>exi</code>	Adds Efficient XML Interchange (EXI) support to outgoing responses. EXI is an W3C standard for XML encoding that shrinks xml to a smaller size than normal GZip compression. More information is available at <a href="#">EXI</a> .
<code>gzip</code>	Adds GZip compression to in and outgoing messages through CXF components. Code comes with CXF.

**WARNING**

Due to the way CXF features work, the compression services either need to be installed BEFORE the desired CXF service is started or the CXF service needs to be refreshed / restarted after the compression service is installed.

## 15.8.2. Configuration

None

## 15.8.3. Implementation Details

### Imported Services

None

### Exported Services

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

# 16. Integrating DDF Security

Version: 2.9.0

This section supports integration of this application with external frameworks.

## 16.1. Security CAS

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

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

### 16.1.1. Components

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

### 16.1.2. Security CAS Client

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

#### Installing CAS Client

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

#### Configuring CAS Client

*Table 35. Settings*

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

## Implementation Details

### Imported Services

None

### Exported Services

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

## 16.1.3. Security CAS Implementation

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

### Configuring

None.

## Implementation Details

### Imported Services

None

### Exported Services

None

## 16.1.4. Security CAS Server

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

### Configuring Security CAS Server

N/A - Not a bundle

### Implementation Details

N/A - Not a bundle

## 16.1.5. Security CAS Token Validator

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

### Installing

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

### Configuring

#### Settings

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

### Implementation Details

#### Imported Services

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

#### Exported Services

Registered Interfaces	Implementation Class	Properties Set
<code>ddf.catalog.util.DdfConfigurationWatcher</code> <code>org.apache.cxf.sts.token.validator.Tok enValidator</code>	<code>ddf.security.cas.WebSSOTokenValidator</code>	CAS Server URL and Encryption Service reference

## 16.1.6. Security CAS CXF Servlet Filter

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

### Installing

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

### Configuring

#### Settings

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

#### WARNING

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

### Implementation Details

#### Imported Services

Registered Interface	Availability	Multiple
<code>javax.servlet.Filter</code>	required	false

#### Exported Services

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

## 16.2. Security Core

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

## 16.2.1. Components

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

## 16.2.2. Security Core Commons

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

### Configuring

None

### Implementation Details

#### Imported Services

None

#### Exported Services

None

## 16.2.3. Security Core Implementation

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

### Configuring

None

### Install and Uninstall

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

### Implementation Details

#### Imported Services

Registered Interface	Availability	Multiple
<code>org.apache.shiro.realm.Realm</code>	optional	true

#### Exported Services

Registered Interface	Implementation Class	Properties Set
<code>ddf.security.service.SecurityManager</code>	<code>ddf.security.service.impl.SecurityManagerImpl</code>	None

### 16.2.4. Security Encryption

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

#### Components

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

## 16.3. Security Encryption API

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

### 16.3.1. Installing Security Encryption API

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

### 16.3.2. Configuring

None

### 16.3.3. Implementation Details

## **Imported Services**

None

## **Exported Services**

None

### **16.3.4. Security Encryption Commands**

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

#### **Installing Security Encryption Commands**

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

#### **Configuring**

None

## **Implementation Details**

### **Imported Services**

None

### **Exported Services**

None

### **16.3.5. Security Encryption Implementation**

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

#### **Installing Security Encryption Implementation**

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

#### **Configuring**

None

## Implementation Details

### Imported Services

None

### Exported Services

Registered Interface	Implementation Class	Properties Set
<code>ddf.security.encryption.Encrypt ionService</code>	<code>ddf.security.encryption.impl.En cryptionServiceImpl</code>	Key

## 16.4. 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.

### 16.4.1. Components

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

## 16.5. Installing the Embedded LDAP Server

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

### 16.5.1. Run the Embedded LDAP Instance

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

### 16.5.2. Configuration

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

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

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

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

### 16.5.3. Trust Certificates

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

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

### 16.5.4. Connect to Standalone LDAP Servers

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

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

### 16.5.5. Embedded LDAP Configuration

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

### 16.5.6. Default Settings

#### Ports

Protocol	Default Port
LDAP	1389
LDAPS	1636

Protocol	Default Port
StartTLS	1389

## Users

### LDAP Users

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

### LDAP Admin

Username	Password	Groups	Attributes	Description
admin	secret			Administrative User for LDAP

## Schemas

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

Schema File Name	Schema Description
<b>00-core.ldif</b>	This file contains a core set of attribute type and objectclass definitions from several standard LDAP documents, including <a href="#">draft-ietf-boreham-numsubordinates</a> , <a href="#">draft-findlay-ldap-groupofentries</a> , <a href="#">draft-furuseth-ldap-untypedobject</a> , <a href="#">draft-good-ldap-changelog</a> , <a href="#">draft-ietf-ldup-subentry</a> , <a href="#">draft-wahl-ldap-adminaddr</a> , 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.
<b>01-pwpolicy.ldif</b>	This file contains schema definitions from <a href="#">draft-behera-ldap-password-policy</a> , which defines a mechanism for storing password policy information in an LDAP directory server.
<b>02-config.ldif</b>	This file contains the attribute type and <b>objectclass</b> definitions for use with the directory server configuration.
<b>03-changelog.ldif</b>	This file contains schema definitions from <a href="#">draft-good-ldap-changelog</a> , which defines a mechanism for storing information about changes to directory server data.
<b>03-rfc2713.ldif</b>	This file contains schema definitions from RFC 2713, which defines a mechanism for storing serialized Java objects in the directory server.
<b>03-rfc2714.ldif</b>	This file contains schema definitions from RFC 2714, which defines a mechanism for storing CORBA objects in the directory server.
<b>03-rfc2739.ldif</b>	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.
<b>03-rfc2926.ldif</b>	This file contains schema definitions from RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.
<b>03-rfc3112.ldif</b>	This file contains schema definitions from RFC 3112, which defines the authentication password schema.
<b>03-rfc3712.ldif</b>	This file contains schema definitions from RFC 3712, which defines a mechanism for storing printer information in the directory server.
<b>03-uddiv3.ldif</b>	This file contains schema definitions from RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.
<b>04-rfc2307bis.ldif</b>	This file contains schema definitions from the <a href="#">draft-howard-rfc2307bis</a> specification, used to store naming service information in the directory server.
<b>05-rfc4876.ldif</b>	This file contains schema definitions from RFC 4876, which defines a schema for storing Directory User Agent (DUA) profiles and preferences in the directory server.

Schema File Name	Schema Description
<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.

## 16.5.7. Configuration

### Start and Stop

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

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

### Settings

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

Configuration Name	Description
LDAP Port	Sets the port for LDAP ( <code>plaintext</code> and <code>StartTLS</code> ). 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.

## 16.5.8. Limitations

Current limitations for the embedded LDAP instances include:

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

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

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

### 16.5.9. External Links

Location to the default base LDIF file in the DDF [source code](#).

[OpenDJ documentation](#)

### 16.5.10. LDAP Administration

OpenDJ provides a number of tools for LDAP administration. Refer to the [OpenDJ Admin Guide](#).

#### Download the Admin Tools

Download [OpenDJ \(Version 2.4.6\)](#) and the included tool suite.

#### Use the Admin Tools

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

#### Example Commands for Disabling/Enabling a User's Account

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

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

#### Verify the Account is Disabled

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

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

## Enable the Account

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

## Verify the Account is Enabled

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

```

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

```

## 16.6. Security PEP

The DDF 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.

### 16.6.1. Components

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

### 16.6.2. Security PEP Interceptor

The Security PEP Interceptor bundle contains the

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

## Installing Security PEP Interceptor

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

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

## Configuring

None

## Implementation Details

### Imported Services

None

### Exported Services

None

## 16.7. Security STS

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

### 16.7.1. Components

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

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

## 16.7.2. Security STS Client Config

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

### Installing Security STS Client Config

This bundle is installed by default.

### Configuring

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

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

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

### 16.7.3. Implementation Details

#### Imported Services

Registered Interface	Availability	Multiple
<code>ddf.catalog.DdfConfigurationWatcher</code>	required	true
<code>org.osgi.service.cm.ConfigurationAdmin</code>	required	false

#### Exported Services

None

### 16.7.4. External/WS-S STS Support

#### Security STS WSS

This configuration works just like the STS Client Config for the internal STS, but produces standard requests instead of the custom DDF ones. It supports two new auth types for the context policy manager, WSSBASIC and WSSPKI.

## Security STS Address Provider

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

### 16.7.5. Security STS LDAP Claims Handler

The DDF Security STS LDAP Claims Handler bundle adds functionality to the STS server that allows it to retrieve claims from an LDAP server. It also adds mappings for the LDAP attributes to the STS SAML claims.

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

#### Installing Security STS LDAP Claims Handler

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

#### Configuring

##### Settings

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

Configuration Name	Default Value	Additional Information
LDAP URL	<code>ldap://localhost:1389`</code>	
LDAP Bind User DN	<code>cn=admin</code>	
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>

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

## Implementation Details

### Imported Services

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

### Exported Services

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

## 16.7.6. Security STS LDAP Login

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

### Installing Security STS LDAP Claims Handler

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

### Configuring

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

Configuration Name	Default Value	Additional Information
LDAP URL	ldaps://localhost:1636	

Configuration Name	Default Value	Additional Information
LDAP Bind User DN	<code>cn=admin</code>	
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>	
SSL Keystore Alias	<code>localhost</code>	This alias is used when connecting to the LDAP using SSL/TLS (LDAPS) or startTLS.

## Implementation Details

### Imported Services

None

### Exported Services

None

## 16.8. Security STS Service

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

### Installing Security STS Realm

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

### Configuring

None

## Implementation Details

### Imported Services

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

## Exported Services

Registered Interfaces	Implementation Class	Properties Set
ddf.catalog.util.DdfConfigurator onWatcher org.apache.shiro.realm.Realm	ddf.security.realm.sts.StsRealm	None

### 16.8.4. Security STS Server

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

**NOTE**

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

### 16.8.5. Installing Security STS Server

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

### 16.8.6. Configuring

#### Settings

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

Configuration Name	Default Value	Additional Information
SAML Assertion Lifetime	1800	
Token Issuer	localhost	The name of the server issuing tokens. Generally this is the cn or hostname of this machine on the network.
Signature Username	localhost	Alias of the private key in the STS Server's keystore used to sign messages.
Encryption Username	localhost	Alias of the private key in the STS Server's keystore used to encrypt messages.

### 16.8.7. Implementation Details

#### Imported Services

Registered Interface	Availability	Multiple
<code>org.apache.cxf.sts.claims.ClaimHandler</code>	optional	true
<code>org.apache.cxf.sts.token.validator.TokenValidator</code>	optional	true

## Exported Services

None

## 16.9. Security PDP

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

### 16.9.1. Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
<code>security-pdp-authrealm</code>	<code>security-pdp-authz</code>	Security PDP Java Realm

## 16.10. Security PDP AuthZ Realm

The DDF Security PDP AuthZ Realm exposes a realm service that makes decisions on authorization requests using the attributes stored within the 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.

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

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

### 16.10.1. Configuring

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

Configuration Name	Default Value	Additional Description
Match-All Mappings		These map user attributes to 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.

### Implementation Details

#### Imported Services

None

#### Exported Services

Registered Interfaces	Implementation Class	Properties Set
<code>org.apache.shiro.realm.Realm</code>	<code>ddf.security.pdp.realm.AuthzRealm</code>	None
<code>org.apache.shiro.authz.Authorizer</code>		

### 16.10.2. Guest Interceptor

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

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

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

## Installing Guest Interceptor

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

## Configuring Guest Interceptor

### 16.10.3. Configuring via the Admin Console

1. Navigate to the Admin Console at <https://localhost:8993/admin>
2. Click the DDF Security application tile
3. Click the **Configuration** tab
4. Click the **Security STS Guest Claims Handler** configuration
5. Click the + next to Attributes to add a new attribute
6. Add any 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.

## 16.11. Security IdP

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

### 16.11.1. Components

Bundle Name	Located in Feature	Description
<b>security-idp-sp</b>	<b>security-idp</b>	IdP Service Provider
<b>security-idp-server</b>	<b>security-idp</b>	IdP Server

## **16.11.2. Installing Security IdP**

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

## **16.11.3. Security IdP Service Provider**

The IdP client that interacts with the specified Identity Provider.

## **16.11.4. Security IdP Server**

An internal Identity Provider solution.

## **16.11.5. Limitations**

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

# 17. Integrating DDF Solr

Version: 2.9.0

This section supports integration of this application with external frameworks.

Some notable features of the Solr Catalog Provider include:

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

## 17.1. Embedded Solr Catalog Provider Pros and Cons

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

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

### 17.1.1. When to Use

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

## 17.2. Solr Catalog External Provider Pros and Cons

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

Feature	Pro	Con
(Administrative) Tools	<ul style="list-style-type: none"> <li>* Contains Solr Admin GUI, which allows admins to query, check health, see metrics, see configuration files and preferences, etc.</li> <li>* External open source tools like Luke will work to allow admins to check index contents.</li> <li>* JMX metrics reporting is enabled.</li> </ul>	
Security	<ul style="list-style-type: none"> <li>* Inherits app server security.</li> </ul>	<ul style="list-style-type: none"> <li>* Web Console must be secured and is openly accessible.</li> <li>* REST-like HTTP/XML and JSON APIs must be secured.</li> <li>* Current Catalog Provider implementation requires sending unsecured messages to Solr. Without a coded solution, requires network or firewall restrictions in order to secure.</li> </ul>
Performance	<ul style="list-style-type: none"> <li>* If scaled, high performance.</li> <li>* Near real-time indexing.</li> </ul>	<ul style="list-style-type: none"> <li>* Possible network latency impact</li> <li>* Extra overhead when sent over HTTP. Extra parsing for XML, JSON, or other interface formats.</li> <li>* Possible limitations upon requests and queries dependent on HTTP server settings.</li> </ul>
Backup/Recovery	<ul style="list-style-type: none"> <li>* Built-in recovery tools that allow in-place backups (does not require server shutdown).</li> <li>* Backup of Solr indexes can be scripted.</li> </ul>	<ul style="list-style-type: none"> <li>* Recovery is performed as an HTTP request.</li> </ul>

### 17.2.1. When to Use

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

### 17.2.2. Implementation Details

#### Indexing Text

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

# 18. Integrating DDF Spatial

Version: 2.9.0

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

This guide supports integration of this application with external frameworks.

## 18.1. Integrating DDF with CSW

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

### 18.1.1. CSW v2.0.2 Endpoint

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

#### Using the CSW Endpoint

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

#### GetCapabilities Operation

The `GetCapabilites` operation is meant to describe the operations the catalog supports and the URLs used to access those operations.

The CSW endpoint supports both `HTTP GET` and `HTTP POST` requests for the `GetCapabilities` operation. The response to either request will always be a `csw:Capabilities` XML document. This XML document is defined by the [CSW-Discovery XML Schema](#).

#### GetCapabilities HTTP GET

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

#### GetCapabilities KVP Encoding

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

#### GetCapabilities HTTP POST

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

**GetCapabilities** element of the <http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd> [CSW-Discovery XML Schema].

### **GetCapabilities XML Request**

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

### **GetCapabilities Response**

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

## Sample Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Capabilities xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml=
"http://www.opengis.net/gml" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:ns6=
"http://www.w3.org/2001/SMIL20/" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dct=
"http://purl.org/dc/terms/" xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance" version="2.0.2"
ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
  <ows:ServiceIdentification>
    <ows:Title>Catalog Service for the Web</ows:Title>
    <ows:Abstract>DDF CSW Endpoint</ows:Abstract>
    <ows:ServiceType>CSW</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>DDF</ows:ProviderName>
    <ows:ProviderSite/>
    <ows:ServiceContact/>
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get ns2:href="https://localhost:8993/services/csw"/>
          <ows:Post ns2:href="https://localhost:8993/services/csw">
            <ows:Constraint name="PostEncoding">
              <ows:Value>XML</ows:Value>
            </ows:Constraint>
          </ows:Post>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="sections">
        <ows:Value>ServiceIdentification</ows:Value>
        <ows:Value>ServiceProvider</ows:Value>
        <ows:Value>OperationsMetadata</ows:Value>
        <ows:Value>Filter_Capabilities</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="DescribeRecord">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get ns2:href="https://localhost:8993/services/csw"/>
          <ows:Post ns2:href="https://localhost:8993/services/csw">
            <ows:Constraint name="PostEncoding">
              <ows:Value>XML</ows:Value>
            </ows:Constraint>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
  </ows:OperationsMetadata>
</csw:Capabilities>
```

```

        </ows:Post>
    </ows:HTTP>
</ows:DCP>
<ows:Parameter name="typeName">
    <ows:Value>csw:Record</ows:Value>
    <ows:Value>gmd:MD_Metadata</ows:Value>
</ows:Parameter>
<ows:Parameter name="OutputFormat">
    <ows:Value>application/xml</ows:Value>
    <ows:Value>application/json</ows:Value>
    <ows:Value>application/atom+xml</ows:Value>
    <ows:Value>text/xml</ows:Value>
</ows:Parameter>
<ows:Parameter name="schemaLanguage">
    <ows:Value>http://www.w3.org/XMLSchema</ows:Value>
    <ows:Value>http://www.w3.org/XML/Schema</ows:Value>
    <ows:Value>http://www.w3.org/2001/XMLSchema</ows:Value>
    <ows:Value>http://www.w3.org/TR/xmlschema-1/</ows:Value>
</ows:Parameter>
</ows:Operation>
<ows:Operation name="GetRecords">
    <ows:DCP>
        <ows:HTTP>
            <ows:Get ns2:href="https://localhost:8993/services/csw"/>
            <ows:Post ns2:href="https://localhost:8993/services/csw">
                <ows:Constraint name="PostEncoding">
                    <ows:Value>XML</ows:Value>
                </ows:Constraint>
            </ows:Post>
        </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="ResultType">
        <ows:Value>hits</ows:Value>
        <ows:Value>results</ows:Value>
        <ows:Value>validate</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="OutputFormat">
        <ows:Value>application/xml</ows:Value>
        <ows:Value>application/json</ows:Value>
        <ows:Value>application/atom+xml</ows:Value>
        <ows:Value>text/xml</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="OutputSchema">
        <ows:Value>urn:catalog:metacard</ows:Value>
        <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
        <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="typeNames">

```

```

        <ows:Value>csw:Record</ows:Value>
        <ows:Value>gmd:MD_Metadata</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="ConstraintLanguage">
        <ows:Value>Filter</ows:Value>
        <ows:Value>CQL_Text</ows:Value>
    </ows:Parameter>
    <ows:Constraint name="FederatedCatalogs">
        <ows:Value>Source1</ows:Value>
        <ows:Value>Source2</ows:Value>
    </ows:Constraint>
</ows:Operation>
<ows:Operation name="GetRecordById">
    <ows:DCP>
        <ows:HTTP>
            <ows:Get ns2:href="https://localhost:8993/services/csw"/>
            <ows:Post ns2:href="https://localhost:8993/services/csw">
                <ows:Constraint name="PostEncoding">
                    <ows:Value>XML</ows:Value>
                </ows:Constraint>
            </ows:Post>
        </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="OutputSchema">
        <ows:Value>urn:catalog:metacard</ows:Value>
        <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
        <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
        <ows:Value>http://www.iana.org/assignments/media-types/application/octet-
stream</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="OutputFormat">
        <ows:Value>application/xml</ows:Value>
        <ows:Value>application/json</ows:Value>
        <ows:Value>application/atom+xml</ows:Value>
        <ows:Value>text/xml</ows:Value>
        <ows:Value>application/octet-stream</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="ResultType">
        <ows:Value>hits</ows:Value>
        <ows:Value>results</ows:Value>
        <ows:Value>validate</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="ElementSetName">
        <ows:Value>brief</ows:Value>
        <ows:Value>summary</ows:Value>
        <ows:Value>full</ows:Value>
    </ows:Parameter>
</ows:Operation>

```

```

<ows:Operation name="Transaction">
    <ows:DCP>
        <ows:HTTP>
            <ows:Post ns2:href="https://localhost:8993/services/csw">
                <ows:Constraint name="PostEncoding">
                    <ows:Value>XML</ows:Value>
                </ows:Constraint>
            </ows:Post>
        </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="typeNames">
        <ows:Value>xml</ows:Value>
        <ows:Value>appxml</ows:Value>
        <ows:Value>csw:Record</ows:Value>
        <ows:Value>gmd:MD_Metadata</ows:Value>
        <ows:Value>tika</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="ConstraintLanguage">
        <ows:Value>Filter</ows:Value>
        <ows:Value>CQL_Text</ows:Value>
    </ows:Parameter>
</ows:Operation>
<ows:Parameter name="service">
    <ows:Value>CSW</ows:Value>
</ows:Parameter>
<ows:Parameter name="version">
    <ows:Value>2.0.2</ows:Value>
</ows:Parameter>
</ows:OperationsMetadata>
<ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities>
        <ogc:GeometryOperands>
            <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
            <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
        </ogc:GeometryOperands>
        <ogc:SpatialOperators>
            <ogc:SpatialOperator name="BBOX"/>
            <ogc:SpatialOperator name="Beyond"/>
            <ogc:SpatialOperator name="Contains"/>
            <ogc:SpatialOperator name="Crosses"/>
            <ogc:SpatialOperator name="Disjoint"/>
            <ogc:SpatialOperator name="DWithin"/>
            <ogc:SpatialOperator name="Intersects"/>
            <ogc:SpatialOperator name="Overlaps"/>
            <ogc:SpatialOperator name="Touches"/>
            <ogc:SpatialOperator name="Within"/>
        </ogc:SpatialOperators>
    </ogc:Spatial_Capabilities>
</ogc:Filter_Capabilities>

```

```

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

```

## DescribeRecord Operation

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

### *Namespace*

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

### *Service*

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

### *Version*

The version of the service being used (2.0.2).

### *OutputFormat*

The requester wants the response to be in this intended output. Currently, only one format is supported (`application/xml`). If this parameter is supplied, it is validated against the known type. If this parameter is not supported, it passes through and returns the XML response upon success.  
**SchemaLanguage:** The schema language from the request. This is validated against the known list of schema languages supported (refer to <http://www.w3.org/XML/Schema>).

## DescribeRecord HTTP GET

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

### DescribeRecord KVP Encoding

```
http://<DDF_HOST>:<DDF_PORT>/services/csw?service=CSW&version=2.0.2&request=DescribeRecord&NAME_SPACE=xmlns(http://www.opengis.net/cat/csw/2.0.2)&outputFormat=application/xml&schemaLanguage=http://www.w3.org/XML/Schema
```

## DescribeRecord HTTP POST

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

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

## DescribeRecord Response

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

## DescribeRecord Response

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

                </xsd:appinfo>
                <xsd:documentation xml:lang="en">
                    This schema defines the basic record types that must be supported
                    by all CSW implementations. These correspond to full, summary, and
                    brief views based on DCMI metadata terms.
                </xsd:documentation>

                </xsd:annotation>
                <xsd:import namespace="http://purl.org/dc/terms/" schemaLocation="rec-
dcterms.xsd"/>
                <xsd:import namespace="http://purl.org/dc/elements/1.1/" schemaLocation="rec-
dcmes.xsd"/>
                <xsd:import namespace="http://www.opengis.net/ows" schemaLocation=
"../../ows/1.0.0/owsAll.xsd"/>
                <xsd:element abstract="true" id="AbstractRecord" name="AbstractRecord" type=
"csw:AbstractRecordType"/>
                <xsd:complexType abstract="true" id="AbstractRecordType" name=
"AbstractRecordType"/>
                <xsd:element name="DCMIRecord" substitutionGroup="csw:AbstractRecord" type=
"csw:DCMIRecordType"/>
                <xsd:complexType name="DCMIRecordType">
                    <xsd:annotation>
                        <xsd:documentation xml:lang="en">
                            This type encapsulates all of the standard DCMI metadata terms,
                            including the Dublin Core refinements; these terms may be mapped
                            to the profile-specific information model.
                        </xsd:documentation>

```

```

</xsd:annotation>
<xsd:complexContent>
  <xsd:extension base="csw:AbstractRecordType">
    <xsd:sequence>
      <xsd:group ref="dct:DCMI-terms"/>

    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>

</xsd:complexType>
<xsd:element name="BriefRecord" substitutionGroup="csw:AbstractRecord" type="csw:BriefRecordType"/>
<xsd:complexType final="#all" name="BriefRecordType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This type defines a brief representation of the common record
      format. It extends AbstractRecordType to include only the
      dc:identifier and dc:type properties.
    </xsd:documentation>

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

        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
<xsd:element name="SummaryRecord" substitutionGroup="csw:AbstractRecord" type="csw:SummaryRecordType"/>
<xsd:complexType final="#all" name="SummaryRecordType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This type defines a summary representation of the common record
    </xsd:documentation>
  </xsd:annotation>

```

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: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:restriction>
  </xs:simpleType>

```

```

<xs:enumeration value="planningCadastre"/>
<xs:enumeration value="society"/>
<xs:enumeration value="structure"/>
<xs:enumeration value="transportation"/>
<xs:enumeration value="utilitiesCommunication"/>

</xs:restriction>

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

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

</xs:complexType>
<xs:element name="MD_CharacterSetCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_CharacterSetCode_PropertyType">
<xs:sequence minOccurs="0">
<xs:element ref="gmd:MD_CharacterSetCode"/>

