

1. DDF Home	5
1.1 Documentation Guide	12
1.1.1 Introduction	12
1.1.2 Standards Supported	13
1.1.3 Available Endpoints	13
1.1.4 Applications	13
1.1.5 Choosing a Guide	14
1.1.6 Documentation Downloads	17
1.2 DDF User's Guide	17
1.2.1 Understanding Metadata and Metacards	17
1.2.2 Searching Metadata	17
1.3 DDF Administrator's Guide	19
1.3.1 Installing DDF	20
1.3.1.1 Installing and Uninstalling Applications	27
1.3.2 Configuring DDF	29
1.3.2.1 Configuring via the Admin Console	29
1.3.2.2 Configuring DDF Using the Web Console	35
1.3.2.3 Configuring DDF Using the System Console	37
1.3.2.4 Configuring DDF Using Configuration (.cfg) Files	40
1.3.2.4.1 Enabling SSL	41
1.3.2.4.2 HTTP Port Configuration	44
1.3.2.4.3 Use Port 80 as a Non-Root User	44
1.3.2.4.4 Enable HTTP Access Logging	47
1.3.2.5 Configuring a Java Keystore for Secure Communications	48
1.3.2.6 Configuring Solr Catalog Provider	50
1.3.3 Configuring Catalog Provider	51
1.3.4 Configuring Notifications	51
1.3.5 Managing Web Service Security	53
1.3.5.1 Auditing	55
1.3.5.2 CAS SSO Configuration	63
1.3.5.2.1 Configure CAS for LDAP	67
1.3.5.2.2 Configure CAS for X509 User Certificates	70
1.3.5.3 Certificate Management	75
1.3.5.3.1 Cert File Management	75
1.3.5.3.2 Certificate Configuration Management	78
1.3.5.4 Encryption Service	83
1.3.5.5 Redaction and Filtering	83
1.3.5.6 Security Token Service	85
1.3.5.6.1 BinarySecurityToken (CAS) SAML Security Token Request/Response	95
1.3.5.6.2 UsernameToken Bearer SAML Security Token Request/Response	103
1.3.5.6.3 X.509 PublicKey SAML Security Token Request/Response	110
1.3.5.7 XACML Policy Decision Point (PDP)	120
1.3.5.8 Expansion Service	132
1.3.5.9 Configure WSS Using Standalone Servers	135
1.3.5.10 Hardening	137
1.3.5.11 Directory Permissions	139
1.3.5.12 Deployment Guidelines	140
1.3.5.13 Assuring Authenticity	141
1.3.5.14 Security	153
1.3.6 Starting DDF	153
1.3.7 Console Commands	157
1.3.7.1 Catalog Commands	158
1.3.7.2 Command Scheduler	163
1.3.7.3 Subscriptions Commands	164
1.3.7.4 Application Commands	171
1.3.7.5 Platform Commands	175
1.3.7.6 Persistence Commands	176
1.3.8 Ingesting Data	178
1.3.9 Appendix	179
1.3.9.1 DDF Directory Contents after Installation	179
1.3.9.2 Software Versioning	180
1.3.9.3 Troubleshooting	180
1.3.9.3.1 Blank Web Console	181
1.3.9.3.2 CXF BusException	181
1.3.9.3.3 Distribution Will Not Start	182
1.3.9.3.4 Exception Starting DDF	182
1.3.9.3.5 DDF Is Unresponsive to Incoming Requests	182
1.4 DDF Integrator's Guide	183
1.4.1 Quick Start	184
1.4.2 Using the CometD Endpoint for Asynchronous Communication	186
1.4.2.1 Asynchronous Notifications and Activities	188
1.4.2.2 Asynchronous Queries	192