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

</xs:complexType>
<xs:element name="MD_SpatialRepresentationTypeCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_SpatialRepresentationTypeCode_PropertyType">
<xs:sequence minOccurs="0">
<xs:element ref="gmd:MD_SpatialRepresentationTypeCode"/>

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

</xs:complexType>
<xs:element name="MD_ProgressCode" substitutionGroup="gco:CharacterString"
type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_ProgressCode_PropertyType">
<xs:sequence minOccurs="0">
<xs:element ref="gmd:MD_ProgressCode"/>

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

```

```

</xs:complexType>
<xs:element name="MD_KeywordTypeCode" substitutionGroup="gco:CharacterString"
type="gco:CodeListValue_Type"/>
<xs:complexType name="MD_KeywordTypeCode_PropertyType">
<xs:sequence minOccurs="0">
<xs:element ref="gmd:MD_KeywordTypeCode"/>

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

</xs:complexType>
<xs:element name="DS_AssociationTypeCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>
<xs:complexType name="DS_AssociationTypeCode_PropertyType">
<xs:sequence minOccurs="0">
<xs:element ref="gmd:DS_AssociationTypeCode"/>

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

</xs:complexType>
<xs:element name="DS_InitiativeTypeCode" substitutionGroup=
"gco:CharacterString" type="gco:CodeListValue_Type"/>
<xs:complexType name="DS_InitiativeTypeCode_PropertyType">
<xs:sequence minOccurs="0">
<xs:element ref="gmd:DS_InitiativeTypeCode"/>

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

</xs:complexType>

</xs:schema>
</csw:SchemaComponent>
</csw:DescribeRecordResponse>

```

## DescribeRecord HTTP POST With TypeNames

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

[.DescribeRecord XML Request](#)

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

## DescribeRecord Response

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

## DescribeRecord Response

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

                </xsd:appinfo>
                <xsd:documentation xml:lang="en">
                    This schema defines the basic record types that must be supported
                    by all CSW implementations. These correspond to full, summary, and
                    brief views based on DCMI metadata terms.
                </xsd:documentation>

                </xsd:annotation>
                <xsd:import namespace="http://purl.org/dc/terms/" schemaLocation="rec-
dcterms.xsd"/>
                <xsd:import namespace="http://purl.org/dc/elements/1.1/" schemaLocation="rec-
dcmes.xsd"/>
                <xsd:import namespace="http://www.opengis.net/ows" schemaLocation=
"../../ows/1.0.0/owsAll.xsd"/>
                <xsd:element abstract="true" id="AbstractRecord" name="AbstractRecord" type=
"csw:AbstractRecordType"/>
                <xsd:complexType abstract="true" id="AbstractRecordType" name=
"AbstractRecordType"/>
                <xsd:element name="DCMIRecord" substitutionGroup="csw:AbstractRecord" type=
"csw:DCMIRecordType"/>
                <xsd:complexType name="DCMIRecordType">
                    <xsd:annotation>
                        <xsd:documentation xml:lang="en">
                            This type encapsulates all of the standard DCMI metadata terms,
                            including the Dublin Core refinements; these terms may be mapped
                            to the profile-specific information model.
                        </xsd:documentation>

```

```

</xsd:annotation>
<xsd:complexContent>
  <xsd:extension base="csw:AbstractRecordType">
    <xsd:sequence>
      <xsd:group ref="dct:DCMI-terms"/>

    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>

</xsd:complexType>
<xsd:element name="BriefRecord" substitutionGroup="csw:AbstractRecord" type="csw:BriefRecordType"/>
<xsd:complexType final="#all" name="BriefRecordType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This type defines a brief representation of the common record
      format. It extends AbstractRecordType to include only the
      dc:identifier and dc:type properties.
    </xsd:documentation>

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

        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
<xsd:element name="SummaryRecord" substitutionGroup="csw:AbstractRecord" type="csw:SummaryRecordType"/>
<xsd:complexType final="#all" name="SummaryRecordType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This type defines a summary representation of the common record
    </xsd:documentation>
  </xsd:annotation>

```

format. It extends AbstractRecordType to include the core properties.

```

</xsd:documentation>

        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="csw:AbstractRecordType">
                <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:identifier"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" ref=
"dc:title"/>
                    <xsd:element minOccurs="0" ref="dc:type"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:subject"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:format"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dc:relation"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:modified"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:abstract"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"dct:spatial"/>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" ref=
"ows:BoundingBox"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="Record" substitutionGroup="csw:AbstractRecord" type=
"csw:RecordType"/>
    <xsd:complexType final="#all" name="RecordType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
This type extends DCMIRecordType to add ows:BoundingBox;
it may be used to specify a spatial envelope for the
catalogued resource.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension base="csw:DCMIRecordType">

```

```

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

</xsd:sequence>

</xsd:extension>

</xsd:complexContent>

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

```

## GetRecords Operation

The **GetRecords** operation is the principal means of searching the catalog. The matching entries may be included with the response. The client may assign a req`uestId (absolute URI). A distributed search is performed if the **DistributedSearch** element is present and the catalog is a member of a federation. Profiles may allow alternative query expressions. There are two types of request types: one for **GET** and one for **POST**. Each request has the following common data parameters:

### *Namespace*

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

### *Service*

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

### *Version*

The version of the service being used (2.0.2).

### *OutputFormat*

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

### *OutputSchema*

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

### *ElementSetName*

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

### **GetRecords** HTTP GET

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

### **GetRecords** KVP Encoding

```
http://<DDF_HOST>:<DDF_PORT>/services/csw?service=CSW&version=2.0.2&request=GetRecords&o  
utputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPACE=  
xmlns(csw=http://www.opengis.net/cat/csw/2.0.2)&resultType=results&typeNames=csw:Record&  
ElementSetName=brief&ConstraintLanguage=CQL_TEXT&constraint=AnyText Like '%25'
```

### **GetRecords** HTTP POST

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

## GetRecords XML Request

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

## GetRecords Specific Source

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

**NOTE**

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

## GetRecords XML Request

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

## GetRecords Response

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

## GetRecords XML Response

```
<csw:GetRecordsResponse version="2.0.2" xmlns:dc="http://purl.org/dc/elements/1.1/"  
    xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows" xmlns:xs=  
    "http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
    <csw:SearchStatus timestamp="2014-02-19T15:33:44.602-05:00"/>  
    <csw:SearchResults numberofRecordsMatched="41" numberofRecordsReturned="4"  
    nextRecord="5" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">  
        <csw:SummaryRecord>  
            <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>  
            <dc:title>Product10</dc:title>  
            <dc:type>pdf</dc:type>  
            <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                <ows:LowerCorner>20.0 10.0</ows:LowerCorner>  
                <ows:UpperCorner>20.0 10.0</ows:UpperCorner>  
            </ows:BoundingBox>  
        </csw:SummaryRecord>  
        <csw:SummaryRecord>  
            <dc:identifier>c607440db9b0407e92000d9260d35444</dc:identifier>  
            <dc:title>Product03</dc:title>  
            <dc:type>pdf</dc:type>  
            <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                <ows:LowerCorner>6.0 3.0</ows:LowerCorner>  
                <ows:UpperCorner>6.0 3.0</ows:UpperCorner>  
            </ows:BoundingBox>  
        </csw:SummaryRecord>  
        <csw:SummaryRecord>  
            <dc:identifier>034cc757abd645f0abe6acaccfe194de</dc:identifier>  
            <dc:title>Product03</dc:title>  
            <dc:type>pdf</dc:type>  
            <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                <ows:LowerCorner>6.0 3.0</ows:LowerCorner>  
                <ows:UpperCorner>6.0 3.0</ows:UpperCorner>  
            </ows:BoundingBox>  
        </csw:SummaryRecord>  
        <csw:SummaryRecord>  
            <dc:identifier>5d6e987bd6084bd4919d06b63b77a007</dc:identifier>  
            <dc:title>Product01</dc:title>  
            <dc:type>pdf</dc:type>  
            <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                <ows:LowerCorner>2.0 1.0</ows:LowerCorner>  
                <ows:UpperCorner>2.0 1.0</ows:UpperCorner>  
            </ows:BoundingBox>
```

```
</csw:SummaryRecord>
</csw:SearchResults>
</csw:GetRecordsResponse>
```

## GetRecords GMD OutputSchema

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

### *GetRecords XML Request*

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

## GetRecords Response

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

## GetRecords XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<csw:GetRecordsResponse xmlns:dct="http://purl.org/dc/terms/" xmlns:xml=
"http://www.w3.org/XML/1998/namespace" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:ows="http://www.opengis.net/ows" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dc=
"http://purl.org/dc/elements/1.1/" version="2.0.2">
    <csw:SearchStatus timestamp="2016-03-23T11:31:34.531-06:00"/>
    <csw:SearchResults numberOfRecordsMatched="7" numberofRecordsReturned="1" nextRecord
="2" recordSchema="http://www.isotc211.org/2005/gmd" elementSet="summary">
        <MD_Metadata xmlns="http://www.isotc211.org/2005/gmd" xmlns:gco=
"http://www.isotc211.org/2005/gco">
            <fileIdentifier>
                <gco:CharacterString>d5f6acd5ccf34d18af5192c38a276b12</gco:CharacterStrin
g>
            </fileIdentifier>
            <hierarchyLevel>
                <MD_ScopeCode codeListValue="nitf" codeList="urn:catalog:metacard"/>
            </hierarchyLevel>
            <contact/>
            <dateStamp>
                <gco:DateTime>2015-03-04T17:23:42.332-07:00</gco:DateTime>
            </dateStamp>
            <identificationInfo>
                <MD_DataIdentification>
                    <citation>
                        <CI_Citation>
                            <title>
                                <gco:CharacterString>product.ntf</gco:CharacterString>
                            </title>
                            <date>
                                <CI_Date>
                                    <date>
                                        <gco:DateTime>2015-03-04T17:23:42.332-
07:00</gco:DateTime>
                                    </date>
                                    <dateType>
                                        <CI_DateTypeCode codeList="urn:catalog:metacard"
codeListValue="created"/>
                                    </dateType>
                                </CI_Date>
                            </date>
                            </CI_Citation>
                        </citation>
                        <abstract>
                            <gco:CharacterString></gco:CharacterString>
                        </abstract>
                    </MD_DataIdentification>
                </identificationInfo>
            </MD_Metadata>
        </csw:SearchResults>
    </csw:GetRecordsResponse>
```

```

<pointOfContact>
    <CI_ResponsibleParty>
        <organisationName>
            <gco:CharacterString></gco:CharacterString>
        </organisationName>
        <role/>
    </CI_ResponsibleParty>
</pointOfContact>
<language>
    <gco:CharacterString>en</gco:CharacterString>
</language>
<extent>
    <EX_Extent>
        <geographicElement>
            <EX_GeographicBoundingBox>
                <westBoundLongitude>
                    <gco:Decimal>32.975277</gco:Decimal>
                </westBoundLongitude>
                <eastBoundLongitude>
                    <gco:Decimal>32.996944</gco:Decimal>
                </eastBoundLongitude>
                <southBoundLatitude>
                    <gco:Decimal>32.305</gco:Decimal>
                </southBoundLatitude>
                <northBoundLatitude>
                    <gco:Decimal>32.323333</gco:Decimal>
                </northBoundLatitude>
            </EX_GeographicBoundingBox>
        </geographicElement>
    </EX_Extent>
</extent>
</MD_DataIdentification>
</identificationInfo>
<distributionInfo>
    <MD_Distribution>
        <distributor>
            <MD_Distributor>
                <distributorContact/>
                <distributorTransferOptions>
                    <MD_DigitalTransferOptions>
                        <onLine>
                            <CI_OnlineResource>
                                <linkage>
                                    <URL>http://example.com</URL>
                                </linkage>
                            </CI_OnlineResource>
                        </onLine>
                    </MD_DigitalTransferOptions>
                </distributorTransferOptions>
            </MD_Distributor>
        </distributor>
    </MD_Distribution>
</distributionInfo>

```

```

        </distributorTransferOptions>
      </MD_Distributor>
    </distributor>
  </MD_Distribution>
</distributionInfo>
</MD_Metadata>
</csw:SearchResults>
</csw:GetRecordsResponse>

```

## GetRecordById Operation

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

Clients can also retrieve products from the catalog using the **GetRecordById** operation. The client sets the output schema to <http://www.iana.org/assignments/media-types/application/octet-stream> and the output format to **application/octet-stream** within the request. The endpoint will do the following: check that only one Id is provided, otherwise an error will occur as multiple products cannot be retrieved. If both output format and output schema are set to values mentioned above, the catalog framework will retrieve the resource for that Id. The HTTP content type is then set to the resource's MIME type and the data is sent out. The endpoint also supports the resumption of partial downloads. This would typically occur at the request of a browser when a download was prematurely terminated.

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

### *Namespace*

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

### *Service*

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

### *Version*

The version of the service being used (2.0.2).

### *OutputFormat*

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

supported, it passes through and returns the XML response upon success.

#### *OutputSchema*

This is the schema language from the request. This is validated against the known list of schema languages supported (refer to <http://www.w3.org/XML/Schema>). Additionally the output schema <http://www.iana.org/assignments/media-types/application/octet-stream> is recognized and used to retrieve a product.

#### *ElementSetName*

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

#### *Id*

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

### **GetRecordById** HTTP GET

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

#### **GetRecords** KVP Encoding

```
http://<DDF_HOST>:<DDF_PORT>/services/csw?service=CSW&version=2.0.2&request=GetRecordById
&NAMESPACE=xmlns="http://www.opengis.net/cat/csw/2.0.2"&ElementSetName=full&outputFormat=
application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&id=fd7ff1535dfe47db8793
b550d4170424,ba908634c0eb439b84b5d9c42af1f871
```

### **GetRecordById** HTTP POST

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

## GetRecordById XML Request

```
<GetRecordById xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  service="CSW"
  version="2.0.2"
  outputFormat="application/xml"
  outputSchema="http://www.opengis.net/cat/csw/2.0.2"
  xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
  ../../csw/2.0.2/CSW-discovery.xsd">
  <ElementSetName>full</ElementSetName>
  <Id>182fb33103414e5cbb06f8693b526239</Id>
  <Id>c607440db9b0407e92000d9260d35444</Id>
</GetRecordById>
```

## GetRecordByIdResponse

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

## Sample - GetRecordByIdResponse

```
<csw:GetRecordByIdResponse xmlns:dc="http://purl.org/dc/elements/1.1/"  
    xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows"  
    xmlns:xs="http://www.w3.org/2001/XMLSchema"  
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
    <csw:Record>  
        <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>  
        <dct:bibliographicCitation>182fb33103414e5cbb06f8693b526239</dct:bibliographicCitation>  
            <dc:title>Product10</dc:title>  
            <dct:alternative>Product10</dct:alternative>  
            <dc:type>pdf</dc:type>  
            <dc:date>2014-02-19T15:22:51.563-05:00</dc:date>  
            <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
            <dct:created>2014-02-19T15:22:51.563-05:00</dct:created>  
            <dct:dateAccepted>2014-02-19T15:22:51.563-05:00</dct:dateAccepted>  
            <dct:dateCopyrighted>2014-02-19T15:22:51.563-05:00</dct:dateCopyrighted>  
            <dct:dateSubmitted>2014-02-19T15:22:51.563-05:00</dct:dateSubmitted>  
            <dct:issued>2014-02-19T15:22:51.563-05:00</dct:issued>  
            <dc:source>ddf.distribution</dc:source>  
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                <ows:LowerCorner>20.0 10.0</ows:LowerCorner>  
                <ows:UpperCorner>20.0 10.0</ows:UpperCorner>  
            </ows:BoundingBox>  
        </csw:Record>  
        <csw:Record>  
            <dc:identifier>c607440db9b0407e92000d9260d35444</dc:identifier>  
            <dct:bibliographicCitation>c607440db9b0407e92000d9260d35444</dct:bibliographicCitation>  
                <dc:title>Product03</dc:title>  
                <dct:alternative>Product03</dct:alternative>  
                <dc:type>pdf</dc:type>  
                <dc:date>2014-02-19T15:22:51.563-05:00</dc:date>  
                <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
                <dct:created>2014-02-19T15:22:51.563-05:00</dct:created>  
                <dct:dateAccepted>2014-02-19T15:22:51.563-05:00</dct:dateAccepted>  
                <dct:dateCopyrighted>2014-02-19T15:22:51.563-05:00</dct:dateCopyrighted>  
                <dct:dateSubmitted>2014-02-19T15:22:51.563-05:00</dct:dateSubmitted>  
                <dct:issued>2014-02-19T15:22:51.563-05:00</dct:issued>  
                <dc:source>ddf.distribution</dc:source>  
                <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                    <ows:LowerCorner>6.0 3.0</ows:LowerCorner>  
                    <ows:UpperCorner>6.0 3.0</ows:UpperCorner>  
                </ows:BoundingBox>  
            </csw:Record>  
    </csw:GetRecordByIdResponse>
```

Table 36. CSW Record to Metacard Mapping

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

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

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

## Transaction Operation

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

The CSW Transactions endpoint only supports [HTTP POST](#) requests since there are no KVP operations.

### Transaction Insert Sub-Operation [HTTP POST](#)

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

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

## Sample XML Transaction Insert Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    verboseResponse="true"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Insert typeName="csw:Record">
        <csw:Record
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dct="http://purl.org/dc/terms/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <dc:identifier></dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
            <dc:subject>Hydrography--Dictionaries</dc:subject>
            <dc:format>application/pdf</dc:format>
            <dc:date>2006-05-12</dc:date>
            <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla scelerisque cursus mi.</dct:abstract>
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
                <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
                <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:Record>
    </csw:Insert>
</csw:Transaction>
```

## Transaction Insert Response

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

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

## Sample XML Transaction Insert Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ogc="http://www.opengis.net/ogc"
                           xmlns:gml="http://www.opengis.net/gml"
                           xmlns:ns3="http://www.w3.org/1999/xlink"
                           xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                           xmlns:ns5="http://www.w3.org/2001/SMIL20/"
                           xmlns:dc="http://purl.org/dc/elements/1.1/"
                           xmlns:ows="http://www.opengis.net/ows"
                           xmlns:dct="http://purl.org/dc/terms/"
                           xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                           xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                           version="2.0.2"
                           ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
  <csw:TransactionSummary>
    <csw:totalInserted>1</csw:totalInserted>
    <csw:totalUpdated>0</csw:totalUpdated>
    <csw:totalDeleted>0</csw:totalDeleted>
  </csw:TransactionSummary>
  <csw:InsertResult>
    <csw:BriefRecord>
      <dc:identifier>2dbcfba3f3e24e3e8f68c50f5a98a4d1</dc:identifier>
      <dc:title>Aliquam fermentum purus quis arcu</dc:title>
      <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
      <ows:BoundingBox crs="EPSG:4326">
        <ows:LowerCorner>-6.171 44.792</ows:LowerCorner>
        <ows:UpperCorner>-2.228 51.126</ows:UpperCorner>
      </ows:BoundingBox>
    </csw:BriefRecord>
  </csw:InsertResult>
</csw:TransactionResponse>
```

## Transaction Update Sub-Operation HTTP POST

The Update sub-operation is a method to specify values used to change existing information in the catalog. If individual record property values are specified in the **Update** element, using the **RecordProperty** element, then those individual property values of a catalog record are replaced. The **RecordProperty** contains a **Name** and **Value** element. The **Name** element is used to specify the name of the record property to be updated. The **Value** element contains the value that will be used to update the record in the catalog. The values in the **Update** will completely replace those that are already in the record. A property is removed only if the **RecordProperty** contains a **Name** but not a **Value**.

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

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

#### Sample XML Transaction Update Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Update>
        <csw:Record
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dct="http://purl.org/dc/terms/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                <dc:identifier>2dbcfba3f3e24e3e8f68c50f5a98a4d1</dc:identifier>
                <dc:title>Aliquam fermentum purus quis arcu</dc:title>
                <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
                <dc:subject>Hydrography--Dictionaries</dc:subject>
                <dc:format>application/pdf</dc:format>
                <dc:date>2008-08-10</dc:date>
                <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla scelerisque cursus mi.</dct:abstract>
                <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
                    <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
                    <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
                </ows:BoundingBox>
            </csw:Record>
        </csw:Update>
    </csw:Transaction>
```

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

### Sample XML Transaction Update Request with filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Update>
        <csw:RecordProperty>
            <csw:Name>title</csw:Name>
            <csw:Value>Updated Title</csw:Value>
        </csw:RecordProperty>
        <csw:RecordProperty>
            <csw:Name>date</csw:Name>
            <csw:Value>2015-08-25</csw:Value>
        </csw:RecordProperty>
        <csw:RecordProperty>
            <csw:Name>format</csw:Name>
            <csw:Value></csw:Value>
        </csw:RecordProperty>
        <csw:Constraint version="2.0.0">
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>title</ogc:PropertyName>
                    <ogc:Literal>Aliquam fermentum purus quis arcu</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
        </csw:Constraint>
    </csw:Update>
</csw:Transaction>
```

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

### Sample XML Transaction Update Request with CQL filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Update>
        <csw:RecordProperty>
            <csw:Name>title</csw:Name>
            <csw:Value>Updated Title</csw:Value>
        </csw:RecordProperty>
        <csw:RecordProperty>
            <csw:Name>date</csw:Name>
            <csw:Value>2015-08-25</csw:Value>
        </csw:RecordProperty>
        <csw:RecordProperty>
            <csw:Name>format</csw:Name>
            <csw:Value></csw:Value>
        </csw:RecordProperty>
        <csw:Constraint version="2.0.0">
            <ogc:CqlText>
                title = 'Aliquam fermentum purus quis arcu'
            </ogc:CqlText>
        </csw:Constraint>
    </csw:Update>
</csw:Transaction>
```

### Transaction Update Response

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

## Sample XML Transaction Update Response

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

## Transaction Delete Sub-Operation **HTTP POST**

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

The following example shows a delete request for all records with a SpatialReferenceSystem name equal to [WGS-84](#).

### Sample XML Transaction Delete Request with filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" version="2.0.2"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc">
  <csw:Delete typeName="csw:Record" handle="something">
    <csw:Constraint version="2.0.0">
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>SpatialReferenceSystem</ogc:PropertyName>
          <ogc:Literal>WGS-84</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Delete>
</csw:Transaction>
```

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

### Sample XML Transaction Delete Request with CQL filter constraint

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

## Transaction Delete Response

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

## Sample XML Transaction Delete Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
    ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd"
    version="2.0.2">
    <csw:TransactionSummary>
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>0</csw:totalUpdated>
        <csw:totalDeleted>1</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

## Subscription GetRecords Operation

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

### Subscription GetRecords HTTP GET

#### GetRecords KVP Encoding

```
http://<DDF_HOST>:<DDF_PORT>/services/csw/subscription?service=CSW&version=2.0.2&request=
GetRecords&o
utputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPAC
E=
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%2Fcs
w%2Fsubscripti
on%2Fevent&
ConstraintLanguage=CQL_TEXT&constraint=Text Like '%25'
```

### Subscription GetRecords HTTP POST

### **Subscription GetRecords XML Request**

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

### **Subscription GetRecords HTTP PUT**

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

### **Subscription GetRecords XML Request**

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

### **Subscription GetRecords Response**

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

## Subscription GetRecords XML Response

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2008-09-28T18:49:45" xmlns=
"http://www.opengis.net/cat/csw/2.0.2"
                           xmlns:ogc="http://www.opengis.net/ogc"
                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                           instance"
                           xsi:schemaLocation="http://www.opengis.n
et/cat/csw/2.0.2 ../../../../../../csw/2.0.2/CSW-discovery.xsd">
<EchoedRequest>
  <GetRecords
    requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
    service="CSW"
    version="2.0.2"
    maxRecords="4"
    startPosition="1"
    resultType="results"
    outputFormat="application/xml"
    outputSchema="urn:catalog:metacard">
    <ResponseHandler>https://some.ddf/services/csw/subscription/event</ResponseHandle
r>
  <Query typeName="Record">
    <ElementSetName>summary</ElementSetName>
    <Constraint version="1.1.0">
      <ogc:Filter>
        <ogc:PropertyIsLike wildCard "%" singleChar "_" escapeChar="\">\>
          <ogc:PropertyName>xml</ogc:PropertyName>
          <ogc:Literal>%</ogc:Literal>
        </ogc:PropertyIsLike>
      </ogc:Filter>
    </Constraint>
  </Query>
  </GetRecords>
</EchoedRequest>
<RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

## Subscription GetRecords event Response

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

## **Subscription GetRecords event XML Response**

```
<csw:GetRecordsResponse version="2.0.2" xmlns:dc="http://purl.org/dc/elements/1.1/"  
    xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows" xmlns:xs=  
    "http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
    <csw:SearchStatus timestamp="2014-02-19T15:33:44.602-05:00"/>  
    <csw:SearchResults numberOfRecordsMatched="1" numberOfRecordsReturned="1" nextRecord  
    ="5" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">  
        <csw:SummaryRecord>  
            <dc:identifier>182fb33103414e5cbb06f8693b526239</dc:identifier>  
            <dc:title>Product10</dc:title>  
            <dc:type>pdf</dc:type>  
            <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>  
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">  
                <ows:LowerCorner>20.0 10.0</ows:LowerCorner>  
                <ows:UpperCorner>20.0 10.0</ows:UpperCorner>  
            </ows:BoundingBox>  
        </csw:SummaryRecord>  
    </csw:SearchResults>  
</csw:GetRecordsResponse>
```

## **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.n
et/cat/csw/2.0.2 ../../../../../../csw/2.0.2/CSW-discovery.xsd">
  <EchoedRequest>
    <GetRecords
      requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
      service="CSW"
      version="2.0.2"
      maxRecords="4"
      startPosition="1"
      resultType="results"
      outputFormat="application/xml"
      outputSchema="urn:catalog:metacard">
      <ResponseHandler>https://some.ddf/services/csw/subscription/event</ResponseHandle
r>
      <Query typeName="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
          <ogc:Filter>
            <ogc:PropertyIsLike wildCard "%" singleChar "_" escapeChar="\">\>
              <ogc:PropertyName>xml</ogc:PropertyName>
              <ogc:Literal>%</ogc:Literal>
            </ogc:PropertyIsLike>
          </ogc:Filter>
        </Constraint>
      </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

### **18.1.2. Install and Uninstall**

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

### **18.1.3. Configuration**

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

## 18.1.4. Known Issues

None

# 18.2. CSW v2.0.2 Source

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

## 18.2.1. Using

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

## 18.2.2. Installing and Uninstalling

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

## 18.2.3. Configuring

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

### Configure the CSW Source

*Table 37. Configurable Properties*

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

Title	Property	Type	Description	Default Value	Required
Username	username	String	Username to log into the CSW service		No
Password	password	String	Password to log into the CSW service		No
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	Yes
Force Longitude/Latitude coordinate order	isLonLatOrder	Boolean	Force Longitude/Latitude coordinate order	false	Yes
Use posList in LinearRing	usePosList	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	Yes
Metacard Mappings	metacardMappings	String	Mapping of the Metacard 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=id,resource-uri=source	No
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1)	5	Yes

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

## 18.2.4. Known Issues

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

## 18.3. GMD CSW APISO v2.0.2 Source

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

### 18.3.1. Using

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

### 18.3.2. Installing and Uninstalling

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

### 18.3.3. Configuring

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

#### Configure the CSW Source

*Table 38. Configurable Properties*

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

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

Title	Property	Type	Description	Default Value	Required
Metocard Mappings	<code>metocardMappin gs</code>	String	Mapping of the Metacard Attribute names to their CSW property names. The format should be 'title=dc:title'.	<code>id=apiso:Ident ifier,effectiv e=apiso:Public ationDate,crea ted=apiso:Cre ationDate,modif ied=apiso:Revi sionDate,title =apiso:Alterna teTitle,AnyTex t=apiso:AnyTex t,ows:Bounding Box=apiso:Boun dingBox</code>	No
Poll Interval	<code>pollInterval</code>	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1)	5	Yes
Connection Timeout	<code>connectionTime out</code>	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	30000	Yes
Receive Timeout	<code>receiveTimeout</code>	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	60000	Yes
Output schema	<code>outputSchema</code>	String	Output Schema	<a href="http://www.isot&lt;br/&gt;c211.org/2005/g&lt;br/&gt;md">http://www.isot c211.org/2005/g md</a>	Yes

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

## 18.4. Integrating DDF with KML

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

## 18.5. KML Network Link Endpoint

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

## 18.5.1. Using

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

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

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

### Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:ns2="http://www.google.com/kml/ext/2.2"
      xmlns:ns3="http://www.w3.org/2005/Atom" xmlns:ns4=
      "urn:oasis:names:tc: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>
```

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

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

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

```

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

```

## 18.5.2. Installing and Uninstalling

The [spatial-kml-networklinkendpoint](#) feature is installed by default with the Spatial App.

## 18.5.3. Configuring

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

### Configurable Properties

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

#### **18.5.4. Known Issues**

None.

## **18.6. KML Query Response Transformer**

The KML Query Response Transformer is responsible for translating a query response into a KML-formatted document. The KML will contain an HTML description for each 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.

### **18.6.1. Using**

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

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

## Example Output

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

```

<key>normal</key>
<styleUrl>#bluenormal</styleUrl>
</Pair>
<Pair>
  <key>highlight</key>
  <styleUrl>#bluehighlight</styleUrl>
</Pair>
</StyleMap>
<Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
  <name>MultiPolygon</name>
  <description><!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=windows-1252" http-equiv="content-type">
    <style media="screen" type="text/css">
      .label {
        font-weight: bold
      }
      .linkTable {
        width: 100%
      }
      .thumbnailDiv {
        text-align: center
      }
      img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    </style>
  </head>
  <body>
    <div class="thumbnailDiv"><a href="http://localhost:8181/services/catalog/sources/ddf.distribution/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>
  </body>
</Placemark>

```

```

<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>
        <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>

```

## 18.6.2. Installing and Uninstalling

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

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

## 18.6.3. Configuring

None.

## 18.6.4. Implementation Details

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

## 18.6.5. Known Issues

None.

# 18.7. 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.

## 18.7.1. Using

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

## Example Output

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

```

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

```

</tr></table></text>
  </BalloonStyle>
</Style>
<StyleMap id="default">
  <Pair>
    <key>normal</key>
    <styleUrl>#bluenormal</styleUrl>
  </Pair>
  <Pair>
    <key>highlight</key>
    <styleUrl>#bluehighlight</styleUrl>
  </Pair>
</StyleMap>
<MultiGeometry>
  <Point>
    <coordinates>102.0,2.0</coordinates>
  </Point>
  <MultiGeometry>
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0 102.0,2.0</
coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
    <Polygon>
      <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>

```

## 18.7.2. Installing and Uninstalling

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

## 18.7.3. Configuring

None.

## 18.7.4. Implementation Details

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

## 18.7.5. Known Issues

None.

# 18.8. 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](#).

## 18.8.1. Configuring

The properties below describe how to configure a Style Mapping. The configuration name is [Spatial KML Style Map Entry](#).

*Table 39. Configurable Properties*

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

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

## 18.8.2. Example Values

```

xmlns="http://www.opengis.net/kml/2.2"
  xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xmlns:ns3="http://www.w3.org/2005/Atom">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description><!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=windows-1252" http-equiv="content-type">
    <style media="screen" type="text/css">
      .label {
        font-weight: bold
      }
      .linkTable {
        width: 100%
      }
      .thumbnailDiv {
        text-align: center
      }
      img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    </style>
  </head>
  <body>
    <div class="thumbnailDiv"><a href="http://localhost:8181/services/catalog/sources/ddf.distribution/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>2014-01-07T14:58:16-0700</td>
      </tr>
    </table>
    <table class="linkTable">
      <tr>
        <td><a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource">View</a></td>
      </tr>
    </table>
  </body>
</html>
</description>
<ExtendedData>
<!-- This is a multi-line string containing the XML representation of the MultiPolygon geometry. It is included here for completeness, but is not displayed in the browser due to its length. -->
</ExtendedData>
</Placemark>

```

```

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

```

### 18.8.3. Installing and Uninstalling

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

## 18.8.4. Implementation Details

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

## 18.8.5. Known Issues

None.

## 18.9. Integrating DDF with WFS

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

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

## 18.10. Working with WFS Sources

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

### 18.10.1. WFS Features

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

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

#### Convert a WFS Feature

In order to expose WFS features to DDF clients, the WFS feature must be converted into the common data format of the DDF, a 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.

#### Create 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/converter/Converter.html> converter 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 `createMetocardFromFeature(reader, metocardType)` method implemented in the `AbstractFeatureConverter` is recommended.

```
public Metocard unmarshal(HierarchicalStreamReader reader, UnmarshallingContext ctx) {
    MetocardImpl mc = createMetocardFromFeature(reader, metocardType);
    //set your feature specific fields on the metocard 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=
"org.catalog.common.converter.factory.FeatureConverterFactory"/>
</blueprint>

```

## 18.11. WFS v1.0.0 Source

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

### 18.11.1. Using

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

### 18.11.2. Installing and Uninstalling

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

### 18.11.3. Configuring

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

#### Configuring WFS Source

##### Configurable Properties

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

Title	Property	Type	Description	Default Value	Required
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	Yes
Username	username	String	Username to log in to the WFS service. Password to log in to the WFS service.		No
Password	password	String	Password to log in to the WFS service.		No
Non Queryable Properties	nonQueryableProperties	List of Strings	Multivalued list of properties in the feature XML that should not be used as filters.		No
Poll Interval	pollInterval	Integer	Poll interval to check if the source is available (in minutes; minimum = 1).	5	Yes
Forced Spatial Filter Type	forceSpatialFilter	String	Force the selected Spatial Filter Type to be the only available Spatial Filter.	None	No
Connection Timeout	connectionTimeout	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds	30000	Yes

Title	Property	Type	Description	Default Value	Required
Receive Timeout	<code>receiveTimeout</code>	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	Yes

## 18.11.4. WFS URL

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

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

*Example GeoServer 2.5 Syntax*

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

In this case, the WFS URL would be

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

## 18.11.5. Known Issues

None.

# 18.12. WFS v2.0.0 Source

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

## 18.12.1. Using

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

## 18.12.2. Installing and Uninstalling

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

### 18.12.3. Configuring

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

#### Configuring WFS 2.0.0 Source

##### Configurable Properties

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

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

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

#### 18.12.4. WFS URL

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

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

*Example GeoServer 2.5 Syntax*

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

In this case, the WFS URL would be

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

#### 18.12.5. Known Issues

None.

#### 18.12.6. 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.

## 18.12.7. Using

Use the WFS [MetacardMapper](#) 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 [MetacardMapper](#). The [MetacardMapper](#) is set by creating a feature file configuration which specifies the appropriate mapping. The mappings are specific to a given feature type.

## 18.12.8. Installing and Uninstalling

The WFS [MetacardMapper](#) can be installed and uninstalled using the normal processes described in the Configuring DDF section.

## 18.12.9. Configuring

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

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

### Configuring WFS MetacardMapper

#### Configurable Properties

Title	Property	Type	Description	Default Value	Required
Feature Type	<code>featureType</code>	String	Feature Type. Format is <code>{URI}local-name</code>		Yes
Metocard Attribute to WFS Feature Property Mapping	<code>metocardAttrToFeaturePropMap</code>	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="2.9.0"
  description="WFS Feature to Metocard mappings for GeoServer Example
  {http://www.openplans.org/topp}states">
  <config name="org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetacardMapper-
  geoserver.http://www.openplans.org/topp.states">
    featureType = {http://www.openplans.org/topp}states
    service.factoryPid = org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetacardMapper
    featurePropToMetacardAttrMap = states.STATE_NAME=title
  </config>
</feature>
```

## Known Issues

None.

# 19. Integrating DDF Search UI

Version: 2.9.0

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

This page supports integration of this application with external frameworks.

## 19.1. CometD

The Standard Search UI utilizes [CometD](#) to communicate with the DDF. This protocol is used to execute searches, retrieve results, and receive notifications.

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

### 19.1.1. Query

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

*Table 40. Query Parameters*

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

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

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

Then the following example query could be executed:

```
/service/query
```

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

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

## 19.2. Notifications

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

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

### 19.2.1. Persistence

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

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

# 20. Extending DDF

Version: 2.9.0

## 20.1. Building DDF

Follow these procedures to build DDF from source code.

### 20.1.1. Prerequisites

- Install [J2SE 8 SDK](#).
- Verify that the `JAVA_HOME` environment variable is set to the newly installed JDK location, and that the PATH includes `%JAVA_HOME%\bin` (for Windows) or `$JAVA_HOME$/bin` (\*nix).
- Install [Git](#), if not previously installed.
- Install [Maven 3.1.0](#) or later. Verify that the `PATH` includes the `MVN_HOME/bin` directory.
  - In addition, access to a Maven repository with the latest project artifacts and dependencies is necessary in order for a successful build. The following sample `settings.xml` (the default settings file) can be used to access the public repositories with the required artifacts. For more help on how to use the `settings.xml` file, refer to the [Maven settings reference page](#).

*Sample `settings.xml` file*

```
<settings>
<!-- If proxy is needed
<proxies>
<proxy>
</proxy>
</proxies>
-->
</settings>
```

TIP

*Handy Tip on Encrypting Passwords*

See this [Maven guide](#) on how to encrypt the passwords in your `settings.xml`.

### 20.1.2. Procedures

#### Clone the DDF Repository

*Using HTTPS*

```
https://github.com/codice/ddf.git
```

## Using SSH

```
git@github.com:codice/ddf.git
```

**NOTE** Generally, SSH is faster than HTTPS, but requires setting an SSH git and passphrase with [Github](#). Additionally, there may be restrictions on the use of SSH on some networks.

### Run the Build

- Build command example for one individual repository.

```
# Build is run from the top level of the specified repository in a command line prompt or terminal.  
cd ddf-support  
mvn clean install  
  
# At the end of the build, a BUILD SUCCESS will be displayed.
```

**NOTE** The zip distribution of DDF is contained in the DDF app in the distribution/ddf/target directory after the DDF app is built.

**NOTE** It may take several moments for Maven to download the required dependencies in the first build. Build times may vary based on network speed and machine specifications.

**WARNING** In certain circumstances, the build may fail due to a '[java.lang.OutOfMemory: Java heap space](#)' error. This error is due to the large number of sub-modules in the DDF build, which causes the heap space to run out in the main Maven JVM. To fix this issue, set the system variable [MAVEN\\_OPTS](#) with the value [-Xmx2048m](#) before running the build. Example on Linux system with the bash shell: [export MAVEN\\_OPTS='-Xmx2048m'](#)

### 20.1.3. Troubleshooting Build Errors on ddf-admin and ddf-ui on a Windows Platform

Currently, the developers are using the following tools:

Name	Version
bower	1.3.2
node.js	v0.10.26
npm	1.4.3

There have been intermittent build issues during the bower install. The error code that shows is an EPERM related to either 'renaming' files or 'unlinking' files. This issue has been tracked multiple times on the bower github page. The following link contains the most recent issue that was tracked: <https://github.com/bower/bower/issues/991>

This issue will be closely monitored for a full resolution. Until a proper solution is found, there are some options that may solve the issue.

1. Re-run the build. Occasionally, the issue occurs on first run and will resolve itself on the next.
2. Clean out the cache. There may be a memory issue, and a cache clean may help solve the issue.

**NOTE**

```
bower cache clean  
npm cache clean
```

An occasional reinstall may solve the issue.

```
npm uninstall -g bower && npm install -g bower
```

3. Download and use Cygwin to perform the build. This may allow a user to simulate a run on a \*nix system, which may not experience these issues.

These options are taken from suggestions provided on github issue tickets. There have been several tickets created and closed, and several workarounds have been suggested. However, it appears that the issue still exists. Once more information develops on the resolution of this issue, this page will be updated.

## 20.2. Developing for DDF

This section discusses the several extension points and components permitted by the DDF APIs. Using code examples, diagrams, and references to specific instances of each API, this guide provides details on how to develop and extend various DDF components.

## 20.3. DDF Development Guidelines

**NOTE**

Development requires full knowledge of the DDF Catalog.

DDF is written in Java and requires a moderate amount of experience with the Java programming language, along with Java terminology, such as packages, methods, classes, and interfaces. DDF uses a small OSGi runtime to deploy components and applications. Before developing for DDF, it is necessary that developers have general knowledge on OSGi and the concepts used within. This includes, but is

not limited to, Catalog Commands and the following topics:

- The Service Registry
  - How services are registered
  - How to retrieve service references
- Bundles
  - Their role in OSGi
  - How they are developed

Documentation on OSGi can be viewed at the [OSGi Alliance website](#). Helpful literature for beginners includes *OSGi and Apache Felix 3.0 Beginner's Guide* by Walid Joseph Gédéon and *OSGi in Action: Creating Modular Applications in Java* by Richard Hall, Karl Pauls, Stuart McCulloch, and David Savage.

### 20.3.1. Recommended Hardware

Because of its modular nature, DDF may require a few or many system resources, depending on which bundles and features are deployed. In general, DDF will take advantage of available memory and processors. A 64-bit JVM is required, and a typical installation is performed on a single machine with 16GB of memory and eight processor cores.

The DDF source code is not tied to any particular IDE. However, if a developer is interested in setting up the Eclipse IDE, they can view the [Sonatype guide](#) on developing with Eclipse.

### 20.3.2. Getting Set Up

To develop on DDF, access to the source code via Github is required.

#### Integrated Development Environments (IDE)

The DDF source code is not tied to any particular IDE. However, if a developer is interested in setting up the Eclipse IDE, they can view the [Sonatype guide](#) on developing with Eclipse.

### 20.3.3. Formatting Source Code

A code formatter for the Eclipse IDE that can be used across all DDF projects will allow developers to format code similarly and minimize merge issues in the future.

DDF uses an updated version of the [Apache ServiceMix Code Formatter](#) for code formatting.

#### Load the Code Formatter Into the Eclipse IDE

1. Download the [Eclipse Code Formatter](#).

2. In Eclipse, select **Window Preferences**. The Preferences window opens.
3. Select **Java Code Style Formatter**.
4. Select the **Edit...** button and then Select the downloaded file.
5. Select the **OK** button.

### **Load the Code Formatter Into IntelliJ IDEA**

IntelliJ IDEA 13 is capable of importing Eclipse's Code Formatter directly from within IntelliJ without the use of any plugins.

1. Download the [IntelliJ Code Formatter](#).
2. Open **IntelliJ IDEA**.
3. Select **File Settings Code Style Java**.
4. Select **Manage**.
5. Select the **Import** button and select the file.

### **Format Your Source Code Using Eclipse**

A developer may write code and format it before saving.

1. Before the file is saved, highlight all of the source code in the IDE editor window.
2. Right-click on the highlighted code.
3. Select **Source Format**. The code formatter is applied to the source code and the file can be saved.

### **Set Up Save Actions in Eclipse**

A developer can also set up Save Actions to format the source code automatically.

1. Open Eclipse.
2. Select **Window Preferences (Eclipse Preferences on Mac)**. The Preferences window opens.
3. Select **Java Editor Save Actions**.
4. Select **Perform the selected actions on save**.
5. Select **Format source code**.
6. Select **Format all lines or Format edited lines**, as necessary.
7. Optionally, select **Organize imports** (recommended).

8. Select the **Apply** button.

9. Select the **OK** button.

### Format Source Code Using IntelliJ

In the toolbar, select **Code → Reformat Code** or use the keyboard shortcut **Ctrl+Alt+L**.

### 20.3.4. DDF Software Versioning

DDF follows the [Semantic Versioning White Paper](#) for bundle versioning.

### 20.3.5. Ensuring Compatibility

#### Compatibility Goals

The DDF framework, like all software, will mature over time. Changes will be made to improve efficiency, add features, and fix bugs. To ensure that components built for DDF and its sub-frameworks are compatible, developers must use caution when establishing dependencies from developed components.

#### Guidelines for Maintaining Compatibility

##### DDF Framework

For components written at the DDF Framework level (see Developing at the Framework Level), adhere to the following specifications:

Standard/Specification	Version	Current Implementation (subject to change)
OSGi Framework	4.2	Apache Karaf 2.x
		Eclipse Equinox
OSGi Enterprise Specification	4.2	Apache Aries (Blueprint) Apache Felix <a href="#">FileInstall</a> and <a href="#">ConfigurationAdmin</a>

**IMPORTANT**

Avoid developing dependencies on the implementations directly, as compatibility in future releases is not guaranteed.

### 20.3.6. DDF Catalog API

For components written for the DDF Catalog (see Developing Catalog Components), only dependencies on the current major version of the Catalog API should be used. Detailed documentation of the Catalog API can be found in the Catalog API Javadocs.

Dependency	Version Interval	Notes
DDF Catalog API	[2.0, 3.0)	Major version will be incremented (to 3.0) if/when compatibility is broken with the 2.x API.

### 20.3.7. OGC Filter

An OGC Filter is a Open Geospatial Consortium (OGC) standard that describes a query expression in terms of XML and Key-Value Pairs (KVP).

DDF originally had a custom query representation that some found difficult to understand and implement. In switching to a well-known standard like the OGC Filter, developers benefit from various third party products and third party documentation, as well as any previous experience with the standard. 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.

#### OGC filter in the DDF Catalog

The DDF Catalog Framework uses the implementation provided by Geotools, which provides a Java representation of the standard.

Geotools originally provided standard Java classes for the OGC Filter Encoding 1.0, under the package name `org.opengis.filter`, which is where `org.opengis.filter.Filter` is located. Java developers should use the Java objects exclusively to complete query tasks, rather than parsing or viewing the XML representation.

## 20.4. Development Recommendations

### 20.4.1. Javascript

Avoid using `console.log`

### 20.4.2. Package Names

Use singular package names.

### 20.4.3. Author Tags

Author tags are discouraged from being placed in the source code, as they can be a barrier to collaboration and have potential legal ramifications.

### 20.4.4. Unit Testing

All code should contain unit tests that are able to test out any localized functionality within that class.

When working with OSGi, code may have references to various services and other areas that are not available at compile-time. One way to work around the issue of these external dependencies is to use a mocking framework.

*Recommended Framework*

- NOTE** The recommended framework to use with DDF is [Mockito](#). This test-level dependency is managed by the ddf pom and is used to standardize the version being used across DDF.

## 20.4.5. Logging

There are many logging frameworks available for Java.

*Recommended Framework*

- NOTE** To maintain the best compatibility, the recommended logging framework is Simple Logging Facade for Java (SLF4J) (<http://www.slf4j.org/>), specifically the slf4j-api. SLF4J allows a very robust logging API while letting the backend implementation be switched out seamlessly. Additionally, it is compatible with pax logging and natively implemented by logback.

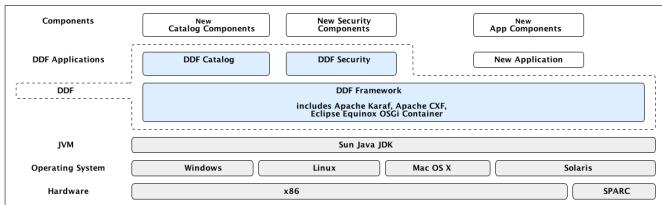
DDF code uses the first five SLF4J log levels:

1. trace (the least serious)
2. debug
3. info
4. warn
5. error (the most serious)

Examples:

```
//Check if trace is enabled before executing expense XML processing
if (LOGGER.isTraceEnabled()) {
    LOGGER.trace("XML returned: {}", XMLUtils.toString(xml));
}
//It is not necessary to wrap with LOGGER.isTraceEnabled() since slf4j will not construct
//the String unless
//trace level is enabled
LOGGER.trace("Executing search: {}", search);
```

## 20.5. "White Box" DDF Architecture



*Figure 7. Architecture Diagram*

As depicted in the architectural diagram above, 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 DDF out-of-the-box.

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 (<https://tools.codice.org/#DDFArchitecture-AdditionalUpstreamDependencies>).

DDF as a 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.

### 20.5.1. Nomenclature

#### *Capability*

A general term used to refer to an ability of the system

#### *Application*

One or more features that together form a cohesive collection of capabilities

#### *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.

### 20.5.2. OSGi Core

DDF makes use of OSGi v4.2 to provide several capabilities:

- Has a Microkernel-based foundation, which is lightweight due to its origin in embedded systems.
- Enables integrators to easily customize components to run on their system.
- Software applications are deployed as OSGi components, or bundles. Bundles are modules that can be deployed into the OSGi container (Eclipse Equinox OSGi Framework by default).
- Bundles provide flexibility allowing integrators to choose the bundles that meet their mission needs.
- Bundles provide reusable modules that can be dropped in any container.
- Provides modularity, module-based security, and low-level services, such as Hypertext Transfer Protocol (HTTP), logging, events (basic publish/subscribe), and dependency injection.
- Implements a dynamic component model that allows application updates without downtime. Components can be added or updated in a running system.
- Standardized Application Configuration (ConfigurationAdmin and MetaType)

OSGi is not an acronym, but if more context is desired the name Open Specifications Group Initiative has been suggested.

More information on OSGi is available at <http://www.osgi.org/>.

### **20.5.3. Built on Apache Karaf**

Apache Karaf is a FOSS product that includes an OSGi framework and adds extra functionality, including:

#### *Admin Console*

Useful for configuring applications, installing/uninstalling features, and viewing services such as metrics.

#### *Command Console*

Provides command line administration of the OSGi container. All functionality in the Command Console can also be performed via this command line console.

#### *Logging*

Provides centralized logging to `data/log/ddf.log`. Security logging is provided at `data/log/security.log`. Ingest error logging can be viewed in `data/log/ingest_error.log`.

#### *Provisioning*

Of libraries or applications.

#### *Security*

Provides a security framework based on Java Authentication and Authorization Service (JAAS).

### *Deployer*

Provides hot deployment of new bundles by dropping them into the <INSTALL\_DIR>/deploy directory.

### *Blueprint*

Provides an implementation of the OSGi Blueprint Container specification that defines a dependency injection framework for dealing with dynamic configuration of OSGi services.

- DDF uses the Apache Aries implementation of Blueprint. More information can be found at [Blueprint](#).

### *Spring DM*

An alternative dependency injection framework. DDF is not dependent on specific dependency injection framework, but Blueprint is recommended.