1.5 DDF Developer's Guide	197
1.5.1 Building DDF	197
1.5.2 DDF Development Prerequisites	201
1.5.3 Formatting Source Code	202
1.5.4 Ensuring Compatibility	205
1.5.5 Development Recommendations	207
1.5.5.1 Fat Bundles	208
1.5.5.2 OSGi Services	210
1.5.6 UI Development Recommendations	214
1.5.7 "White Box" DDF Architecture	214
1.5.8 Developing DDF Applications	216
1.5.9 OGC Filter with DDF Frequently Asked Questions	218
1.5.10 Utilities	219
1.5.10.1 DDF-libs	219
1.5.10.1.1 DDF HTTP Proxy	219
1.5.10.2 DDF Load Balancer	222
1.5.10.3 DDF STOMP	225
1.6 DDF Applications	230
1.6.1 DDF Administrative Application	231
1.6.1.1 Managing DDF Administrative Application	232
1.6.1.1.1 Modules	234
1.6.1.1.2 Admin Console Access Control	238
1.6.1.2 Integrating with the Admin Application Service	239
1.6.1.3 Extending DDF Administrative Application	243
1.6.1.4 Securing DDF Administrative Application	243
1.6.1.5 DDF Admin Release Notes	243
1.6.1.5.1 Release Version: ddf-admin-1.0.1	243
1.6.1.5.2 Release Version: ddf-admin-1.1.1	244
1.6.2 DDF Catalog Application	244
1.6.2.1 Using DDF Catalog Application	245
1.6.2.2 Managing DDF Catalog Application	246
1.6.2.2.1 DDF Catalog Application Install and Uninstall	246
1.6.2.2.2 Federation UI	248
1.6.2.3 Integrating DDF Catalog Application	251
1.6.2.3.1 Integrating Endpoints	251
1.6.2.3.2 DDF Data Migration	258
1.6.2.3.3 Integrating Catalog Framework	259
1.6.2.3.4 Developer Use of OGC Filter	262
1.6.2.4 Extending DDF Catalog Application	265
1.6.2.4.1 Catalog Application Services	266
1.6.2.4.2 Catalog Development Fundamentals	267
1.6.2.4.3 Extending Catalog Plugins	279
1.6.2.4.4 Extending Operations	291
1.6.2.4.5 Extending Data and Metadata Basics	292
1.6.2.4.6 Extending Catalog Framework	296
1.6.2.4.7 Extending Sources	303
1.6.2.4.8 Extending Catalog Transformers	312
1.6.2.4.9 Extending Federation	341
1.6.2.4.10 Extending Eventing	343
1.6.2.4.11 Extending Resource Components	349
1.6.2.4.12 Developing Catalog Components	357
1.6.2.4.13 Javadocs	358
1.6.2.5 Securing DDF Catalog Application	358
1.6.2.6 DDF Catalog Application Release Notes	359
1.6.2.6.1 Release Version: catalog-2.6.1	359
1.6.2.6.2 Release Version: catalog-2.5.0	360
1.6.2.6.3 Release Version: ddf-catalog-2.4.1	366
1.6.3 DDF Content Application	367
1.6.3.1 Managing DDF Content Application	368
1.6.3.1.1 Installing DDF Content Application	368
1.6.3.2 Integrating DDF Content Application	370
1.6.3.2.1 DDF Content Framework	370
1.6.3.2.2 DDF Content Core	373
1.6.3.2.3 DDF Content REST CRUD Endpoint	377
1.6.3.3 Extending DDF Content Application	384
1.6.3.4 Securing DDF Content Application	384
1.6.3.5 DDF Content Application Release Notes	385
1.6.3.5.1 Release Version: content-2.4.0	385
1.6.3.5.2 Release Version: content-2.4.2	385
1.6.4 DDF Platform Application	386
1.6.4.1 Managing DDF Platform Application	386
1.6.4.1.1 Installing DDF Platform Application	387