## **20.5.4. Additional Upstream Dependencies**

DDF is a customized distribution of Apache Karaf, and therefore includes all the capabilities of Apache Karaf. DDF also includes additional FOSS components to provide a richer set of capabilities. Integrated components include their own dependencies, but at the platform level, DDF includes the following upstream dependencies:

### *Apache CXF*

Apache CXF is an open source services framework. CXF helps build and develop services using front end programming APIs, such as JAX-WS and JAX-RS. More information can be found at <http://cxf.apache.org>.

### *Apache Commons*

Provides a set of reusable Java components that extends functionality beyond that provided by the standard JDK (More info available at <http://commons.apache.org>)

### *OSGeo GeoTools*

Provides spatial object model and fundamental geometric functions, which are used by DDF spatial criteria searches. More information can be found at <http://geotools.org/>.

### *Joda Time*

Provides an enhanced, easier to use version of Java date and time classes. More information can be found at <http://joda-time.sourceforge.net>.

For a full list of dependencies, refer to the Software Version Description Document (SVDD).

## **20.6. OSGi Services**

Services consist of:

## An API Bundle

The API, written as Java interfaces, defines the contract of the service and should, to the extent possible, reference only those concrete classes that are loaded by the root classloader. These classes being in the `java.*` packages. These exceptions to the `java.*` rule can be made:

- Extra interfaces can be declared by the API and used as input parameters and return values from API methods.
- Because of their complexity and relative permanence, generated JAXB classes can be exported from an API bundle for use by its consumers.

## *At least one implementation bundle*

As the intent of loosely coupled services is to allow a variety of implementations to be deployed into the container, it is common for there to be more than one concrete implementation of a service. However, that is not a requirement. A single implementation can suffice. It should include a `blueprint.xml` associating the implementation class(es) with interface(s), providing any other wiring of beans, services, and metadata necessary, and registering with the container.

### 20.6.1. Dependency Injection Frameworks

DDF uses resource injection to retrieve and register services to the OSGi registry. There are many resource injection frameworks that are used to complete these operations. Blueprint and Spring DM are both used by DDF.

**NOTE** It is recommended to use Blueprint over Spring DM wherever possible.

There are many tutorials and guides available on the Internet for both of these frameworks.

### 20.6.2. Blueprint - Retrieving a Service Instance

#### *Blueprint example of retrieving and injecting services*

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

    <reference id="ddfCatalogFramework" interface="ddf.catalog.CatalogFramework" />

    <bean class="my.sample.NiftyEndpoint" >
        <argument ref="ddfCatalogFramework" />
    </bean>

</blueprint>
```

Line #	Action
3	Retrieves a Service from the Registry

Line #	Action
6	Instantiates a new object, injecting the retrieved Service as a constructor argument

### 20.6.3. Spring DM - Retrieving a Service Instance

*Spring DM example of retrieving and injecting services*

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:osgi="http://www.springframework.org/schema/osgi">

    <osgi:reference id="ddfCatalogFramework" interface="ddf.catalog.CatalogFramework" />

    <bean class="my.sample.NiftyEndpoint">
        <constructor-arg ref="ddfCatalogFramework" />
    </bean>
</beans>
```

Line #	Action
5	Retrieves a Service from the Registry
8	Instantiates a new object, injecting the retrieved Service as a constructor argument

### 20.6.4. Blueprint - Registering a Service into the Registry

*Creating a bean and registering it into the Service Registry*

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

    <bean id="transformer" class="my.sample.NiftyTransformer"/>

    <service ref="transformer" interface="ddf.catalog.transform.QueryResponseTransformer"
    />

</blueprint>
```

Line #	Action
3	Instantiates a new object
5	Registers the object instance created in Line 3 as a service that implements the <code>ddf.catalog.transform.QueryResponseTransformer</code> interface

## 20.6.5. 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](#). Bundles are Java JAR files that have additional metadata in the **MANIFEST.MF** file that is relevant to an OSGi container.

### *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>DDF DOCS</Bundle-Name>
            <Export-Package />
            <Bundle-SymbolicName>ddf.docs</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.

## 20.6.6. Additional Bundling Resources

- Blueprint
  - <http://aries.apache.org/modules/blueprint.html>
  - <http://www.ibm.com/developerworksopensource/library/os-osgiblueprint/>
  - <http://static.springsource.org/osgi/docs/2.0.0.M1/reference/html/blueprint.html>
- Spring DM
  - <http://www.springsource.org/osgi>
  - Lessons Learned from it-agile (PDF)
  - <http://www.martinlippert.org/events/OOP2010-OSGiLessonsLearned.pdf>
- Creating Bundles
  - <http://blog.springsource.com/2008/02/18/creating-osgi-bundles/>
- Bundle States
  - <http://static.springsource.org/osgi/docs/1.2.1/reference/html/bnd-app-ctx.html>

## 20.6.7. Working with Features

Features XML files group other features and/or bundle(s) for ease of installation/uninstallation.

```

<feature name="catalog-app" install="auto" version="2.9.0"
  description="The DDF Catalog provides a framework for storing, searching, processing,
and transforming information.\nClients typically perform query, create, read, update, and
delete (QCRUD) operations against the Catalog.\nAt 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.::DDF Catalog">
  <feature>platform-app</feature>
  <feature>catalog-core</feature>
  <feature>catalog-core-metricsplugin</feature>
  <feature>catalog-core-sourcemetricsplugin</feature>
  <feature>catalog-transformer-thumbnail</feature>
  <feature>catalog-transformer-metadata</feature>
  <feature>catalog-transformer-xsltengine</feature>
  <feature>catalog-transformer-resource</feature>
  <feature>catalog-rest-endpoint</feature>
  <feature>catalog-opensearch-endpoint</feature>
  <feature>catalog-opensearch-source</feature>
  <feature>catalog-transformer-json</feature>
  <feature>catalog-transformer-atom</feature>
  <feature>catalog-transformer-geoformatter</feature>
  <feature>catalog-transformer-xml</feature>
  <feature>catalog-transformer-tika</feature>
  <feature>catalog-security-plugin</feature>
  <feature>catalog-admin-module-sources</feature>
  <feature>catalog-core-backupplugin</feature>
  <feature>catalog-plugin-jpeg2000</feature>
</feature>

<feature name="catalog-core" install="manual" version="2.9.0"
  description="Catalog Core feature containing the API, third party bundles necessary
to run ddf-core.">
  <feature>catalog-core-api</feature>
  <bundle>mvn:ddf.catalog.core/catalog-core-commons/2.9.0</bundle>
  <bundle>mvn:ddf.catalog.core/catalog-core-camelcomponent/2.9.0</bundle>
  <bundle>mvn:ddf.measure/measure-api/2.9.0</bundle>
  <bundle>mvn:org.codice.thirdparty/picocontainer/1.2_1</bundle>
  <bundle>mvn:org.codice.thirdparty/vecmath/1.3.2_1</bundle> <!-- for GeoTools -->
  <bundle>mvn:org.codice.thirdparty/geotools-suite/8.4_1</bundle>
  <bundle>mvn:org.codice.thirdparty/jts/1.12_1</bundle>
  <bundle>mvn:ddf.catalog.core/catalog-core-federationstrategy/2.9.0</bundle>
  <bundle>mvn:org.codice.thirdparty/lucene-core/3.0.2_1</bundle>
  <bundle>mvn:ddf.catalog.core/ddf-pubsub/2.9.0</bundle>
  <bundle>mvn:ddf.catalog.core/catalog-core-eventcommands/2.9.0</bundle>
  <bundle>mvn:ddf.catalog.core/ddf-pubsub-tracker/2.9.0</bundle>
  <bundle>mvn:ddf.catalog.core/catalog-core-urlresourcereader/2.9.0</bundle>
  <bundle>mvn:ddf.catalog.core/filter-proxy/2.9.0</bundle>
  <bundle>mvn:ddf.catalog.core/catalog-core-commands/2.9.0</bundle>

```

```

<bundle>mvn:ddf.catalog.core/catalog-core-metacardgroomerplugin/2.9.0</bundle>
<bundle>mvn:ddf.catalog.core/metacard-type-registry/2.9.0</bundle>
<bundle>mvn:ddf.catalog.core/catalog-core-standardframework/2.9.0</bundle>
<bundle>mvn:ddf.catalog.core/catalog-core-resourcesizeplugin/2.9.0</bundle>

<configfile finalname="/data/solr/metacard_cache/conf/solrconfig.xml"
>mvn:ddf.platform.solr/platform-solr-server-standalone/2.9.0/xml/solrconfig</configfile>
<configfile finalname="/data/solr/metacard_cache/conf/schema.xml"
>mvn:ddf.platform.solr/platform-solr-server-standalone/2.9.0/xml/schema</configfile>
<configfile finalname="/data/solr/metacard_cache/conf/protwords.txt"
>mvn:ddf.platform.solr/platform-solr-server-standalone/2.9.0/txt/protwords</configfile>
<configfile finalname="/data/solr/metacard_cache/conf/stopwords_en.txt"
>mvn:ddf.platform.solr/platform-solr-server-
standalone/2.9.0/txt/stopwords_en</configfile>
<configfile finalname="/data/solr/metacard_cache/conf/stopwords.txt"
>mvn:ddf.platform.solr/platform-solr-server-standalone/2.9.0/txt/stopwords</configfile>
<configfile finalname="/data/solr/metacard_cache/conf/synonyms.txt"
>mvn:ddf.platform.solr/platform-solr-server-standalone/2.9.0/txt/synonyms</configfile>
</feature>
```

## 20.6.8. Making Sure a Features File Will Display Properly in the Installer

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.

- Be sure that only one feature in the `features.xml` has the `auto-install` tag.

```
install='auto'
```

This is the feature that the installer displays to the user (name, description, version, etc.). It is typically named after the application itself and the description provides a complete application description.

- Be sure that the one feature specified to `auto-install` has a complete list of all of its dependencies in order to ensure the dependency tree can be constructed correctly.

### Auto-starting an Application Feature

Within the `features.xml` file for an application, one feature will have the install attribute set to `auto`. Within this feature, refer to any dependencies of the application as well as any features that should start automatically. 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="catalog-app" install="auto">
    <feature>platform-app</feature>
    <feature>catalog-core</feature>
    <feature>catalog-core-metricsplugin</feature>
    <feature>catalog-core-sourcemetricsplugin</feature>
    <feature>catalog-transformer-thumbnail</feature>
</feature>
```

## 20.7. 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\*hive). 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.

### 20.7.1. 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. Format should be **[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.

## 20.7.2. 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>/opt/release/ddf/distribution/docs/target/doesNotExist</resourcesDir>
<!--
Location of the features.xml file. If it references properties that need to be filtered,
e.g., 2.9.0, it will need to be
filtered by the maven-resources-plugin.
-->
<featuresFile>/opt/release/ddf/distribution/docs/target/classes/features.xml</featuresFil
e>
<!-- Name of the kar file (.kar extension added by default). If not specified, defaults
to docs-2.9.0 -->
<finalName>ddf-ifis-2.9.0</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.

### 20.7.3. 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.

### 20.7.4. 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_INSTALL_DIR>/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_INSTALL_DIR>/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

## 20.8. Migration API

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 import (*coming soon*) and export operations.

**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.

An export operation can be performed through the Command Console or through the Admin Console. When a 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.

### 20.8.1. 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.

### 20.8.2. 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 (e.g., data stored in Solr) should be handled by a different service that implements the `DataMigratable` interface, not by implementors of this class.

### 20.8.3. 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.

## 20.9. Developing CometD Clients for Asynchronous Search and Retrieval

DDF uses the [CometD](#) framework to perform asynchronous operations on the catalog. This is most apparent in the UI application, which is able to perform multiple queries and display them asynchronously as the results are returned. CometD has a [comprehensive manual](#) that details how to interact with and best use the framework with a variety of clients (e.g., javascript, java, perl, and python).

The SearchUI code can be used as reference implementation of how a JavaScript client can be created to integrate with the CometD endpoint. Examples are included below for convenience.

### 20.9.1. General Operations

#### Initialization

CometD offers both Dojo and jQuery bindings that allow CometD to be easily used with those frameworks. For more information on the individual bindings, refer to the [JavaScript Library page](#) in the CometD reference manual. The UI application uses the jQuery library. To initialize the CometD connection, an instance of CometD must be created and configured to point to the server and call the handshake, which creates the network connection. The CometD endpoint is exposed at [/cometd](#), and it is recommended to not use websockets when connecting.

### *Example Initialization with JQuery*

```
// create a reference to the standard cometd object  
Cometd.Comet = $.cometd;  
  
// disable websocket protocol  
Cometd.Comet.websocketEnabled = false;  
  
// configure the location of the cometd endpoint on the server  
Cometd.Comet.configure({  
  url: 'http://server:8181/cometd'  
});  
  
// create network connection to server  
Cometd.Comet.handshake({});
```

### **Subscribe**

A client can asynchronously receive information from the server using subscriptions. Once the client subscribes to a particular topic, it is sent information when the server sends a response on that topic. Subscriptions are performed in the UI to retrieve information for notifications, tasks, and query results.

### *Example Subscription*

```
// subscribe to a topic, calling the function whenever a new message comes in  
var subscription = Cometd.Comet.subscribe("/ddf/notifications/**",  
  function(resp) { ... } );  
  
// unsubscribe from a topic  
Cometd.Comet.unsubscribe(subscription);
```

**NOTE** | Further documentation is available from [CometD/javascript\\_subscribe.html](#).

### **Publish**

Publishing allows clients to send information to the server. This allows the clients to perform queries on the server and perform operations on tasks, such as canceling a download.

### *Example Publish*

```
// perform a query on the server where data contains the search criteria  
Cometd.Comet.publish('/service/query', { data: { ... } });
```

## Publish Notifications

Any application running in DDF can publish notifications that can be viewed by the Search UI or received by another notifications client.

1. Set a properties map containing entries for each of the parameters listed above in the Usage section.
2. Set the OSGi event topic to `ddf/notification/<application-name>/<notification-type>` Notice that there is no preceding slash on an OSGi event topic name, while there is one on the CometD channel name. The OSGi event topic corresponds to the CometD channel this is published on.
3. Post the notification to the OSGi event defined in the previous step.

### *Example for Publishing Notification*

```
Dictionary<String, Object> properties = new Hashtable<String, Object>();
properties.put("application", "Downloads");
properties.put("title", resourceResponse.getResource().getName());
Long sysTimeMillis = System.currentTimeMillis();
properties.put("message", generateMessage(status,
resourceResponse.getResource().getName(), bytes, sysTimeMillis,
detail));
properties.put("user", getProperty(resourceResponse, USER));
properties.put("status", "Completed");
properties.put("bytes", 1024);
properties.put("timestamp", sysTimeMillis);
Event event = new Event("ddf/notification/catalog/downloads",
properties);
eventAdmin.postEvent(event);
```

**NOTE** Further documentation is available at [http://docs.cometd.org/2/reference/javascript\\_publish.html](http://docs.cometd.org/2/reference/javascript_publish.html).

## Retrieving Persisted Notifications and Activities

To retrieve persisted notifications or activities, publish an empty message on the corresponding base channel. This will trigger a response to an awaiting Cometd subscription. Refer to [Developing CometD Clients for Asynchronous Search and Retrieval](#) for instructions on how to publish to a CometD channel.

### *Example: Persistence Retrieval*

```
Cometd.Comet.publish("/ddf/notifications", {});
Cometd.Comet.publish("/ddf/activities", {});
```

### *Asynchronous Query*

**NOTE** An asynchronous query is performed using the CometD Endpoint, which is currently packaged with the DDF Search UI application.

## 20.10. 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.

### 20.10.1. 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.

#### 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 41. Implementations of Subject:*

Classname	Description
<code>ddf.security.impl.SubjectImpl</code>	Extends <code>org.apache.shiro.subject.support.DelegatingSubject</code>

#### 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 42. 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.

## 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.

## 20.10.2. Realms

### Authenticating Realms

`org.apache.shiro.realm.AuthenticatingRealm`

Authenticating Realms are used to authenticate an incoming authentication token and create a Subject on successfully authentication.

Implementations of Authenticating Realms that come with 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 43. Other implementations of the Security API within DDF

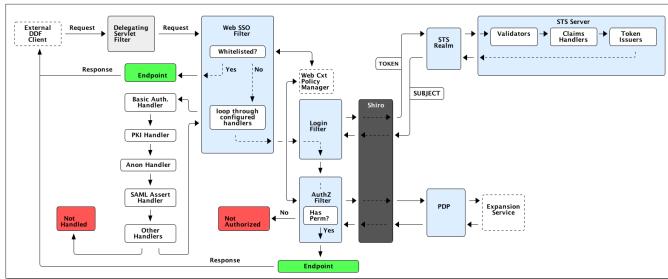
Classname	Description
<code>org.codice.ddf.platform.filter.delegate.DelegateServletFilter</code>	The <a href="#">DelegateServletFilter</a> detects any servlet filters that have been exposed as OSGi services and places them in-order in front of any servlet or web application running on the container.
<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.
<code>org.codice.ddf.security.handler.saml.SAMLAssertionHandler</code>	<p>This handler is executed by the WebSSOFilter for any contexts configured to use it.</p> <p>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.</p> <p>The handler will first check for the existence of a cookie named "org.codice.websso.saml.token" to extract a Base64 + deflate SAML assertion from the request.</p> <p>If an assertion is found it will be converted to a <a href="#">SecurityToken</a>. Failing that, the handler will check for a JSESSIONID cookie that might relate to a current SSO session with the container.</p> <p>If the JSESSIONID is valid, the SecurityToken will be retrieved from the cache in the LoginFilter.</p>
<code>org.codice.ddf.security.handler.basic.BasicAuthenticationHandler</code>	<p>Checks for basic authentication credentials in the http request header.</p> <p>If they exist, they are retrieved and passed to the <a href="#">LoginFilter</a> for exchange.</p>
<code>org.codice.ddf.security.handler.pki.PKIHandler</code>	<p>Handler for PKI based authentication.</p> <p>X509 chain will be extracted from the HTTP request and converted to a <a href="#">BinarySecurityToken</a>.</p>
<code>org.codice.ddf.security.handler.guest.GuestHandler</code>	<p>Handler that allows guest user access via a guest user account.</p> <p>The guest account credentials are configured via the <code>org.codice.ddf.security.claims.guest.GuestClaimsHandler</code>.</p> <p>The <a href="#">GuestHandler</a> also checks for the existence of basic auth credentials or PKI credentials that might be able to override the use of the anonymous user.</p>
<code>org.codice.ddf.security.filter.login.LoginFilter</code>	<p>This filter runs immediately after the WebSSOFilter and exchanges any authentication information found in the request with a Subject via Shiro.</p>
<code>org.codice.ddf.security.filter.authorization.AuthorizationFilter</code>	<p>This filter runs immediately after the <a href="#">LoginFilter</a> and checks any permissions assigned to the web context against the attributes of the user via Shiro.</p>

Classname	Description
<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 focus on making the decision.
<code>ddf.security.pdp.realm.AuthZRealm</code>	<p>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.</p> <p>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.</p> <p>This approach is often easier to configure for those that are not familiar with XACML.</p> <p>Note that no <code>PolicyExtension</code> implementations are provided out of the box.</p>
<code>org.codice.ddf.security.validator.*</code>	<p>A number of STS validators are provided for X.509 (<code>BinarySecurityToken</code>), <code>UsernameToken</code>, SAML Assertion, and DDF custom tokens.</p> <p>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).</p> <p>The authentication realm allows an administrator to restrict which services they wish to use to authenticate users.</p> <p>For example: installing the <code>security-sts-ldaplogin</code> feature will enable a JAAS realm with the name "ldap".</p> <p>This realm can then be specified on any context using the Web Context Policy Manager.</p> <p>That realm selection is then passed via the token sent to the STS to determine which validator to use.</p>

## 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+.

### 20.10.3. 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.

### 20.10.4. 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, `bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=`, is the encrypted

value.

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

```
bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

## 20.10.5. 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.

## 20.10.6. 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.

## 20.10.7. 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

##### Description

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.

##### Motivation

A developer may want to add a custom claims handler to retrieve attributes from an external attribute store.

##### Steps

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>
        <xsschema elementFormDefault="qualified" targetNamespace="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
            <xss:element name="RequestSecurityToken" type="wst:AbstractRequestSecurityTokenType"/>
            <xss:element name="RequestSecurityTokenResponse" type="wst:AbstractRequestSecurityTokenType"/>
            <xss:complexType name="AbstractRequestSecurityTokenType">
                <xss:sequence>
                    <xss:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xss:sequence>
                <xss:attribute name="Context" type="xs:anyURI" use="optional"/>
                <xss:anyAttribute namespace="##other" processContents="lax"/>
            </xss:complexType>
            <xss:element name="RequestSecurityTokenCollection" type="wst:RequestSecurityTokenCollectionType"/>
            <xss:complexType name="RequestSecurityTokenCollectionType">
                <xss:sequence>
                    <xss:element name="RequestSecurityToken" type="wst:AbstractRequestSecurityTokenType" minOccurs="2" maxOccurs="unbounded"/>
                </xss:sequence>
            </xss:complexType>
            <xss:element name="RequestSecurityTokenResponseCollection" type="wst:RequestSecurityTokenResponseCollectionType"/>
            <xss:complexType name="RequestSecurityTokenResponseCollectionType">
                <xss:sequence>
                    <xss:element ref="wst:RequestSecurityTokenResponse" minOccurs="1" maxOccurs="unbounded"/>
                </xss:sequence>
                <xss:anyAttribute namespace="##other" processContents="lax"/>
            </xss:complexType>
        </xsschema>
    </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/RSTR/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>
<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>
                            <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>

```

## **20.10.8. 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.

## **20.10.9. 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">
      <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:RequestSecurityToken>
    </soap:Body>
  </soap:Envelope>
```

```

</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">
>U1QtMTQtYUtmcDYxcFRtS0FxZG1pVDMzOWMtY2FzfGh0dHBz0i8vdG9rZW5pc3N1ZXI60Dk5My9zZXJ2aWNlc
y9T
ZWN1cm10eVRva2VuU2VydmljZQ==</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
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZHl1YXIxGDAWBgNVBAoMD0xvY2toZWVkIE1h
cnRpbjENMAsGA1UECwwESTDRTEPMA0GA1UEAwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGN1
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTA1VT
MRAwDgYDVQQIDA0Bcm16b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2h1ZWQg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDAZjbG1lbnQxHDAaBqkqhkiG9w0BCQEWWDWk0
Y2VAbG1jby5jb20wgZ8wdQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTY17nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVcNEmHndkf9bWU1AoFNaXW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnh1NW9+1xBDAfBgnVHSMEGDAwBQx11VLtYXL
vFGpFdHnh1NW9+1xBDAMBgnVHRMEBTADAQH/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#" />
                </ds:Transforms>
              </ds:SignedInfo>
            </ds:Signature>
          </saml2:Assertion>
        </RequestedSecurityToken>
      </RequestSecurityTokenResponse>
    </RequestSecurityTokenResponseCollection>
  </soap:Body>
</soap:Envelope>
```

```

<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>6wnWbft6Pz5X0F5Q9AG59gcGwLY=</ds:Dige
stValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>h+NvkgXGdQtca3/eKebhAKgG38tHp3i2n5uLLy8xXX
Ig02qyKgEP0FCowp2LiYlsQU9YjKfSwCUbH3WR6jhAv9zj29CE+ePfEny7MeXvgNl3wId+vcHqtI/DGGhhgt0Mb
x/tyX1BhHQUwKRlcHajxHeecwmvV7D85NMdV48tI=</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>MIIDmjCCAw0gAwIBAgIBBDANBgkqhkiG9
w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZH11YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxBcml6b25hMREwDwYDVQQHEwhH
MDQwNzE4MzcxBcml6b25hMREwDwYDVQQHEwhH
b29keWvhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxBMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgxtN9
uB44diIqq3J0zDGfDhGLu6mjpuH01hrKITv42hB0hhmH7LS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lf17IvsVVsNrtAEQlj0Mmw9sxS3SuSQRQ+bD8jq7Uj1hpoF7DdqPv8Kb0C00GwIDAQAB
o4IBBjCCAQIwCQYDVROTBAlwADAsBglghkgBvhCAQ0EHxYdT3B1b1NTTCBHZW5lcmF0ZWQgQ2V
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgnNVHSMEgZ8wgZyAFBcn
en6/j05DzaVw0RwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZH11YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk70cw07gwDQYJKoIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkr0bKpRbQkvkkzcyZl06VrAxRQ+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:holder-of-key">
                    <saml2:SubjectConfirmationData xsi:type=
"saml2:KeyInfoConfirmationDataType">
                        <ds:KeyInfo xmlns:ds=
"http://www.w3.org/2000/09/xmldsig#">
                            <ds:X509Data>
                                <ds:X509Certificate>MIIC5DCCAk2gAwIBAgIJAKj7R
OPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUzEQ

```

MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZH1LYIXgDAwBgNVBAoMD0xvY2t0ZWVkIE1h  
 cnRpbjENMAsGA1UECwwESTRDRTEPMA0GA1UEAwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGN1  
 QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTA1VT  
 MRAwDgYDVQQIDAdBcm16b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2hLZWQg  
 TWFydGlubMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDAZjbG1lbnQxHDAaBqkqhkiG9w0BCQEWDWk0  
 Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAiPhxCBLYE7xfDLcITS9SsPG  
 4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ  
 vvACgGVnEmHndkf9bWU1A0fNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB  
 AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnhLNW9+lxBDAfBgnVHSMEGDAwBQx11VLtYXL  
 vFGpFdHnhLNW9+lxBDAMBgnVHRMEBTADAQH/MA0GCSqSIB3DQEBCwUA4GAHYs20I0K6yVXzyS  
 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/SecurityToke
nService</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:a
c: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</w
sa: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">
          <t:KeyType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
            <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/RSTRC/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">
      <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/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="_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#" />
                    </ds:Transforms>
                  </ds:Reference>
                </ds:SignedInfo>
              </ds:Signature>
            </saml2:Assertion>
          </RequestedSecurityToken>
        </RequestSecurityTokenResponse>
      </RequestSecurityTokenResponseCollection>
    </soap:Body>
  </soap:Envelope>
```

```

<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>Re0qEbGZlyplW5kqiyX0jPnVEA=</ds:Dige
stValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>X5Kzd54PrKI1GVV2XxzCmWFRzHRoybF7hU6zxbEhSL
MR0AWS9R7Me3epq91Xqe0wvIDDbwmE/oJNC7vI0fIw/rqXkx4aZsY5a5nbAs7f+aXF9TGdk82x2eNhNGYpViq0YZJ
fsJ5WSyMtG8w5nRekmDMy9oTLsHG+Y/OhJDEwq58=</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>MIIDmjCCAw0gAwIBAgIBBDANBkgqhkiG9
w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZH11YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxBcml6b25hMREwDwYDVQQHEwhH
b29keWvhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgxtN9
uB44diIqq3J0zDGfDhGLu6mjpuH01hrKITv42hB0hhmH7LS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lf17IvsVVsNrtAEQ1j0Mmw9sxS3SuSQRQX+bD8jq7Uj1hpoF7DdqpV8Kb0C00GwIDAQAB
o4IBBjCCAQIwCQYDVROTBAlwADAsBglghkgBvhCAQ0EHxYdT3B1b1NTTCBHZW5lcmF0ZWQgQ2V
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgnNVHSMEgZ8wgZyAFBcn
en6/j05DzaVw0RwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZH11YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk70cw07gwDQYJKoIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkr0bKpRbQkvkkzcyZl06VrAxRQ+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-
c: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+5Y7hIWIP6SpsEg9XA==</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"/>
        </wst:Claims>
      </wst:RequestSecurityToken>
    </soapenv:Body>
  </soapenv:Envelope>
```

```

<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>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAk
GA1UEBhMCVVMxGDAWBgNVBAoT
D1UuUy4gR29ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLewNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjU00VoXDTE2MDUwNzAwMjU00VowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR29ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLewNQS0kx
EzARBgNVBAAsTCKnPTlRSQUNUT1IxDzANBgNVBAMTBmNsawWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBA0q6L1/jjZ5cyhjhHEbOHr5WQpb0KACYbrsn8lg85LGNoAfcwImr9KBmOxGb
ZCxHYIkW7pJ+kppH8DbbbDMviIvvdkvrAIU0180Brn2wReCBGQ01Imdc3+WzFF2svW75d6wi2ZVd
eMvU015p/pAD/sdIfXmAfyu8+tqtio8KVZGkTnlg3AMzfeSrkci5UHMVwj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpZN2pC85tj9xIoWGH5dFWG1fPcYRgzGPxsbyiG0ylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAh{i}WRGyPlY08w1LzixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNx43NZLBBInDjDp1NZJoMB0GA1UdDgQWBBRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2NybC5nZHMu
bm10LmRpc2EubWlsL2Nybc9ET0RKSVDQ0FfMjcuY3JsMIGUoIGRoIG0hoGLbGRhcDovL2NybC5n
ZHMubm10LmRpc2EubWlsL2NuJTNkRE9EJTIwSk1UQyUyMENBLTI3JTJjb3U1M2RQS0k1MmNvdSUz
ZERvRCUyY281M2RVL1MuJTIwR29ZXJubWVudCUyY2M1M2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN002JpbmFyeTAjBjNVHSAEHDAAmAsgCWCgsAF1AgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzAChjFodHRw0i8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRw0i8vb2NzcC5uc24wLnJjdMubm10LmRp
c2EubWlsMA0GCSqGSIB3DQEBBQUAA4IBAQCGUJPGe4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSgxnT4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIHr
hwtE3xo6qQRRWl03/c1C4RnTev1crFVJQVBF3yfpRu8udJ2SOGdqU0vjUSu1h7aMkYJMHlu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+jmnpbwq+XsCwg1LjC4dmbjKv9s4QK+/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/RSTR/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+UuYlGOuTwDYsUPcS2qI0nAgMD1VsDP7y1fDJxeqsq7HYhFKsnqRfebMM4WL
H1D/lJ4rD4U0+i9l3tuiHml7SN24WM1/b0qfDUCoDqmwg8afUJ3r4vmTNPxfwf0ss8BZ/80DgZzm08ndlKxDfvcN7
OrExbV/3/45JwF/MMPZoqv12MJGfx56E9fErJNuzezpWnRqP01WPxyffKMA1VaB9zF6gvVnUqcW2k/Z8X9lN705jo
uBI281ZnIfsIPuBJERFtYNVDHsIXM1pJnrY6FlKIaOsisi55LQu3Ruirl/n82pU7BT5aWtxwrn7akBg==
</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
    <ds:X509Certificate>MIIFHTCCBAwgAwIBAgICJe8wDQYJKoZIhvcNAQEFBQ
AwXDELMAkGA1UEBhMCVVMxGDAwBgNVBAoT
D1UuUy4gR29ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgnVBAMTDkRP
RCBKSVDIENBLTI3MB4XDTEzMDUwNzAwMjYzN1oXDTE2MDUwNzAwMjYzN1owbjELMAkGA1UEBhMC
VVMxGDAwBgNVBAoTD1UuUy4gR29ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxFDASBgNVBAMTC3Rva2VuAXNzdWVyMIIBIjANBgkqhkiG9w0B
AQEFAOCAQ8AMIIBCgKCAQEAx01/U4M1wG+wL1JxX2RL1glj101fkJXMK3Kft3zD//N8x/Dcwwvs
ngCQjXrV6YhbB2V7scHwnThPv3RSwYYi062z+g6ptfBbKGGBLSZ0zLe3fyJR4Rxb1FKsELFgPHfX
vgUHS/keG5uSRk9S/0kqps/yxKB7+Z1xeFxsIz5QywXvBpMiXtc2zF+M7BsbSIDSx5LcPcDFBwjF
c66rE3/y/25VMht9EZX1QoKr7f8rWD4xgd5J6DYMFWEcniCz4BDJH9sfTw+n1P+CYgrhwsIWGqxt
cDME9t6SWR3GLT4Sdtr8ziIM5uUtehPIV3rVC3/u23JbYEeS8mpnp0bxt5eHQIDAQABo4IB1TCC
AdEwHwYDVR0jBBgwFoAUIxQ0IE1fLjc1ksEGwCOMOmU1kmgwHQYDVR00BBYEFGBjdkdey+bMHMhC
Z7gwiQ/mJf5VMA4GA1UdDwEB/wQEAvIFoDCB2gYDVR0fBIHSMIHMDagNKAyhjBodHRw0i8vY3Js
Lmdkcy5uaXQuZGlzYS5taWwvY3JsL0RPREpJVENDQV8yNy5jcmwwgZSggZGggY6GgYtsZGFw0i8v
Y3JsLmdkcy5uaXQuZGlzYS5taWwvY241M2RET0Q1mjBKSVDJTIwQ0EtMjclMmNvdSUzzFBLSUy
Y291JTNkRG9EJTjbyUzZFuuUy41MjBhb3Zlcm5tZW50JTjYyUzZFTP2NlcnRpZmljYXRlcmV2
b2NhG1vbmxpc3Q7YmluyXJ5MCMGA1UdIAQcMBowCwYJYIZIAWUCAQsFMAAsGCWCGSAFlAgELEjB9
BggRBeFBQcBAQRxMG8wPQYIKwYBBQUHMAKGWh0dHA6Ly9jcmwwuZ2RzLm5pdC5kaXNhLm1pbC9z
aWduL0RPREpJVENDQV8yNy5jZXIwLgYIKwYBBQUHMAKGWh0dHA6Ly9vY3NwLm5zbjAucmN2cy5u
aXQuZGlzYS5taWwvDQYJKoZIhvcNAQEFBQADggEBAlHZQTINU3bMpJ/PkwTYLWPmwCqAYgEUzSYx
bNcVY5MWD8b4XCdw5nM3GnFl0qr4IrHeyy0zsEbIebTe3bv0l1pHx0Uyj059nAhx/AP8DjVtuRU1
/Mp4b6uJ/4yaoMjIGceqBzHqhHIJinG0Y2azua7eM9hVbWZsa912ihbiupCq22mYuHFP7NUNzBvV
j03YUcsy/sES5sRx9Rops/CBN+LUUY0dJ0xYWxo8oAbtF8ABE5ATLAwqz4ttsToKPUYh1sdx5Ef
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>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcN
AQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT
D1UuUy4gR29ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIEBLT13MB4XDTEzMDUwNzAwMjU00VoXDTE2MDUwNzAwMjU00VowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR29ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCKNPTlRSQUNUT1IxDzANBgNVBAMTBmNsawWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBA0q6L1/jjZ5cyjhjHEb0Hr5WQpb0KACYbrsn8lg85LGNoAfcwImr9KBmOxGb
ZCxHYIhkW7pJ+kpppH8bbbviIvvdkvrAIU0180BRn2wReCBGQ01Imdc3+WzFF2svW75d6wi2ZVd
eMvUO15p/pAD/sdIfXmAfuy8+tqt08KVZGkTnlg3AMzfeSrkcis5UHMVWj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpZN2pC85tj9xIoWGH5dFWG1fPcYRgzGPxsbyiG0y1bJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAhWRGyPlY08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNXY43NZLBBlnDjpLNZJoMB0GA1UdDgQWBBRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2Nybc5nZHMu
bm10LmRp2EubWls2Nybc9ET0RKSVRDQ0FfMjcuY3JsMIGUoIGRoIG0hoGLbGRhcDovL2Nybc5n
ZHMubm10LmRp2EubWls2NuJTNkRE9EJTIwSk1UQyUyMENBLTI3JTjb3U1M2RQS0k1MmNvdSUz
ZERvRCUyY281M2RVLLMuJTIwR29ZXJubWVudCUyY2M1M2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN002JpbmFyeTAjBjNVHSAEHDAAmAsGCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUBFzAChjFodHRw0i8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVRDQ0FfMjcuY2VymC4GCCsGAQUBFzABhiJodHRw0i8vb2NzcC5uc24wLnJjdnuMubm10LmRp
c2EubWlsMA0GCSqGSIB3DQEBBQUAA4IBAQCGUJPgh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSzxnT4Q9X8WwalqVY0fNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIHr
hwtE3xo6qQRRWl03/c1C4RnTev1crFVJQVF3yfpRu8udJ2S0GdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+jmnpbwq+Xscxg1LjC4dmbjKVr9s4QK+/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>

```

## 20.11. 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.

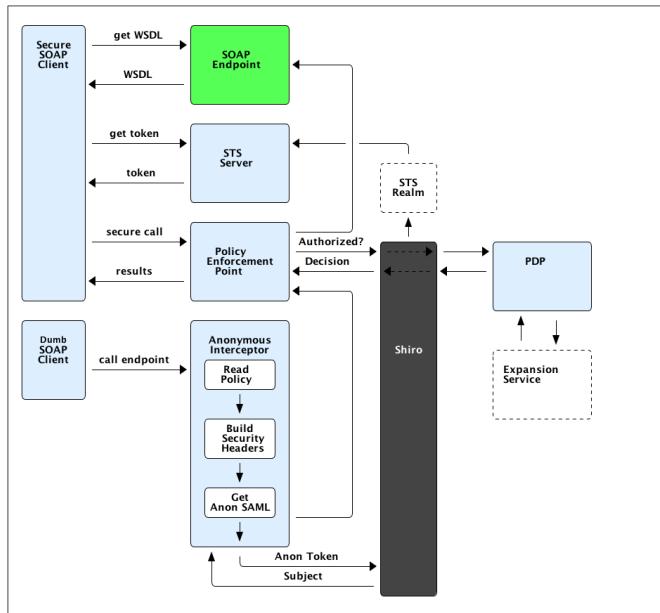
### Key (Attribute)

The examples below use the following collection of rule sets:

## Key (Attribute)

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.

## 20.12. Securing SOAP



### 20.12.1. 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.

### 20.12.2. Dumb SOAP Client

If calling 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.

# 21. Extending DDF Admin

Version: 2.9.0

This section supports developers creating extensions of the existing DDF Admin application.

## 21.1. Whitelist

The following packages have been exported by the DDF 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`

# 22. Extending DDF Catalog

Version: 2.9.0

The DDF Catalog provides a framework for storing, searching, processing, and transforming information. Clients typically perform 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.

This guide supports developers creating extensions of the existing framework.

## 22.1. Whitelist

The following packages have been exported by the DDF 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.metacardtype`
- `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.metocard.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

## 22.2. Catalog Architecture

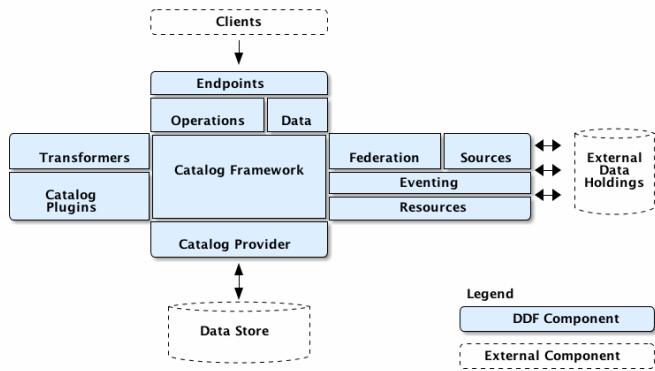


Figure 8. Catalog Architecture

## 22.3. Catalog Application Services

As an OSGi system, DDF does Intra-module conversations via services. The following summarizes DDF

internal services within the Catalog application.

### 22.3.1. Catalog Framework

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.

### 22.3.2. Sources

A source is a system consisting of a catalog containing Metacards.

### 22.3.3. CatalogProvider

The Catalog Provider is an API used to interact with data providers, such as file systems or databases, to query, create, update, or delete data. The provider also translates between DDF objects and native data formats.

### 22.3.4. ConnectedSource

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.

### 22.3.5. FederatedSource

A Federated Source is a remote source that can be optionally included or excluded from queries.

### 22.3.6. Plugins

Plugins are additional tools to use to add additional business logic at certain points, depending on the type of plugin. Plugins can be designed to run before or after certain processes. They are often used for validation, optimization, or logging.

#### "Pre-" Plugins

These plugins are executed before an action is taken.

Plugin	Description
Pre-IngestPlugin	Performs any changes to a resource prior to ingesting it.
Pre-Query Plugin	Performs any changes to query before executing.
Pre-Resource Plugin	Performs any changes to a resource associated with a metocard prior to download.

<b>Plugin</b>	<b>Description</b>
Pre-Subscription Plugin	Performs any changes before creating a subscription.
Pre-Delivery Plugin	Performs any changes before delivered a subscribed event.

### “Post-“ Plugins

<b>Plugin</b>	<b>Description</b>
Post-Ingest Plugin	Performs actions after ingest is completed.
Post-Query Plugin	Performs any changes to response after query completes.
Post-Get Resource Plugin	performs any changes to a resource after download

### 22.3.7. Transformers

Transformers are used to alter the format of a resource or its metadata to or from the catalog’s metocard format

<b>Transformer</b>	<b>Description</b>
Input Transformers	create metacards from input.
Metocard Transformers	translates a metocard from catalog metadata to a specific data format.
Query Response Transformers	translates a list of Result objects to a desired format.

## 22.4. Catalog Development Fundamentals

This section introduces the fundamentals of working with the Catalog API the OGC Filter for Queries.

### 22.4.1. 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 to base future development.

## 22.4.2. Use of the Whiteboard Design Pattern

The DDF 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.

## 22.4.3. Working with Queries

Clients use `DDF.catalog.operation.Query` objects to describe which metacards are needed from Sources. Query objects have two major components:

- Filter
- 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.

### 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.

## Creating a query

The easiest way to create a Query is to use `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 1*

```
/*
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);
```

## 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.

## 22.5. Working with Filters

An OGC Filter is a Open Geospatial Consortium (OGC) standard (<http://www.opengeospatial.org/standards/filter>) that describes a query expression in terms of Extensible Markup Language (XML) and key-value pairs (KVP). The DDF Catalog Framework does not use the XML representation of the OGC Filter standard. DDF instead utilizes the Java implementation provided by Geotools (<http://geotools.org/>). 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
```

### 22.5.1. Using Filters

### 22.5.2. FilterBuilder API

To abstract developers from the complexities of working with the `Filter` interface directly and implementing the DDF Profile of the `Filter` specification, the DDF 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.

### 22.5.3. 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 all 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 be match (Boolean NOT).

#### Attribute

`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.

## XPath

`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);
```

## Directly Implementing the Filter (Advanced)

**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

## 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.

## Creating Filters

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](#).

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.

### Example Creating-Filters-1

#### *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(Metocard.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.

### Example Creating-Filters-2

## 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 metocard

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).

### PropertyIsLike Special Characters

Character	Description
Wildcard	Matches zero or more characters.
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.

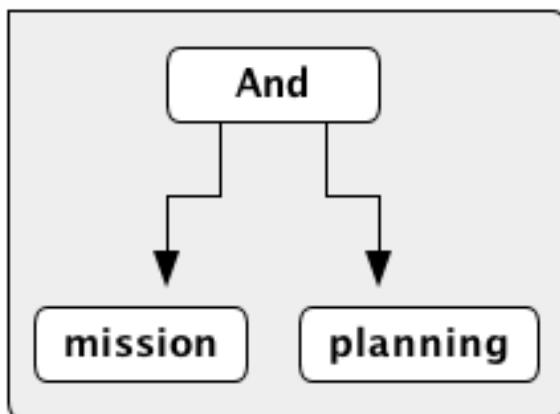
### Example Creating-Filters-3

*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 Example Creating-Filters-3

Filters used in DDF can always be represented in a tree diagram.



#### XML View of Example Creating-Filters-3

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 Operation

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(Metacard.ANY_TEXT))),
    filterFactory.literal("")),
  searchPhrase,
  wildcardChar,
  singleChar,
  escapeChar,
  isCaseSensitive);

QueryImpl query = new QueryImpl(fuzzyFilter);
```

## 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 interface `org.opengis.filter.FilterVisitor`.

The `FilterVisitor` uses the <http://www.oodesign.com/visitor-pattern.html>[Visitor pattern]. 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.

## Examples

### 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`.

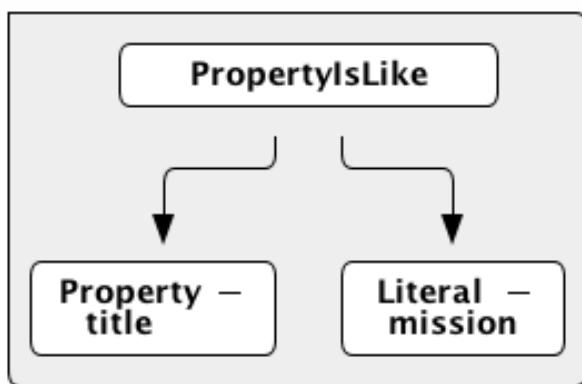


Figure 9. Figure Parsing-Filters1

## 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').
```

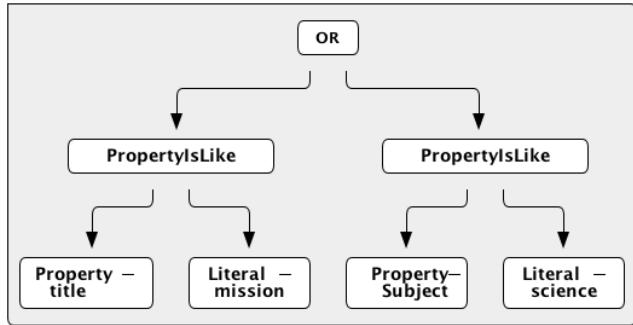


Figure 10. Figure Parsing-Filters2

### FilterAdapter/Delegate Process for Figure Parsing-Filters2

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-Filters2**

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;
    ...
}
```

## **22.5.4. Filter Profile**

### **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

### Metocard Attribute To Type Mapping

The filter profile maps filters to metocard types. The following table displays the common metocard attributes with their respective types for reference.

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.

PropertyIs	Between	EqualTo	GreaterThan	GreaterThanOrEqual	OrEqualTo	LessThan	LessThanOrEqual	OrEqualTo	Like	NotEqualTo	Null
BINARY_TYPE		X									
BOOLEAN_TYPE		X									
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

The following table describes each comparison operator.

Table 44. Comparison Operators

Operator	Description
PropertyIsBetween	Lower Property Upper

<b>Operator</b>	<b>Description</b>
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 45. Logical Operators*

	<b>And</b>	<b>Not</b>	<b>Or</b>
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. The following table displays the supported temporal operators.

	<b>After</b>	<b>AnyIn teracts</b>	<b>Before</b>	<b>Begins</b>	<b>Begun By</b>	<b>Durin g</b>	<b>Ended By</b>	<b>Meets</b>	<b>MetBy</b>	<b>Overla ppedB y</b>	<b>TCont ains</b>
DATE_ TYPE	X		X			X					

The following table describes each temporal operator. Literal values can be either date instants or date periods.

<b>Operator</b>	<b>Description</b>
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. The following table displays the supported spatial operators.

BBox	Beyond	Contains	Crosses	Disjoint	Equals	DWithin	Intersects	Overlaps	Touches	Within
GEO_TYPE		X	X	X	X		X	X	X	

The following table describes each spatial operator. Geometries are usually represented as Well-Known Text (WKT).

Operator	Description
Beyond	Property geometries beyond given distance of Literal geometry
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 somewhere overlaps Literal geometry interior
Touches	Property geometry touches but does not overlap Literal geometry
Within	Property geometry completely contains Literal geometry

### 22.5.5. Commons-DDF Utilities

The `commons-DDF`bundle, located in <DDF_HOME_SOURCE_DIRECTORY>/common/commons-DDF``, provides utilities and functionality commonly used across other DDF components, such as the endpoints and providers.

### 22.5.6. Noteworthy Classes

## 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.

## 22.5.7. Working with Settings

DDF provides the ability to obtain DDF settings/properties. The `DdfConfigurationWatcher` will provide an update of properties to watchers. For example, if the port number changes, the `DDF_PORT` property value will be propagated to the watcher(s) in the form of a map.

## 22.5.8. Property Values

To obtain the property values, complete the following procedure.

1. Import and implement the `DDF.catalog.util.DdfConfigurationWatcher` interface.

*Implement DdfConfigurationWatcher*

```
public class SettingsWatcher implements DdfConfigurationWatcher
```

1. Get properties map and search for the property.

*Handle Properties*

```
public void DDFConfigurationUpdated( Map properties )
{
    //Get property by name
    Object value = properties.get( DdfConfigurationManager.DDF_HOME_DIR );
    if ( value != null )
    {
        this.DDFHomeDir = value.toString();
        logger.debug( "DDFHomeDir = " + this.DDFHomeDir );
    }
}
```

1. Export the watcher class as a service in the OSGi Registry. The example below uses the Blueprint dependency injection framework to add this watcher to the OSGi Registry. The `DDF.catalog.DdfConfigurationManager` will search for `ConfigurationWatcher`(s) to send properties updates.

## Blueprint Example of Export

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0">

<!-- create the bean -->
<bean id="SettingsWatcher" class="DDF.catalog.SettingsWatcher">
    <cm:managed-properties
        persistent-id="DDF.catalog.SettingsWatcher"
        update-strategy="container-managed" />
</bean>

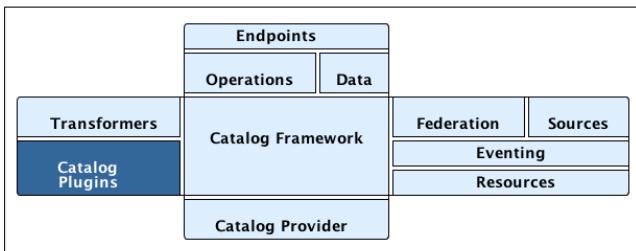
<!-- export the bean in the service registry as a DdfConfigurationWatcher -->
<service ref="SettingsWatcher" interface="DDF.catalog.util.DdfConfigurationWatcher">
</service>

</blueprint>
```

1. Import the DDFpackages to the bundle's manifest for run-time (in addition to any other required packages). **Import-Package:** `DDF.catalog, DDF.catalog.util, DDF.catalog.*`
2. Deploy the packaged service to DDF (refer to the Working with OSGi - Bundles section).