1.6.4.2 Integrating DDF Platform Application	389
1.6.4.2.1 Platform Global Settings	389
1.6.4.2.2 DDF Mime Framework	390
1.6.4.2.3 Metrics Collection	392
1.6.4.2.4 Metrics Reporting Application	394
1.6.4.2.5 Security Core API	401
1.6.4.2.6 Compression Services	402
1.6.4.3 Extending DDF Platform Application	403
1.6.4.3.1 Developing Action Components (Action Framework)	403
1.6.4.4 Securing DDF Platform Application	404
1.6.4.5 DDF Platform Application Release Notes	405
1.6.4.5.1 Release Version: platform-2.3.1	405
1.6.4.5.2 Release Version: ddf-platform-2.4.0	409
1.6.4.5.3 Release Version: platform-2.5.0	412
1.6.4.5.4 Release Version: platform-2.5.1	414
1.6.4.5.5 Release Version: platform-2.6.1	416
1.6.5 DDF Registry Application	417
1.6.5.1 Managing DDF Registry Application	417
1.6.5.2 Integrating DDF Registry Application	418
1.6.5.3 Extending DDF Registry Application	418
1.6.5.4 Securing DDF Registry Application	418
1.6.5.5 DDF Registry Application Release Notes	418
1.6.6 DDF Security Application	418
1.6.6.1 Managing DDF Security Application	419
1.6.6.1.1 DDF Security Application Install and Uninstall	420
1.6.6.2 Integrating DDF Security Application	421
1.6.6.2.1 Security CAS	422
1.6.6.2.2 Security Core	425
1.6.6.2.3 Security PDP	442
1.6.6.2.4 Anonymous Interceptor	444
1.6.6.3 Extending DDF Security Application	445
1.6.6.3.1 Developing Token Validators	445
1.6.6.4 Securing DDF Security Application	446
1.6.6.5 DDF Security Application Release Notes	446
1.6.6.5.1 Release Version: security-2.5.1	446
1.6.7 DDF Solr Catalog Application	447
1.6.7.1 Managing DDF Solr Catalog Application	447
1.6.7.2 Integrating DDF Solr Catalog Application	448
1.6.7.2.1 DDF Catalog Solr Embedded Provider	449
1.6.7.2.2 DDF Catalog Solr External Provider	452
1.6.7.2.3 Standalone Solr Server	455
1.6.7.3 Extending DDF Solr Catalog Application	458
1.6.7.4 Securing DDF Solr Catalog Application	458
1.6.7.5 DDF Solr Catalog Application Release Notes	458
1.6.7.5.1 Release Version: solr-2.4.2	458
1.6.7.5.2 Release Version: solr-2.4.4	460
1.6.8 DDF Spatial Application	461
1.6.8.1 Managing DDF Spatial Application	461
1.6.8.1.1 Installing DDF Spatial Application	461
1.6.8.2 Integrating DDF Spatial Application	463
1.6.8.2.1 Integrating DDF with CSW	463
1.6.8.2.2 Integrating DDF with KML	481
1.6.8.2.3 Integrating DDF with WFS	494
1.6.8.3 Extending DDF Spatial Application	499
1.6.8.4 Securing DDF Spatial Application	499
1.6.8.5 DDF Spatial Application Release Notes	500
1.6.8.5.1 Release Version: spatial-2.6.1	500
1.6.9 DDF Standard Search UI	500
1.6.9.1 Using DDF Standard Search UI	501
1.6.9.2 Managing DDF Standard Search UI	518
1.6.9.2.1 Installing DDF Standard Search UI	518
1.6.9.2.2 Troubleshooting DDF Standard Search UI	521
1.6.9.3 Integrating DDF Standard Search UI	522
1.6.9.4 Extending DDF Standard Search UI	522
1.6.9.5 Securing DDF Standard Search UI	522
1.6.9.6 DDF Standard Search UI Release Notes	522
1.6.9.6.1 Release Version: ui-2.6.1	522
1.7 DDF Release Versions	523
1.7.1 DDF 2.6.0 Release Versions	523
1.8 How-to articles	525
1.8.1 How to Change the Solr Cache Expiration Interval	526
1.8.2 Replace "localhost" with System Certificates	526

1.9 Index	527
1.10 Community	539
1.10.1 Team	539
1.10.2 Community Contributions	540
1.10.3 How to become active in the DDF Community	540
1.10.4 Decision Making	541
1.10.5 Release Guide	541
1.10.6 DDF Roadmap	544
1.10.7 DDF Static Analysis and Build	544
2. CSW Endpoint	545
3. Wiki Page Template	545
4. Administrator's Quick Start	548
5. CSW Source	551

DDF Home

Welcome to the home of the DDF.