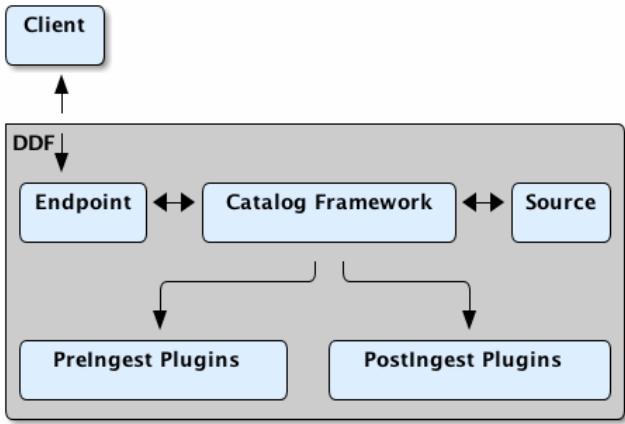
## 22.6. Extending Catalog Plugins

The Catalog Framework calls Catalog Plugins to process requests and responses as they enter and leave the Framework.



### 22.6.1. Existing Plugins

#### Pre-Ingest Plugin



*Figure 11. Ingest Plugin Flow*

## Using

Pre-Ingest plugins are invoked before an ingest operation is sent to a Source. This is an opportunity to take any action on the ingest request, including but not limited to:

- validation
- logging
- auditing
- optimization
- security filtering

## 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.

## Invocation

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.

## Metocard Groomer

The Metocard Groomer Pre-Ingest plugin makes modifications to [CreateRequest](#) and [UpdateRequest](#) metacards.

This plugin makes the following modifications when metacards are in a [CreateRequest](#):

- Overwrites the [Metocard.ID](#) field with a generated, unique, 32 character hexadecimal value
- Overwrites the [Metocard.CREATED](#) date with a current time stamp
- Overwrites the [Metocard.MODIFIED](#) date with a current time stamp

The plugin also makes the following modifications when metacards are in an [UpdateRequest](#):

- If no value is provided for [Metocard.ID](#) in the new metocard, it will be set using the [UpdateRequest ID](#) if applicable.
- If no value is provided, sets the [Metocard.CREATED](#) date with the [Metocard.MODIFIED](#) date so that the [Metocard.CREATED](#) date is not null.
- Overwrites the [Metocard.MODIFIED](#) date with a current time stamp.

### **Installing and UnInstalling**

This plugin can be installed and uninstalled using the normal processes described in the Configuring DDF section.

### **Configuring**

No configuration is necessary for this plugin.

### **Using**

Use this pre-ingest plugin as a convenience to apply basic rules for your metacards.

### **Known Issues**

None

### **Post-Ingest Plugin**

#### **Using**

Post-ingest plugins are invoked after data has been created, updated, or deleted in a Catalog Provider.

#### **Failure Behavior**

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a [PluginExecutionException](#) will be thrown.

#### **Invocation**

Because the event has already occurred and changes from one post-ingest plugin cannot affect others, all Post-Ingest plugins are invoked in parallel and no priority is enforced.

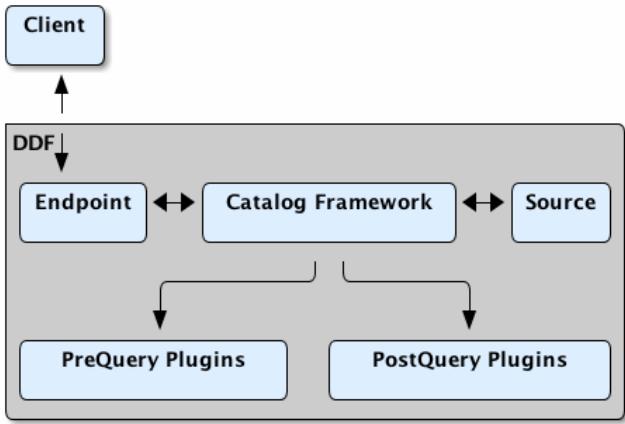


Figure 12. *QueryPlugin Flow*

## Pre-Query Plugin

### Using

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

### 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.

### Invocation

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.

## Post-Query Plugin

## **Using**

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

## **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.

## **Invocation**

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.

### **22.6.2. 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.

#### **Installing and UnInstalling**

This feature can be installed and uninstalled using the normal processes described in the Configuring DDF section.

#### **Configuring**

No configuration is necessary for this plugin.

## **Using**

Use this post-query plugin as a convenience to return query results with accurate resource sizes for cached products.

## **Known Issues**

None

## 22.6.3. Other Types of Plugins

### Pre-Get Resource Plugin

#### Using

Pre-get 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

#### 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.

#### Invocation

Pre-get 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.

### Post-Get Resource Plugin

#### Using

Post-get 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

#### 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.

## Invocation

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.

## Pre-Subscription Plugin

### Using

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

## 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.

## Invocation

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.

## Examples

DDF includes a pre-subscription plugin example in the SDK that illustrates how to modify a subscription's filter. This example is located in the DDF trunk at [sdk/sample-plugins/DDF/sdk/plugin/presubscription](#).

## **Pre-Delivery Plugin**

### **Using**

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

### **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.

### **Invocation**

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.

### **22.6.4. Developing a Catalog Plugin**

### **22.6.5. Policy Plugin**

### **Using**

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.

### **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.

### **22.6.6. Access Plugin**

## Using

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.

## 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.

### 22.6.7. Developing a Catalog Plugin

Plugins extend the functionality of the Catalog Framework by performing actions at specified times during a transaction. Plugins can be *Pre-Ingest*, *Post-Ingest*, *Pre-Query*, *Post-Query*, *Pre-Subscription*, *Pre-Delivery*, *Pre-Resource*, or *Post-Resource*. By implementing these interfaces, actions can be performed at the desired time.

#### Create New Plugins

##### Implement Plugin Interface

The following types of plugins can be created:

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 Metocard 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 MetocardValidityFilterPlugin
Access	<code>DDF.catalog.plugin.AccessPlugin</code>	Runs directly after the PolicyPlugin	An examples are the FilterPlugin and OperationPlugin

## 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. Import DDF packages.
  - b. Register plugin class as service to OSGi registry.
4. Deploy to DDF.

**TIP** Refer to the Javadoc for more information on all Requests and Responses in the `ddf.catalog.operation` and `ddf.catalog.event` packages.

## Pre-Ingest

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;`
- `public UpdateRequest process(UpdateRequest input) throws PluginExecutionException;`

- `public DeleteRequest process(DeleteRequest 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="[[SamplePreIngestPlugin]]" interface="DDF.catalog.plugin.PreIngestPlugin" />`

## Post-Ingest

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" />`

## Pre-Query

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" />`

## Post-Query

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" />
```

## Pre-Delivery

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-Subscription

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-Resource

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" />
```

## Post-Resource

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" />
```

## Policy

1. Create a Java class that implements `PolicyPlugin`.

`public class SamplePolicyPlugin implements DDF.catalog.plugin.PolicyPlugin`

2. Implement the required methods.

- `PolicyResponse processPreCreate(Metocard input, Map<String, Serializable> properties) throws StopProcessingException;`
- `PolicyResponse processPreUpdate(Metocard input, Map<String, Serializable> properties) throws StopProcessingException;`

`PolicyResponse processPreDelete(String attributeName, List<Serializable> attributeValues,`

- `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" />
```

## Access

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" />
```

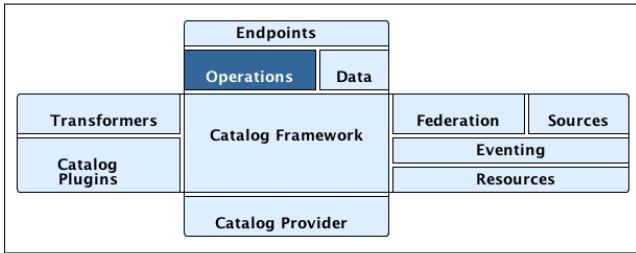
## 22.7. Extending Operations

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

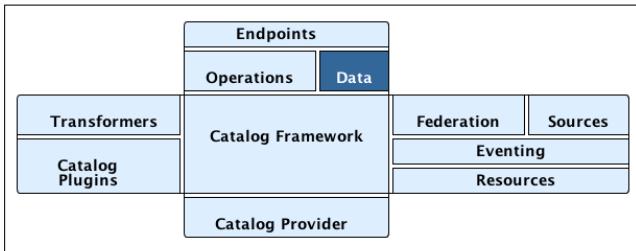
is an associated request/response pair, e.g., the [QueryRequest](#) and [QueryResponse](#) pair for the Catalog Framework's query operation.

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.



### 22.7.1. Extending Data and Metadata Basics

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](#).



### 22.7.2. Metocard

A single instance of metadata in the Catalog (an instance of a metocard type) which generally contains metadata providing a title for the product and describing a product's geo-location, created and modified dates, owner or producer, security classification, etc.

### 22.7.3. Metocard Type

#### 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.

## Default Metocard Type and Attributes

Most metacards within the system are created using with the default metocard 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.

### Required Attributes

DDF.catalog.data.Metocard Constant	Attribute Name	Attribute Format	Description
CONTENT_TYPE	<code>metadata-content-type</code>	STRING	Attribute name accessing for the metadata content type of a Metocard.
CONTENT_TYPE_VERSION	<code>metadata-content-type-version</code>	STRING	Attribute name for the version of the metadata content accessing type of a Metocard.
CREATED	<code>created</code>	DATE	Attribute name for accessing the date/time <b>this Metocard</b> was created.
EFFECTIVE	<code>effective</code>	DATE	Attribute name for accessing the date/time of <b>the product</b> represented by the Metocard.
EXPIRATION	<code>expiration</code>	DATE	Attribute name for accessing the date/time the Metocard is no longer valid and could be removed.
GEOGRAPHY	<code>location</code>	GEOMETRY	Attribute name for accessing the location for this Metocard.
ID	<code>id</code>	STRING	Attribute name for accessing the ID of the Metocard.

<b>DDF.catalog.data.Metacard Constant</b>	<b>Attribute Name</b>	<b>Attribute Format</b>	<b>Description</b>
METADATA	metadata	XML	Attribute name for accessing the XML metadata for this Metocard.
MODIFIED	modified	DATE	Attribute name for accessing the date/time this Metocard was last modified.
RESOURCE_SIZE	resource-size	STRING	Attribute name for accessing the size in bytes of the product this Metocard represents.
RESOURCE_URI	resource-uri	STRING	Attribute name for accessing the URI reference to the product this Metocard represents.
TARGET_NAMESPACE	metadata-target-namespace	STRING	Attribute name for the target namespace of the accessing metadata content type of a Metocard.
THUMBNAIL	thumbnail	BINARY	Attribute name for accessing the thumbnail image of the product this Metocard represents. The thumbnail must be of MIME Type <code>image/jpeg</code> and be less than 128 kilobytes.
TITLE	title	STRING	Attribute name for accessing the title of the Metocard.

**NOTE** It is highly recommended when referencing a default attribute name to use the `DDF.catalog.data.Metacard` constants whenever possible.

**WARNING** Every Source should at the very least return an ID attribute according to Catalog API. Other fields might or might 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 Catalog Providers support extensible metacards. Nevertheless, each Catalog Provider should at least have support for the default [MetocardType](#); i.e., it should be able to store and query on the attributes and attribute formats specified by the default metocard type. Consult the documentation of the Catalog Provider in use for more information on its support of extensible metacards.

### Metocard Extensibility

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 its 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.

### 22.7.4. Metocard Type Registry

**WARNING**

The [MetocardTypeRegistry](#) 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 [MetocardTypeRegistry](#) allows DDF components, primarily CatalogProviders and Sources, to make available the [MetocardTypes](#) that they support. It maintains a list of all supported [MetocardTypes](#) in the [CatalogFramework](#), so that other components such as Endpoints, Plugins, and Transformers can make use of those [MetocardTypes](#). The [MetocardType](#) 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 [MetocardTypeRegistry](#) to find a corresponding [MetocardType](#). The discovered [MetocardType](#) will then be used to help the endpoint populate a metocard based on the specified attributes in the [MetocardType](#). By doing this, all the incoming metadata elements can then be available for processing, cataloging, and searching by the rest of the [CatalogFramework](#).

`MetocardTypes` should be registered with the `MetocardTypeRegistry`. The `MetocardTypeRegistry` makes those `MetocardTypes` available to other DDF `CatalogFramework` components. Other components that need to know how to interpret metadata or metacards should look up the appropriate `MetocardType` from the registry. By having these `MetocardTypes` available to the `CatalogFramework`, these components can be aware of the custom attributes.

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

```
<bean id="sampleComponent" class="DDF.catalog.SampleComponent">
    <argument ref="metocardTypeRegistry" />
</bean>

<!-- Access MetocardTypeRegistry -->
<reference id="metocardTypeRegistry" interface="DDF.catalog.data.MetocardTypeRegistry"/>
```

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

Typically, new `MetocardTypes` 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 `MetocardType` based on a parameter in the incoming metadata. Once the appropriate `MetocardType` is discovered and obtained from the registry, the component will know how to translate incoming raw metadata into a DDF Metacard.

## Attribute

A single field of a metacard, an instance of an attribute type. Attributes are typically indexed for searching by a Source or Catalog Provider.

### Attribute Type

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.

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&lt;Array&gt;of byte</code> .

AttributeFormat	Description
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

## 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 **Attribute**'s 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 random [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.

Name	Type	Description
BASIC_METACARD	MetocardType	representing all required Metocard Attributes
BINARY_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">AttributeType.AttributeFormat.BINARY</a> .
BOOLEAN_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">AttributeType.AttributeFormat.BOOLEAN</a> .
DATE_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">AttributeType.AttributeFormat.DATE</a> .
DOUBLE_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">AttributeType.AttributeFormat.DOUBLE</a> .
FLOAT_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">AttributeType.AttributeFormat.FLOAT</a> .

Name	Type	Description
GEO_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.GEOMETRY</a> .
INTEGER_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.INTEGER</a> .
LONG_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.LONG</a> .
OBJECT_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.OBJECT</a> .
SHORT_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.SHORT</a> .
STRING_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.STRING</a> .
XML_TYPE	AttributeType	A Constant for an AttributeType with <a href="#">Attribute Type.AttributeFormat.XML</a> .

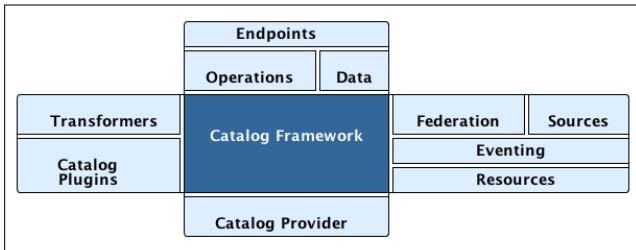
## 22.8. Extending Catalog Framework

This section describes the core components of the Catalog app and Catalog Framework. The Catalog Framework wires all Catalog components together.

It is responsible for routing Catalog requests and responses to the appropriate target.

Endpoints send Catalog requests to the Catalog Framework. The Catalog Framework then invokes Catalog Plugins, Transformers, and Resource Components as needed before sending requests to the intended destination, such as one or more Sources.

The Catalog Framework functions as the routing mechanisms between all catalog components. It decouples clients from service implementations and provides integration points for Catalog Plugins and convenience methods for Endpoint developers.



## 22.8.1. Included Catalog Frameworks

### 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.

### Standard Catalog Framework

The Standard Catalog Framework provides the reference implementation of a Catalog Framework that implements all requirements of the DDF CatalogAPI. `CatalogFrameworkImpl` is the implementation of the DDF Standard Catalog Framework.

### Installing and Uninstalling

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

#### Configurable Properties

*Table 46. Catalog Standard Framework*

Property	Type	Description	Default Value	Required
<code>fanoutEnabled</code>	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	yes

Property	Type	Description	Default Value	Required
<code>productCacheDirectory</code>	String	Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Catalog Framework will attempt to create those directories. Out of the box (without configuration), the product cache directory is <code>&lt;INSTALL_DIR&gt;/data/product-cache</code> . If a relative path is provided it will be relative to the <code>&lt;INSTALL_DIR&gt;</code> .  It is recommended to enter an absolute directory path such as <code>/opt/product-cache` in Linux or 'C:\product-cache</code> in Windows."	(empty)	no
<code>cacheEnabled</code>	Boolean	Check to enable caching of retrieved products to provide faster retrieval for subsequent requests for the same product.	false	no
<code>delayBetweenRetryAttempts</code>	Integer	The time to wait (in seconds) between each attempt to retry retrieving a product from the Source.	10	no
<code>maxRetryAttempts</code>	Integer	The maximum number of attempts to try and retrieve a product from the Source.	3	no
<code>cachingMonitorPeriod</code>	Integer	The number of seconds allowed for no data to be read from the product data before determining that the network connection to the Source where the product is located is considered to be down.	5	no
<code>cacheWhenCanceled</code>	Boolean	Check to enable caching of retrieved products even if client cancels the download.	false	no

<b>Managed Service PID</b>	<code>DDF.catalog.CatalogFrameworkImpl</code>
Managed Service Factory PID	N/A

## Using

The Standard Catalog Framework is the core class of DDF. It provides the methods for query, create, update, delete, and resource retrieval (QCRUD) 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 to create, update, and delete catalog entries are required.
- 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.

## Implementation Details

### Exported Services

Registered Interface	Service Property	Value
<code>DDF.catalog.federation.FederationStrategy</code>	<code>shortname</code>	<code>sorted</code>
<code>org.osgi.service.event.EventHandler</code>	<code>event.topics</code>	<code>DDF/catalog/event/CREATED,</code> <code>DDF/catalog/event/UPDATED,</code> <code>d`df/catalog/event/DELETED`</code>
<code>DDF.catalog.CatalogFramework</code>		
<code>org.codice.DDF.configuration.ConfigurationWatcher</code>		
<code>DDF.catalog.event.EventProcessor</code>		
<code>DDF.catalog.plugin.PostIngestPlugin</code>		

## 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
DDF.catalog.plugin.PolicyPlugin	optional	true
DDF.catalog.plugin.AccessPlugin	optional	true
DDF.catalog.resource.ResourceReader	optional	true
DDF.catalog.source.CatalogProvider	optional	true
DDF.catalog.source.ConnectedSource	optional	true
DDF.catalog.source.FederatedSource	optional	true
DDF.cache.CacheManager		false
org.osgi.service.event.EventAdmin		false

### 22.8.2. Known Issues

None

## 22.9. Catalog Fanout Framework

The Fanout Catalog Framework ([fanout-catalogframework bundle](#)) provides an implementation of the Catalog Framework that acts as a proxy, federating requests to all available sources. All requests are executed as federated queries and resource retrievals, allowing the fanout site to be the sole site exposing the functionality of all of its Federated Sources. The Fanout Catalog Framework is the implementation of the Fanout Catalog Framework.

The Fanout Catalog Framework provides the capability to configure DDF to be a fanout proxy to other federated sources within the enterprise. The Fanout Catalog Framework has no catalog provider

configured for it, so it does not allow catalog modifications to take place. Therefore, create, update, and delete operations are not supported.

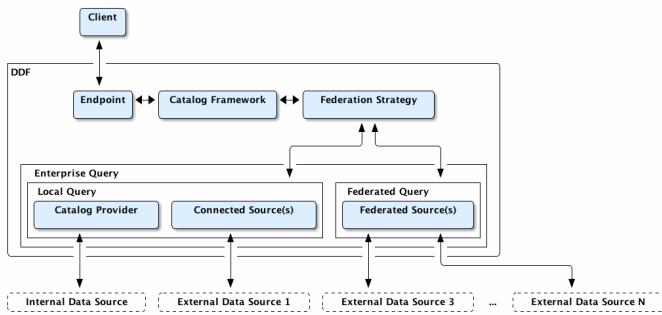


Figure 13. Catalog Fanout Framework

In addition, the Fanout Catalog Framework provides the following benefits:

- Backwards compatibility (e.g., federating with older versions) with existing older versions of DDF.
- A single node being exposed from an enterprise, thus hiding the enterprise from an external client.
- Ensures all queries and resource retrievals are federated.

### Installing and Uninstalling

The Fanout Catalog Framework is bundled as the `catalog-core-fanoutframework` feature and can be installed and uninstalled using the normal processes described in Configuration.

**WARNING**

When this feature is installed, the Standard Catalog Framework feature `catalog-core-standardframework` should be uninstalled, as both catalog frameworks should not be installed simultaneously.

### 22.9.2. Configuring

The Fanout Catalog Framework can be configured using the normal processes described in Configuring DDF.

The configurable properties for the Fanout Catalog Framework are accessed from the Catalog Fanout Framework configuration in the Admin Console.

#### Configurable Properties

Title	Property	Type	Description	Default Value	Required
Default Timeout (in milliseconds)	defaultTimeout	Integer	The maximum amount of time to wait for a response from the Sources.	60000	yes
Product Cache Directory	productCacheDirectory	String	<p>Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Catalog Framework will attempt to create those directories. Out of the box (without configuration), the product cache directory is <code>&lt;INSTALL_DIR&gt;/data/product-cache</code>. If a relative path is provided, it will be relative to the <code>&lt;INSTALL_DIR&gt;</code>.</p> <p>It is recommended to enter an absolute directory path, such as <code>/opt/product-cache</code> in Linux or <code>C:\product-cache</code> in Windows.</p>	(empty)	no
Enable Product Caching	cacheEnabled	Boolean	Check to enable caching of retrieved products to provide faster retrieval for subsequent requests for the same product.	false	no
Delay (in seconds) between product retrieval retry attempts	delayBetweenRetryAttempts	Integer	The time to wait (in seconds) between attempting to retry retrieving a product.	10	no
Max product retrieval retry attempts	maxRetryAttempts	Integer	The maximum number of attempts to retry retrieving a product.	3	no
Caching Monitor Period	cachingMonitorPeriod	Integer	How many seconds to wait and not receive product data before retrying to retrieve a product.	5	no

Title	Property	Type	Description	Default Value	Required
Always Cache Product	cacheWhenCanceled	Boolean	Check to enable caching of retrieved products, even if client cancels the download.	false	no

Managed Service PID	DDF.catalog.impl.service.fanout.FanoutCatalogFramework
Managed Service Factory PID	N/A

## Using

The Fanout Catalog Framework is a core class of DDF when configured as a fanout proxy. It provides the methods for query and resource retrieval operations on the Sources, where all operations are enterprise-wide operations. By contrast, the Standard Catalog Framework supports create/update/delete operations of metacards, in addition to the query and resource retrieval operations.

Use the Fanout Catalog Framework if:

- exposing a single node for enterprise access and hiding the details of the enterprise, such as federate source's names, is desired.
- access to individual federated sources is not required.
- access to a catalog provider to create, update, and delete metacards is not required.

The Fanout 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 fanout configuration can be retrieved using the Fanout 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). Details of the individual federated sources is not included, only the fanout catalog framework.

## Implementation Details

### Exported Services

Registered Interface	Service Property	Value
<code>DDF.catalog.federation.FederationStrategy</code>	shortname	sorted
<code>org.osgi.service.event.EventHandler</code>	event.topics	<code>DDF/catalog/event/CREATED</code> , <code>DDF/catalog/event/UPDATED</code> , <code>DDF/catalog/event/DELETED</code>

Registered Interface	Service Property	Value
DDF.catalog.CatalogFramework		
org.codice.DDF.configuration.ConfigurationWatcher		
DDF.catalog.event.EventProcessor		
DDF.catalog.plugin.PostIngestPlugin		

## Imported Services

Registered Interface	Availability	Multiple
DDF.cache.CacheManager		false
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
DDF.catalog.plugin.PolicyPlugin	optional	true
DDF.catalog.plugin.AccessPlugin	optional	true
DDF.catalog.resource.ResourceReader	optional	true
DDF.catalog.source.ConnectedSource	optional	true
DDF.catalog.source.FederatedSource	optional	true

Registered Interface	Availability	Multiple
<code>org.osgi.service.event.EventAdmin</code>		false

### 22.9.3. Known Issues

None

### 22.9.4. 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
operation	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.

Header	Message Body (Input)	Exchange Modification (Output)
operation = CREATE	List<Metocard> or Metocard	exchange.getIn().getBody() updated with List of Metacards created

Header	Message Body (Input)	Exchange Modification (Output)
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 ingestion.

```
<route>
  <from uri="file:data/sampleData?noop=true" />
    <bean ref="sampleDataToMetocardConverter" method="convertToMetocard"/>\n
      <setHeader headerName="operation">
        <constant>CREATE</constant>
      </setHeader>
      <to uri="catalog:framework"/>
    </route>
```

## 22.9.5. Working with the Catalog Framework

### Catalog Framework Reference

The Catalog Framework can be requested from the OSGi registry.

#### *Blueprint Service Reference*

```
<reference id="catalogFramework" interface="DDF.catalog.CatalogFramework" />
```

### Methods

#### Create, Update, and Delete

Create, Update, and Delete (CUD) methods add, change, or remove stored metadata in the local Catalog Provider.

## 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

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.

### 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.

## Resources

Resource methods retrieve products from Sources.

### 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.

## Sources

Source methods can get a list of Source identifiers or request descriptions about Sources.

## *Source Methods*

```
public Set<String> getSourceIds();
public SourceInfoResponse getSourceInfo(SourceInfoRequest sourceInfoRequest) throws
SourceUnavailableException;
```

## **Transforms**

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;
```

## **22.9.6. Developing Complementary Frameworks**

DDF and the underlying OSGi technology can serve as a robust infrastructure for developing frameworks that complement the DDF Catalog.

### Recommendations for Framework Development

1. Provide extensibility similar to that of the DDF Catalog.
  - a. Provide a stable API with interfaces and simple implementations (refer to [http://www.ibm.com/developerworks/websphere/techjournal/1007\\_charters/1007\\_charters.html](http://www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html)).
2. Make use of the DDF Catalog wherever possible to store, search, and transform information.
3. Utilize OSGi standards wherever possible.
  - a. ConfigurationAdmin
  - b. MetaType
4. Utilize the sub-frameworks available in DDF.
  - a. Karaf
  - b. CXF
  - c. PAX Web and Jetty

## 22.9.7. Developing Console Commands

### Console Commands

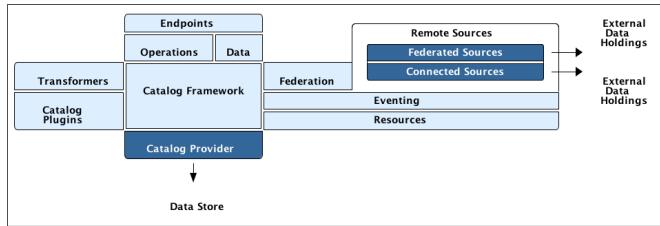
DDF supports development of custom console commands. For more information, see the Karaf website on <http://karaf.apache.org/manual/latest-2.2.x/developers-guide/extending-console.html>[Extending the Console].

### Custom DDF Console Commands

DDF includes custom commands for working with the Catalog, as described in the Console Commands section.

## 22.10. Extending Sources

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.



### 22.10.1. Existing Source Types

#### Catalog Provider

A Catalog provider provides an implementation of a searchable and writable catalog. 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.

#### Using

The Standard Catalog Framework uses only one Catalog provider, determined by the OSGi Framework as the service reference with the highest service ranking. In the case of a tie, the service with the lowest service ID (first created) is used.

The Catalog Fanout Framework App does not use a Catalog provider and will fail any create/update/delete operations even if there are active Catalog providers configured.

The Catalog reference implementation comes with a Solr Catalog Provider out of the box.

## 22.10.2. Remote Sources

Remote sources are read-only data sources that support query operations but cannot be used to create, update, or delete metacards.

**TIP** 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`.

## 22.10.3. Connected Source

A connected source is a remote source that is included in all local and federated queries but remains hidden from external clients. 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.

image::query-flow.png, 500[]

## 22.10.4. Federated Source

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.

DDF Catalog instances can also be federated to each other. Therefore, a DDF Catalog can also act as a federated source to another DDF Catalog.

## 22.10.5. 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 and Uninstalling

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

### Configuring

This component can be configured using the normal processes described in the Configuring DDF section. The configurable properties for the OpenSearch source are accessed from the Catalog OpenSearch Federated Source Configuration in the Web Console.

## Configuring the OpenSearch Source

### Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source Name	<code>shortname</code>	String		DDF-OS	Yes
OpenSearch service URL	<code>endpointUrl</code>	String	The OpenSearch endpoint URL, e.g., DDF's OpenSearch endpoint ( <a href="http://0.0.0.0:8181/services/catalog/query?q={searchTerms}...">http://0.0.0.0:8181/services/catalog/query?q={searchTerms}...</a> )	<code>https://example.com?q={searchTerms}&amp;src={fs:routeTo?}&amp;mr={fs:maxResults?}&amp;count={count?}&amp;mt={fs:maxTimeout?}&amp;dn={idn:userDN?}&amp;lat={geo:lat?}&amp;lon={geo:lon?}&amp;radius={geo:radius?}&amp;bbox={geo:box?}&amp;polygon={geo:polygon?}&amp;dtstart={time:start?}&amp;dtend={time:end?}&amp;dateName={cat:dateName?}&amp;filter={fsa:filter?}&amp;sort={fsa:sort?}</code>	Yes
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.		No

Title	Property	Type	Description	Default Value	Required
Password	password	String	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.		No
Always perform local query	localQueryOnly	Boolean	Always performs a local query by setting src=local OpenSearch parameter in endpoint URL. <b>This must be set if federating to another DDF.</b>	false	Yes
Convert to BBox	shouldConvertToBBox	Boolean	Converts Polygon and Point-Radius searches to a Bounding Box for compatibility with legacy interfaces. Generated bounding box is a very rough representation of the input geometry	true	Yes

## Using

Use the OpenSearch source if querying a CDR-compliant search service is desired.

## Source Details

### Default Security Settings (applicable to all OpenSearch Sources)

These settings are used to provide default security settings for the Title, Description, and Security elements in a record. The purpose of these defaults is that many providers fail to deliver a classification and Owner/Producer with the metadata returned. These default settings are used if a metadata record is returned without security settings. **This feature can be enabled/disabled.**

1. Open the Web Console.
  - a. <http://localhost:8181/system/console>
  - b. Username/Password: admin/admin
2. Click on the Configuration tab.
3. Find Catalog Security Defaults
4. Select whether or not to apply these defaults by checking or unchecking the box marked "Apply Default Security Settings."
5. If the applied defaults are selected, change the settings in the console to the default metadata

security.

a. These settings can also be changed by editing the file <INSTALL\_DIRECTORY>/etc/DDF/DDF.DefaultSiteSecurity.cfg

6. Click Save at the bottom of the configuration window (or save the file).

- In the CDR Open Search Service, if one of the properties has a blank value, the "last-resort" default is U and USA.
- If the <DefaultSiteSecurity.cfg file is deleted, it needs to be replaced otherwise all classification values are set to "last-resort" defaults U and USA.
- After the file is replaced, either restart DDF or the restart the CDR Open Search Service to reload the property values.

**WARNING**

## Query Format

### OpenSearch Parameter to DDF Query Mapping

OpenSearch/CDR Parameter	DDF Data Location
q={searchTerms}	Pulled verbatim from DDF query.
src={fs:routeTo?}	Unused
mr={fs:maxResults?}	Pulled verbatim from DDF query.
count={count?}	Pulled verbatim from DDF query.
mt={fs:maxTimeout?}	Pulled verbatim from DDF query.
dn={idn:userDN?}	DDF Subject
lat={geo:lat?}	Pulled verbatim from DDF query.
lon={geo:lon?}	Pulled verbatim from DDF query.
radius={geo:radius?}	Pulled verbatim from DDF query.
bbox={geo:box?}	Converted from Point-Radius DDF query.
polygon={geo:polygon?}	Pulled verbatim from DDF query.
dtstart={time:start?}	Pulled verbatim from DDF query.
dtend={time:end?}	Pulled verbatim from DDF query.
dateName={cat:dateName?}	Unused
filter={fsa:filter?}	Unused
sort={fsa:sort?}	Translated from DDF query. Format: "relevance" or "date" Supports "asc" and "desc" using colon as delimiter.

## Implementation Details

### Exported Services

Registered Interface	Service Property	Value
DDF.catalog.source.FederatedSource		

### Imported Services

Registered Interface	Availability	Multiple	Filter
DDF.catalog.transform.InputTransformer	required	false	(&(mime-type=text/xml)(id=xml))

### Known Issues

The OpenSearch source does not provide a Connected Source interface.

## 22.10.6. Developing a Source

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. Sources reside in the Sources area of the DDF Overview.

### Creating a New Source

#### 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](#)  
*Used to communicate with back-end storage. Allows for Query and Create/Update/Delete operations.*
- Federated Source - [DDF.catalog.source.FederatedSource](#)  
*Used to communicate with remote systems. Only allows query operations.*
- Connected Source - [DDF.catalog.source.ConnectedSource](#)  
*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.*

The procedure for implementing any of the source types follows a similar format: . Create a new class

that implements the specified Source interface and `ConfiguredService`. . Implement the required methods. . Create an OSGi descriptor file to communicate with the OSGi registry. (Refer to the OSGi Services section.) .. Import DDF packages. .. Register source class as service to the OSGi registry. . Deploy to DDF.

**IMPORTANT**

The `factory-pid` property of the metatype must contain one of the following in the name: service, Service, source, Source

## 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.

### *Blueprint example*

```
<service ref="[[TestCatalogProvider]]" interface="DDF.catalog.source.CatalogProvider" />
```

The DDF Integrator's Guide provides details on the following Catalog Providers that come with DDF out of the box.

**NOTE**

A code example of a Catalog Provider delivered with DDF is the Catalog Solr Embedded Provider.

## Federated Source

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.

## *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 `DDF.catalog.source.solr`

## **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.

## *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.

## **Examples**

- "Site XYZ not found" message rather than the full stack trace with the original site not found exception.
- 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.
- 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."

#### Additional Information

- [Three Rules for Effective Exception Handling](#)

### 22.10.7. Developing a Filter Delegate

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.

#### Creating a New Filter Delegate

A Filter Delegate contains the logic that converts normalized filter input into a form that the targeted 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 DDF 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

PropertyIsLessThanOrEqual To

PropertyIsLike

## Property Comparison

`PropertyIsNotEqualTo`

`PropertyIsNull`

Spatial	Definition
<code>Beyond</code>	True if the geometry being tested is beyond the stated distance of the geometry provided.
<code>Contains</code>	True if the second geometry is wholly inside the first geometry.
<code>Crosses</code>	True if the intersection of the two geometries results in a value whose dimension is less than the geometries and the maximum dimension of the intersection value includes points interior to both the geometries, and the intersection value is not equal to either of the geometries.
<code>Disjoint</code>	True if the two geometries do not touch or intersect.
<code>DWithin</code>	True if the geometry being tested is within the stated distance of the geometry provided.
<code>Intersects</code>	True if the two geometries intersect. This is a convenience method as you could always ask for Not Disjoint(A,B) to get the same result.
<code>Overlaps</code>	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.
<code>Touches</code>	True if and only if the only common points of the two geometries are in the union of the boundaries of the geometries.
<code>Within</code>	True if the first geometry is wholly inside the second geometry.

## Temporal

`After`

`Before`

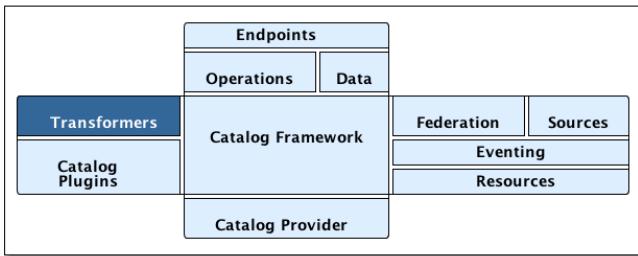
`During`

## 22.11. Extending Catalog 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.

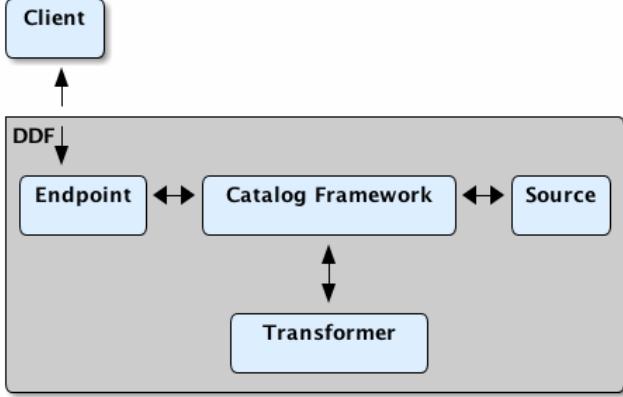
Transformers are utility objects used to transform a set of standard DDF components into a desired format, such as into PDF, GeoJSON, XML, or any other format. For instance, a transformer can be used

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.



*Figure 14. Transformers*

**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 15. Communication Diagram*

### 22.11.1. Working with Transformers

The `DDF.catalog.transform` package includes the `InputTransformer`, `MetacardTransformer`, and `QueryResponseTransformer` interfaces. All implementations can be accessed using the Catalog Framework or OSGi Service Registry, as long as the implementations have been registered with the Service Registry.

### 22.11.2. Catalog Framework

The `CatalogFramework` provides convenient methods to transform `Metacards` and `QueryResponses` using a

reference to the `CatalogFramework`. See Working with the Catalog Framework for more details on the method signatures. It is easy to execute the convenience `transform` methods on the `CatalogFramework` instance.

#### 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);

    // ...
}
```

Line #	Action
4	<code>CatalogFramework</code> is injected, possibly by dependency injection framework.
16	<code>queryResponse</code> is transformed into the RSS format, which is stored in the <code>BinaryContent</code> instance

### 22.11.3. 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);
```

## 22.11.4. 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.getNa  
me(),"(shortname=" + transformerId + ")");  
InputTransformer inputTransformer = (InputTransformer) context.getService(refs[0]);  
Metocard entry = inputTransformer.transform(messageInputStream);
```

## 22.11.5. Included Input Transformers

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.

## 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 Metocard. This input transformer utilizes 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 default input transformer.

### Installing and Uninstalling

This transformer is installed by default. To install or uninstall manually, use the `tika-input-transformer` feature in the Admin Console (<https://localhost:8993/admin>) under DDF Catalog Features or use the System Console.

Install the `catalog-transformer-tika` feature using the  `${admin-console}`. This feature is uninstalled by default.

### Configuring

None

### Using

Use the Tika Input Transformer for ingesting Microsoft documents, OpenOffice documents, or PDF documents into the Catalog.

### Service Properties

Key	Value
mime-type	<ul style="list-style-type: none"> <li>* application/pdf</li> <li>* application/vnd.openxmlformats-officedocument.wordprocessingml.document</li> <li>* application/vnd.openxmlformats-officedocument.spreadsheetml.sheet</li> <li>* application/vnd.openxmlformats-officedocument.presentationml.presentation</li> <li>* application/vnd.openxmlformats-officedocument.presentationml.presentation</li> <li>* application/vnd.ms-powerpoint.presentation.macroenabled.12</li> <li>* application/vnd.ms-powerpoint.slideshow.macroenabled.12</li> <li>* application/vnd.openxmlformats-officedocument.presentationml.slideshow</li> <li>* application/vnd.ms-powerpoint.template.macroenabled.12</li> <li>* application/vnd.oasis.opendocument.text</li> </ul>
shortname	
id	tika
title	Tika Input Transformer
description	Default Input Transformer for all mime types.
service.ranking	-1

### Implementation Details

This input transformer maps the metadata common across all mime types to applicable metocard attributes in the default MetacardType.

### PPTX Input Transformer

The PPTX Input Transformer is the input transformer responsible for translating Microsoft PowerPoint (OOXML only) documents into a Catalog Metocard. 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 and Uninstalling

This transformer is installed by default. To install or uninstall manually, use the `catalog-transformer-pptx` feature in the Admin Console (<https://localhost:8993/admin>) under DDF Catalog Features.

### Configuring

This transformer does not require any configuring.

## Using

Use the PPTX Input Transformer for ingesting Microsoft PowerPoint (OOXML only) documents into the DDF Content Repository and/or the Metadata Catalog.

### Service Properties

Key	Value
mime-type	* application/vnd.openxmlformats-officedocument.presentationml.presentation
id	pptx
title	PPTX Input Transformer
description	Default Input Transformer for the <code>application/vnd.openxmlformats-officedocument.presentationml.presentation</code> mime type.

### 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.

## 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 and Uninstalling

This transformer is installed by default. To install or uninstall manually, use the `video-input-transformer` feature in the Admin Console (<https://localhost:8993/admin>) under DDF Catalog Features or use the System Console.

### Configuring

None

## Using

Use the video input transformer for ingesting video files into the Catalog.

### 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.

## GeoJSON Input Transformer

The GeoJSON input transformer is responsible for translating specific GeoJSON into a Catalog metocard.

### Installing and Uninstalling

Install the catalog-rest-endpoint feature using the [Admin Console](#).

### Configuring

None

### Using

Using the REST Endpoint, for example, HTTP POST a GeoJSON metocard to the Catalog. Once the REST Endpoint receives the GeoJSON Metocard, it is converted to a Catalog metocard.

*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"
http://localhost:8181/services/catalog
```

### Conversion

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, that is not enough information 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.

```
{  
  "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",  
    "metacard-type": "DDF.metacard.custom.type"  
  },  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [  
      30.0,  
      10.0  
    ]  
  }  
}
```

When the GeoJSON Input Transformer gets GeoJSON with the **MetacardType** specified, it will perform a lookup in the **MetacardTypeRegistry** to obtain the specified **MetacardType** in order to understand how to parse the GeoJSON. If no **MetacardType** is specified, the GeoJSON Input Transformer will assume the default **MetacardType**. If an unregistered **MetacardType** is specified, an exception will be returned to the client indicating that the **MetacardType** was not found.

## Package Details

### *Feature Information*

N/A

### *Included Bundles*

N/A

### *Services*

#### *Exported Services*

*Table 47. DDF.catalog.transform.InputTransformer*

mime-type	application/json
id	geojson

## Implementation Details

*Table 48. Exported Services*

Registered Interface	Service Property	Value
DDF.catalog.transform.InputTransformer	mime-type	application/json
	id	geojson

## Known Issues

Does not handle multiple geometries yet.

## 22.11.6. Developing an Input Transformer

### Using Java

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.

## 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>
...
...
```

1. Deploy OSGi Bundle to OSGi runtime.

### Variable Descriptions

Table 49. 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.	<code>atom</code>
<code>title</code>	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	<code>Atom Entry Transformer Service</code>
<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>

## 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.

### Design Pattern

#### 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.

## Message Formats

Table 50. InputTransformer

Exchange Type	Field	Type
Request (comes from <from> in the route)	body	java.io.InputStream
Response (returned after called via <to> in the route)	body	DDF.catalog.data.Metocard

## Examples

### InputTransformer Creation

```

<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.

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.  <b>Note that the specified XSL stylesheet must be on the classpath of the bundle that this blueprint file is packaged in.</b>
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](#)

## 22.11.7. Included Metocard InputTransformers

A metocard transformer transforms a metocard into other data formats.

### HTML Metocard Transformer

The HTML metocard transformer is responsible for translating a metocard into an HTML formatted document.

#### Installing and Uninstalling

Install the `catalog-transformer-html` feature using the Web Console (<http://localhost:8181/system/console>) or System Console.

#### Configuring

None

#### Using

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

```
http://localhost:8181/services/catalog/0123456789abcdef0123456789abcdef?transform=html
```

#### Example Output

```
html metocard.png
```

#### Implementation Details

Registered Interface	Service Property	Value
<code>DDF.catalog.transform.MetocardTransformer</code>	title	View as html...
	description	Transforms query results into html
	shortname (for backwards compatibility)	html

## Known Issues

None

### 22.11.8. XML Metocard Transformer

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

#### Installing and Uninstalling

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

#### Configuring

None

#### Using

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

```
http://localhost:8181/services/catalog/ac0c6917d5ee45bfb3c2bf8cd2ebaa67?transform=xml
```

#### Implementation Details

##### Metocard to XML Mappings

Metocard Variables
<code>XML Element</code>
<code>id</code>
<code>metocard/@gml:id</code>
<code>metocardType</code>
<code>metocard/type</code>
<code>sourceId</code>
<code>metocard/source</code>
<code>all other attributes</code>
<code>metocard/&lt;AttributeType&gt;[name='&lt;AttributeName&gt;']/value</code>
For instance, the value for the metocard attribute named "title" would be found at <code>metocard/string[@name='title']/value</code>

## AttributeTypes

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

## Known Issues

None

### 22.11.9. GeoJSON Metacard Transformer

GeoJSON Metacard Transformer translates a Metacard into GeoJSON.

#### Installing and Uninstalling

Install the [catalog-transformer-json](#) feature.

#### Configuring

None

#### Using

The GeoJSON Metacard Transformer can be used programmatically by requesting a [MetacardTransformer](#) with the id [geojson](#). It can also be used within the REST Endpoint by providing the transform option as [geojson](#).

*Example REST GET method with the GeoJSON MetacardTransformer*

```
http://localhost:8181/services/catalog/0123456789abcdef0123456789abcdef?transform=geojson
```

*Example Output*

```
{  
  "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": [  

```

## Implementation Details

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

## 22.11.10. Known Issues

None

## 22.11.11. Thumbnail Metacard Transformer

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

### Installing and Uninstalling

This transformer is installed out of the box. To uninstall the transformer, you must stop or uninstall the bundle.

### Configuring

None

### Using

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

#### *Sample Blueprint Reference Snippet*

```
<reference id="metocardTransformer" interface="DDF.catalog.transform.MetocardTransformer"
filter="(id=thumbnail)"/>
```

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

### Implementation Details

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

## Known Issues

None

### 22.11.12. Metadata Metocard Transformer

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

#### Installing and Uninstalling

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

#### Configuring

None

#### Using

The Metadata Metocard Transformer can be used programmatically by requesting a MetocardTransformer 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 MetocardTransformer*

```
http://localhost:8181/services/catalog/0123456789abcdef0123456789abcdef?transform=metadat  
a
```

#### Implementation Details

Registered Interface	Service Property	Value
<code>DDF.catalog.transform.MetocardTransformer</code>	mime-type	<code>text/xml</code>
	id	metadata
	shortname (for backwards compatibility)	metadata

## Known Issues

None.

### 22.11.13. Resource Metocard Transformer

The Resource Metocard Transformer retrieves the resource bytes of a metocard by returning the product associated with the metocard.

## Installing and Uninstalling

This transformer is installed or uninstalled with the feature `catalog-transformer-resource`.

## Configuring

None

## Using

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

### *Sample Blueprint Reference Snippet*

```
<reference id="metocardTransformer" interface="DDF.catalog.transform.MetocardTransformer"
filter="(id=resource)"/>
```

## Implementation Details

Service Property	Value	id
resource	shortname	resource
mime-type	application/octet-stream	title

## Known Issues

None

## 22.11.14. Developing a Metocard Transformer

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 `MetocardTransformer` transforms a `Metocard` into a `BinaryContent` instance, which contains the translated `Metocard` into the desired final format. Metocard transformers can be used through the Catalog Framework `transform` convenience method or requested from the OSGi Service Registry by endpoints or other bundles.

### Create a New Metocard Transformer

1. Create a new Java class that implements `DDF.catalog.transform.MetocardTransformer`.

```
public class SampleMetocardTransformer implements DDF.catalog.transform.MetocardTransformer
```

2. Implement the `transform` method.

```
public BinaryContent transform(Metocard metocard, 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 the Working with OSGi section). Export the service to the OSGi registry and declare service properties.

#### *Blueprint descriptor example*

```
...
<service ref="[[SampleMetocardTransformer]]" interface=
"DDF.catalog.transform.MetocardTransformer">
    <service-properties>
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Metocard Transformer]]" />
        <entry key="description" value="[[A new transformer for metacards.]]" />
    </service-properties>
</service>
...
...
```

Deploy OSGi Bundle to OSGi runtime.

#### **Variable Descriptions**

*Table 51. Blueprint Service properties*

Key	Description of Value	Example
shortname	(Required) An abbreviation for the return type of the BinaryContent being sent to the user.	atom
title	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	Atom Entry Transformer Service
description	(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.

### **22.11.15. Included Query Response Transformers**

Query Response transformers convert query responses into other data formats.

#### **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 and Uninstalling

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

## Configuring

None.

## Using

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 Results

## 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/" xmlns:fs="http://a9.com/-/opensearch/extensions/federation/1.0/">
    <ns1:georss="http://www.georss.org/georss">
      <fs:resultSource fs:sourceId="DDF123" />
      <relevance:score>0.19</relevance:score>
      <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:metocard>
</content>
</entry>
</feed>

```

## Query 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 ( <a href="http://en.wikipedia.org/wiki/Universally_Unique_Identifier">http://en.wikipedia.org/wiki/Universally_Unique_Identifier</a> )
/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:sourceId	Source Id from which the Result came. <code>Metocard.getSourceId()</code>

XPath to Atom XML	Value
/feed/entry/relevance:score	Result's relevance score if applicable. <code>Result.getRelevanceScore()</code>
/feed/entry/id	<code>urn:catalog:id:&lt;Metacard.ID&gt;</code>
/feed/entry/title	<code>Metacard.TITLE</code>
/feed/entry/updated	ISO 8601 dateTime of <code>Metacard.MODIFIED</code>
/feed/entry/published	ISO 8601 dateTime of <code>Metacard.CREATED</code>
/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	<code>Metacard.CONTENT_TYPE</code>
/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 <code>Metacard.ID</code> is displayed  .Sample Content with no Metacard Transformation [source,xml] ---- <content type="text">4e1f38d1913b4e93ac622e6c1b258f89</content> ----

## 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 and Uninstalling

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

### Configuring

None

### Using

Using the OpenSearch Endpoint, for example, query with the format option set to the XML shortname

xml.

```
http://localhost:8181/services/catalog/query?q=input&format=xml
```

### 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:metacard 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:metacard>
  <ns3:metacard 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:metacard>
</ns3:metacards>
```

### Implementation Details

Registered Interface	Service Property	Value
DDF.catalog.transform.QueryResp onseTransformer	shortname	xml
	description	Transforms query results into xml
	title	View as XML...

### Known Issues

None

## SearchUI

The 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 SearchUI contains basic fields to query the Catalog and other Sources. The right pane consists of the results returned from the query.

### Installing and Uninstalling

Catalog Transformers App will install this feature when deployed. This transformer's feature, [catalog-transformer-ui](#), can be uninstalled or installed.

### Configuring

In the Admin Console the SearchUI can be configured under the Catalog HTML Query Response Transformer.

*Table 52. Configurable Properties*

Title	Property	Type	Description	Default Value	Required
Header	<code>header</code>	String	Specifies the header text to be rendered on the SearchUI		yes
Footer	<code>footer</code>	String	Specifies the footer text to be rendered on the SearchUI		yes
Template	<code>template</code>	String	Specifies the path to the Template	/templates/searchpage.ftl	yes
Text Color	<code>color</code>	String	Specifies the Text Color of the Header and Footer	yellow	yes
Background Color	<code>background</code>	String	Specifies the Background Color of the Header and Footer	green	yes

### Using

In order to obtain the 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, <http://localhost:8181/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.

#### Query Response Result Mapping

SearchUI Column Title	Catalog Result	Notes
Title	<code>Metacard.TITLE</code>	The title may be hyperlinked to view the full Metacard
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 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

#### Known Issues

If the SearchUI results do not provide usable links on the metocard results, verify that a valid host has been entered in the Platform Global Configuration.

### 22.11.16. Developing a Query Response Transformer

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.

## 22.11.17. Create a New Query Response Transformer

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 the Working with OSGi section). Export the service to the OSGi registry and declare service properties.

*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>  
...
```

1. Deploy OSGi Bundle to OSGi runtime.

## 22.11.18. Variable Descriptions

**Blueprint Service properties**

Key	Description of Value	Example
<code>shortname</code>	An abbreviation for the return-type of the <code>BinaryContent</code> being sent to the user.	atom
<code>title</code>	A user-readable title that describes (in greater detail than the <code>shortname</code> ) 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>

## 22.11.19. XSLT Transformer

## 22.11.20. XSLT Transformer Framework

The XSLT Transformer Framework allows developers to create light-weight Query Response Transformers and Metocard Transformers using only a bundle header and XSLT files. The XSLT Transformer Framework registers bundles, following the XSLT Transformer Framework bundle pattern, as new transformer services. The `service-xslt-transformer` feature is part of the DDF core.

### Examples

Examples of XSLT Transformers using the XSLT Transformer Framework include `service-atom-transformer` and `service-html-transformer`, found in the services folder of the source code trunk.

## 22.11.21. Developing an XSLT Transformer

The XSLT Transformer Framework allows developers to create light-weight Query Response Transformers using only a bundle header and XSLT files. The XSLT Transformer Framework registers bundles, following the XSLT Transformer Framework bundle pattern, as new transformer services. The `service-xslt-transformer` feature is part of the DDF core.

### Examples

Examples of XSLT Transformers using the XSLT Transformer Framework include `service-atom-transformer` and `service-html-transformer`, found in the services folder of the source code trunk.

### Implement an XSLT Transformer

1. Create a new Maven project.
2. Configure the POM to create a bundle using the Maven bundle plugin.
  - a. Add the transform output MIME type to the bundle headers.
3. Add XSLT files.

### Bundle POM Configuration

Configure the Maven project to create an OSGi bundle using the `maven-bundle-plugin`. Change the DDF-Mime-Type to match the MIME type of the transformer output.

## Example POM file

```
...
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.felix</groupId>
      <artifactId>maven-bundle-plugin</artifactId>
      <extensions>true</extensions>
      <configuration>
        <instructions>
          <DDF-Mime-Type>[[Transform Result MIME Type]]</DDF-Mime-Type>
          <Bundle-SymbolicName>docs</Bundle-SymbolicName>
          <Import-Package />
          <Export-Package />
        </instructions>
      </configuration>
    </plugin>
  </plugins>
</build>
...

```

## Including XSLT

The XSLT Transformer Framework will scan for XSLT files inside a bundle. The XSLT file must have a `.xsl` or `.xslt` file in the correct directory location relative to the root of the bundle. The path depends on if the XSLT will act as a Metocard Transformer, Query Response Transformer, or both. The name of the XSLT file will be used as the transformer's shortname.

### XSLT File Bundle Path Patterns

```
// Metocard Transformer
<bundle root>
/OSGI-INF
/DDF
/xslt-metocard-transformer
/<transformer shortname>.[xsl|xslt]

// Query Response Transformer
<bundle root>
/OSGI-INF
/DDF
/xslt-response-queue-transformer
/<transformer shortname>.[xsl|xslt]
```

The XSLT file has access to metocard or Query Reponse XML data, depending on which folder the XSLT

file is located. The Metacard XML format will depend on the metadata schema used by the Catalog Provider.

For Query Response XSLT Transformers, the available XML data for XSLT transform has the following structure:

#### *Query Response XML*

```
<results>
  <metacard>
    <id>[[Metacard ID]]</id>
    <score>[[Relevance score]]</score>
    <distance>[[Distance from query location]]</distance>
    <site>[[Source of result]]</site>
    <type qualifier="type">[[Type]]</type>
    <updated>[[Date last updated]]</updated>
    <geometry>[[WKT geometry]]</geometry>
    <document>
      [[Metacard XML]]
    </document>
  </metacard>
  ...
</results>
```

The XSLT file has access to additional parameters. The `Map<String, Serializable>` arguments from the transform method parameters is merged with the available XSLT parameters.

- Query Response Transformers
  - `grandTotal` - total result count
- Metacard Transformers
  - `id` - metacard ID
  - `siteName` - source ID
  - `services` - list of displayable titles and URLs of available metacard transformers

#### RSS Example

1. Create a Maven project named `service-rss-transformer`.
2. Add the following to its POM file.

## Example RSS POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <packaging>bundle</packaging>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <artifactId>services</artifactId>
    <groupId>DDF</groupId>
    <version>[[DDF release version]]</version>
  </parent>
  <groupId>DDF.services</groupId>
  <artifactId>service-rss-transformer</artifactId>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
        <extensions>true</extensions>
        <configuration>
          <instructions>
            <DDF-Mime-Type>application/rss+xml</DDF-Mime-Type>
            <Bundle-SymbolicName>docs</Bundle-SymbolicName>
            <Import-Package />
            <Export-Package />
          </instructions>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Line #	Comment
8	Use the current release version.
21	Set the MIME type to the RSS MIME type.

1. Add `service-rss-transformer/src/main/resources/OSGI-INF/DDF/xslt-response-queue-transformer/rss.xsl`. The transformer will be a Query Response Transformer with the shortname `rss` based on the XSL filename and path.
2. Add the following XSL to the new file.

## Example RSS XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:gml="http://www.opengis.net/gml" exclude-result-prefixes="xsl gml">

  <xsl:output method="xml" version="1.0" indent="yes" />

  <xsl:param name="grandTotal" />
  <xsl:param name="url" />

  <xsl:template match="/">
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="results">
    <rss version="2.0">
      <channel>
        <title>Query Results</title>
        <link><xsl:value-of select="$url" disable-output-escaping="yes" /></link>
        <description>Query Results of <xsl:value-of select="count(//metocard)" /> out of
<xsl:value-of select="$grandTotal" /></description>
        <xsl:for-each select="metocard/document">
          <item>
            <guid>
              <xsl:value-of select="..../id" />
            </guid>
            <title>
              <xsl:value-of select="Data/title" />
            </title>
            <link>
              <xsl:value-of select="substring-before($url,'/services')" />
<xsl:text>/services/catalog</xsl:text><xsl:value-of select="..../id" />
<xsl:text>?transform=html</xsl:text>
            </link>
            <description>
              <xsl:value-of select="//description" />
            </description>
            <author>
              <xsl:choose>
                <xsl:when test="Data/creator">
                  <xsl:value-of select="Resource/creator//name" />
                </xsl:when>
                <xsl:when test="Data/publisher">
                  <xsl:value-of select="Data/publisher//name" />
                </xsl:when>
                <xsl:when test="Data/unknown">
```

```

<xsl:value-of select="Data/unknown//name" />
</xsl:when>
</xsl:choose>
</author>
<xsl:if test=".//@posted" >
  <pubDate>
    <xsl:value-of select=".//posted" />
  </pubDate>
</xsl:if>
</item>
</xsl:for-each>
</channel>
</rss>
</xsl:template>
</xsl:stylesheet>

```

<b>Line #</b>	<b>Comment</b>
8-9	Example of using additional parameters and arguments.
15	Example of using the Query Response XML data.
21,27	Example of using the Metocard XML data.

## 22.12. Extending Federation

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, processed, and 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 metocard identifies the Source from which the metocard originated.

The Catalog normalizes the incoming query 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.

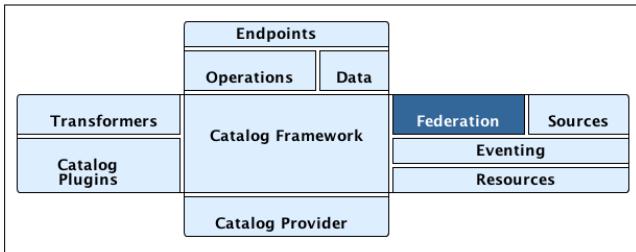


Figure 16. Federation Architecture

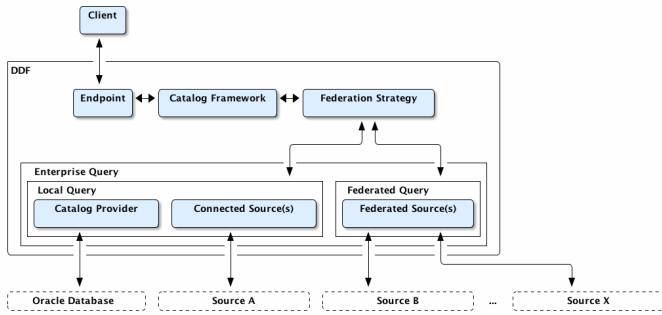


Figure 17. Federation

## 22.12.1. Federation Strategy

A federation strategy federates a query to all of the Remote Sources in the query's list, processes the results in a unique way, then returns the results to the client. For example, implementations can choose to block until all results return then perform a mass sort or return the results back to the client as soon as they are received back from a Federated Source.

### Usage

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.

#### 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.

## Configuration

The Catalog Federation Strategy configuration can be found in the admin console under **Configuration Catalog Federation Strategy**.

Property	Type	Description	Default Value	Required
maxStartIndex	Integer	<p>The maximum query offset number (any number from 1 to unlimited). Setting the number too high would allow offset queries that could result in an out of memory error because the DDF will cycle through all records in memory. Things to consider when setting this value are:</p> <ul style="list-style-type: none"><li>* How much memory is allocated to the DDF Server</li><li>* How many sites are being federated with.</li></ul>	50000	yes
expirationIntervalInMinutes	Long	Interval that Solr Cache checks for expired documents to remove.	10	yes
expirationAgeInMinutes	Long	The number of minutes a document will remain in the cache before it will expire. Default is 7 days.	10080	yes
url	String	HTTP URL of Solr Server	<a href="https://localhost:8993/solr">https://localhost:8993/solr</a>	yes
cachingEverything	Boolean	Cache all results unless configured as native	false	yes

Managed Service PID	<code>DDF.catalog.federation.impl.CachingFederationStrategy</code>
Managed Service Factory PID	N/A

## 22.13. Extending Eventing

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.

To better understand why this would be useful, suppose that there has been increased pirating activity off the coast of Somalia. Because of these events, a group of intelligence analysts is interested in determining the reason for the heightened activity and discovering its cause. To do this, analysts need to monitor interesting events occurring in that area. Without DDF Eventing, the analysts would need to repeatedly query for any records of events or intelligence gathered in that area. Analysts would have to monitor changes or anything of interest. However, with DDF Eventing, the analysts can create a subscription indicating criteria for the types of intelligence of interest. In this scenario, analysts could specify interest in metacards added, updated, or deleted that describe data obtained around the coast of Somalia. Through this subscription, DDF will send event notifications back to the team of analysts containing metadata of interest. Furthermore, they could filter the records not only spatially, but by any other criteria that would zero in on the most interesting records. For example, a fishing company that has operated ships peacefully in the same region for a long time may not be interesting. To exclude metadata about that company, analysts may add contextual criteria indicating to return only records containing the keyword "pirate." With the subscription in place, analysts will only be notified of metadata related to the pirating activity, giving them better situational awareness.

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

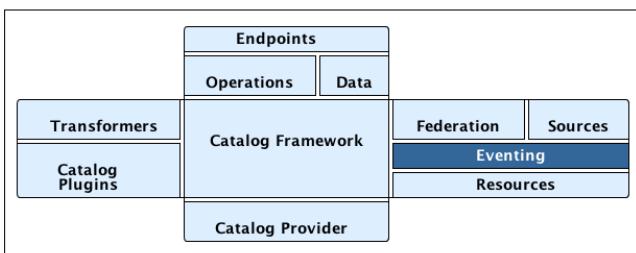


Figure 18. Eventing Architecture

## 22.13.1. Subscription

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

### Subscription Lifecycle

#### 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.**

#### **IMPORTANT**

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.

#### 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.

## **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.

## **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.

## **Standard Event Processor**

The Standard Event Processor is an implementation of the Event Processor and provides the ability to

create/delete subscriptions. Events are generated by the DDF 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 and Uninstalling**

The StandardEvent Processor is automatically installed/uninstalled when the Standard Catalog Framework is installed/uninstalled.

### **Known Issues**

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.

## **22.13.2. Fanout Event Processor**

The Fanout Event Processor is used when DDF is configured as a fanout proxy. The only difference between the Fanout Event Processor and the Standard Event Processor is that the source ID in the metocard of each event is overridden with the fanout's source ID. This is done to hide the source names of the Remote Sources in the fanout's enterprise. Otherwise, the Fanout Event Processor functions exactly like the Standard Event Processor.

### **Installing and Uninstalling**

The Fanout Event Processor is automatically installed/uninstalled when the Catalog Fanout Framework App is installed/uninstalled.

### **Known Issues**

None

## **22.13.3. Working with Subscriptions**

### **Creating a Subscription**

#### **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.

```
// Create a new filter using an imported FilterBuilder
Filter filter = filterBuilder.attribute(Metacard.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);
```

### Creating a Custom Implementation

To create a subscription in DDF the developer needs to implement the `DDF.catalog.event.Subscription` interface. This interface extends `org.opengis.filter.Filter` in order to represent the subscription's filter criteria. Furthermore, the `Subscription` interface contains a `DeliveryMethod` implementation.

When implementing `Subscription`, the developer will need to override the methods `accept` and `evaluate` from the `Filter`. The `accept` method allows the visitor pattern to be applied to the `Subscription`. A `FilterVisitor` can be passed into this method in order to process the `Subscription`'s `Filter`. In DDF, this method is used to convert the `Subscription`'s `Filter` into a predicate format that is understood by the Event Processor. The second method inherited from `Filter` is `evaluate`. This method is used to evaluate an object against the `Filter's criteria in order to determine if it matches the criteria.

**TIP** The functionality of these overridden methods is typically delegated to the `Filter` implementation that the `Subscription` is using.

The developer must also define `getDeliveryMethod`. This class is called when the an event occurs that matches the filter of the subscription.

The other two methods required because `Subscription` implements `Federatable` are `isEnterprise` and `getSourceIds`, which indicate that the subscription should watch for events occurring on all sources in the enterprise or on specified sources.

The following is an implementation stub of `Subscription` that comes with DDF and is available

at DDF.catalog.event.impl.SubscriptionImpl.

### SubscriptionImpl

```
public class SubscriptionImpl implements Subscription {
    private Filter filter;

    private DeliveryMethod dm;

    private Set<String> sourceIds;

    private boolean enterprise;

    public SubscriptionImpl(Filter filter, DeliveryMethod dm, Set<String> sourceIds,
                           boolean enterprise) {
        this.filter = filter;
        this.dm = dm;
        this.sourceIds = sourceIds;
        this.enterprise = enterprise;
    }

    @Override
    public boolean evaluate(Object object) {
        return filter.evaluate(object);
    }

    @Override
    public Object accept(FilterVisitor visitor, Object extraData) {
        return filter.accept(visitor, extraData);
    }

    @Override
    public Set<String> getSourceIds() {
        return sourceIds;
    }

    @Override
    public boolean isEnterprise() {
        return enterprise;
    }

    @Override
    public DeliveryMethod getDeliveryMethod() {
        return dm;
    }
}
```

## 22.13.4. Registering a Subscription

Once a `Subscription` is created, it needs to be registered in the OSGi Service Registry as a `DDF.catalog.event.Subscription` service. This is necessary for the `Subscription` to be discovered by the Event Processor. Typically, this is done in code after the `Subscription` is instantiated. When the `Subscription` is registered, a unique ID will need to be specified using the key `subscription-id`. This will be used to delete the `Subscription` from the OSGi Service Registry. Furthermore, the `ServiceRegistration`, which is the return value from registering a `Subscription`, should be monitored in order to remove the `Subscription` later. The following code shows how to correctly register a `Subscription` implementation in the registry using the above `SubscriptionImpl` for clarity:

### *Registering a Subscription*

```
// Map to keep track of registered Subscriptions. Used for unregistering Subscriptions.  
Map<String, ServiceRegistration<Subscription>> subscriptions = new HashMap<String,  
ServiceRegistration<Subscription>>();  
  
// New subscription using the DDF Implementation of subscription  
Subscription subscription = new SubscriptionImpl(filter, deliveryMethod, sourceIds  
, isEnterprise);  
  
// Specify the subscription-id to uniquely identify the Subscription  
String subscriptionId = "0123456789abcdef0123456789abcdef";  
Dictionary<String, String> properties = new Hashtable<String, String>();  
properties.put("subscription-id", subscriptionId);  
  
// Service registration requires an instance of the OSGi bundle context  
// Register subscription and keep track of the service registration  
ServiceRegistration<Subscription> serviceRegistration = context.registerService(DDF  
.catalog.event.Subscription.class, subscription, properties );  
subscriptions.put(subscriptionId, serviceRegistration);
```

## 22.13.5. Creating a Delivery Method

The Event Processor obtains the subscription's `DeliveryMethod` and invokes one of its four methods when an event occurs. The `DeliveryMethod` then handles that invocation and communicates an event to a specified consumer service outside of DDF.

The Event Processor calls the `DeliveryMethod`'s `created` method when a new metocard matching the filter criteria is added to the Catalog. It calls the `deleted` method when a metocard that matched the filter criteria is removed from the Catalog. `updatedHit` is called when a metocard is updated and the new metocard matches the subscription. `updatedMiss` is different in that it is only called if the old metocard matched the filter but the new metocard no longer does. An example of this would be if the filter contains spatial criteria consisting of Arizona. If a plane is flying over Arizona, the Event Processor will repeatedly call `updatedHit` as the plane flies from one side to the other while updating its position in the Catalog. This happens because the updated records continually match the specified

criteria. If the plane crosses into New Mexico, the previous metocard will have matched the filter, but the new metocard will not. Thus, `updatedMiss` gets called.

The following is an implementation stub for `DeliveryMethod`:

#### *DeliveryMethodImpl*

```
public class DeliveryMethodImpl implements DeliveryMethod {

    @Override
    public void created(Metocard newMetocard) {
        // Perform custom code on create
    }

    @Override
    public void updatedHit(Metocard newMetocard, Metocard oldMetocard) {
        // Perform custom code on update (where both new and old metacards matched filter)
    }

    @Override
    public void updatedMiss(Metocard newMetocard, Metocard oldMetocard) {
        // Perform custom code on update (where one of the two metacards did not match the
        // filter)
    }

    @Override
    public void deleted(Metocard oldMetocard) {
        // Perform custom code on delete
    }
}
```

#### 22.13.6. Deleting a Subscription

To remove a subscription from DDF, the subscription ID is required. Once this is provided, the `ServiceRegistration` for the indicated `Subscription` should be obtained from the `Subscriptions` Map. Then the `Subscription` can be removed by unregistering the service. The following code demonstrates how this is done:

## Delete Subscription

```
String subscriptionId = "0123456789abcdef0123456789abcdef";  
  
//Obtain service registration from subscriptions Map based on subscription ID  
ServiceRegistration<Subscription> sr = subscriptions.get(subscriptionId);  
  
//Unregister Subscription from OSGi Service Registry  
sr.unregister();  
  
//Remove Subscription from Map keeping track of registered Subscriptions.  
subscriptions.remove(subscriptionId);
```

## 22.14. Extending Resource Components

Resource components are used when working with resources, i.e., the data that is represented by the catalogued metadata.

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.

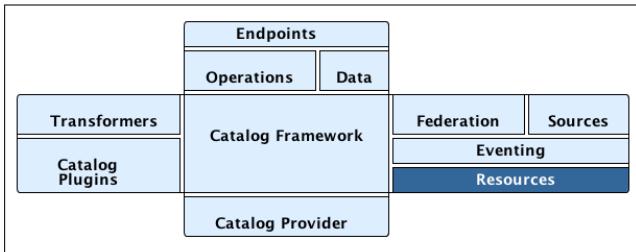


Figure 19. Resources Architecture

## 22.14.1. 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](#).

### 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 and Uninstalling

[URLResourceReader](#) is installed by default with the DDF Catalog.

### Configuring

#### Configurable Properties

##### *URL Resource Reader*

Property	Type	Description	Default Value	Required
<code>rootResourceDirectories</code>	String array	Specifies the only directories the <code>URLResourceReader</code> has access to when attempting to download resources linked using file-based URLs (i.e. the metocard attribute resource-uri uses the file URI scheme). This property is used to restrict the <code>URLResourceReader</code> 's access to the file system. The <code>URLResourceReader</code> can be given full access to the file system by setting the <code>rootResourceDirectories</code> property to the root directory (e.g. <code>/</code> ), but this configuration is not recommended.	<DDF.home>/data/products	yes

## Using

`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.

## Implementation Details

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.

## Known Issues

None

## 22.14.2. Developing a Resource Reader

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.

### Create 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.

An example of how **URLResourceReader** (located in the source code at `/trunk/DDF/catalog/resource/URLResourceReader.java`) implements the `getResource` method. This **ResourceReader** simply 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"
/>
```

### 22.14.3. 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.

#### Examples

The Catalog reference implementation currently does not include any resource writers out of the box.

### 22.14.4. Developing a Resource Writer

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 clients 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 DDF Catalog API Javadoc for more information about the methods required for implementing the interface.

## 22.14.5. Developing a Registry Client

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.

### Example

#### *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](#).

## 22.14.6. Working with Resources

### Metacards and Resources

Metacards are used to describe a resource 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.

### Retrieve Resource

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 in the 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 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.

### **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; however, the `ResourceReader` or `Source` supports options. 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.

### **Store Resource**

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 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`.

```
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
```

## **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.**

### **Additional Information**

- URI on Wikipedia ([http://en.wikipedia.org/wiki/Uniform\\_resource\\_identifier](http://en.wikipedia.org/wiki/Uniform_resource_identifier))
- URI Javadoc (<http://docs.oracle.com/javase/6/docs/api/java/net/URI.html>)

# 23. Extending DDF Platform

Version: 2.9.0

This section supports developers creating extensions of the existing framework.

## 23.1. Whitelist

The following packages have been exported by the DDF Platform application and are approved for use by third parties:

- `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`

**WARNING**

The Platform Application includes other third party packages such as Apache CXF and Apache Camel. These are available for use by third party developers but their versions can change at any time with future releases of the Platform Application.

## 23.2. Developing Action Components (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 a 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 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 out in 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`

### 23.2.1. Usage

To provide a service, such as a link to a record, the `ActionProvider` interface should be implemented. An `ActionProvider` essentially provides a List of `Action`'s 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.

### 23.2.2. 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 the metocard, such as the metadata or thumbnail, or when you are downloading the product. The verb 'Export' or expression 'Export as' is recommended when the metocard is being exported in a different format or

presented after going some transformation.

### 23.2.3. 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**

#### Action ID Service Descriptions

ID	Required Action	Naming Convention
<b>catalog.data.metocard.view</b>	Provides a valid URL to view all of a metocard data. Format of data is not specified; i.e. the representation can be in XML, JSON, or other.	Export as ...
<b>catalog.data.metocard.thumbnail</b>	Provides a valid URL to the bytes of a thumbnail ( <b>Metocard.THUMBNAIL</b> ) with MIME type image/jpeg.	Get Thumbnail
<b>catalog.data.metocard.html</b>	Provides a valid URL that, when invoked, provides an HTML representation of the metocard.	Export as ...
<b>catalog.data.metocard.resource</b>	Provides a valid URL that, when invoked, provides the underlying resource of the metocard.	Get Resource
<b>catalog.data.metocard.metadata</b>	Provides a valid URL to the XML metadata in the metocard ( <b>Metocard.METADATA</b> ).	Get Metadata

## 23.3. Developing Migratables

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 DDF 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.ConfigurationAminMigration`.

The Migratable API includes:

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`

### 23.3.1. Usage

The `org.codice.ddf.migration.Migratable` interface defines these methods:

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 `< DISTRIBUTION 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.

A `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 admin.

A `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 admin.

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.

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` should be extended when creating a `org.codice.ddf.migration.Migratable`.

# 24. Extending DDF Security

Version: 2.9.0

This section supports developers creating extensions of the existing framework.

## 24.1. Whitelist

The following packages have been exported by the DDF 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.canceler
- 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

## 24.2. 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.

# 25. Extending DDF Solr

Version: 2.9.0

## 25.1. Whitelist

The following packages have been exported by the DDF Solr application and are approved for use by third parties:

None.

# 26. Extending DDF Spatial

Version: 2.9.0

## 26.1. Whitelist

The following packages have been exported by the DDF 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`

# 27. Extending DDF Search UI

Version: 2.9.0

This section supports developers creating extensions of the existing framework.

## 27.1. Whitelist

The following packages have been exported by the DDF Search UI application and are approved for use by third parties:

None.