The [Quick Start](#) describes how to get up and running quickly.

Overview

Distributed Data Framework (DDF) is an agile and modular integration framework. It is primarily focused on data integration, enabling clients to insert, query and transform information from disparate data sources via the DDF Catalog. A Catalog API allows integrators to insert new capabilities at various stages throughout each operation. DDF is designed with the following architectural qualities to benefit integrators.

Standardization

- Building on established Free and Open Source Software (FOSS) and open standards avoids vendor lock-in

Extensibility

- System Integrators can extend capabilities by developing and sharing new features

Flexibility

- System Integrators may deploy only those features required

Simplicity of Installation and Operation

- Unzip and run
- Configuration via a web console
- Simplicity of Development
 - Build simple Plain Old Java Objects (POJOs) and wire them in via a choice of dependency injection frameworks
 - Make use of widely available documentation and components for DDF's underlying technologies
 - Modular development supports multi-organizational and multi-regional teams

Included Applications

[DDF Administrative Application](#)
[DDF Catalog Application](#)
[DDF Content Application](#)
[DDF Platform Application](#)
[DDF Security Application](#)
[DDF Solr Catalog Application](#)
[DDF Spatial Application](#)
[DDF Standard Search UI](#)

Please visit the [DDF Administrator's Guide](#) and the [DDF Developer's Guide](#) for more information.

Building

On This Page

- Prerequisites
- Procedures
 - Run the Build
 - Troubleshoot Build Errors on ddf-admin and ddf-ui on a Windows Platform

Prerequisites

- Install J2SE 7 SDK (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
- 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 Maven 3.0.4 or later (<http://maven.apache.org/download.cgi>). 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 (<http://maven.apache.org/settings.html>).

Sample settings.xml file

```
<settings>
  <!-- If proxy is needed
  <proxies>
    <proxy>
      </proxy>
    </proxies>
  -->
</settings>
```

Handy Tip on Encrypting Passwords

See this Maven [guide](#) on how to encrypt the passwords in your settings.xml.

Procedures

Run the Build

In order to run through a full build, be sure to have a clone for all repositories in the same folder:

- ddf (<https://github.com/codice/ddf.git>)
- ddf-admin (<https://github.com/codice/ddf-admin.git>)
- ddf-catalog (<https://github.com/codice/ddf-catalog.git>)
- ddf-content (<https://github.com/codice/ddf-content.git>)
- ddf-parent (<https://github.com/codice/ddf-parent.git>)
- ddf-platform (<https://github.com/codice/ddf-platform.git>)
- ddf-security (<https://github.com/codice/ddf-security.git>)

- ddf-solr (<https://github.com/codice/ddf-solr.git>)
- ddf-spatial (<https://github.com/codice/ddf-spatial.git>)
- ddf-support (<https://github.com/codice/ddf-support.git>)
- ddf-ui (<https://github.com/codice/ddf-ui.git>)

- 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.
```

- Build command example to build all repositories. This must be performed at the top level folder that contains all the repositories. A command list would look like this:

```
# Build is run from the top level folder  
that contains all the repositories in a  
command line prompt or terminal.  
  
cd ddf-support  
mvn clean install  
  
cd ../ddf-parent  
mvn clean install  
  
cd ../ddf-platform  
mvn clean install  
  
cd ../ddf-admin  
mvn clean install  
  
cd ../ddf-security  
mvn clean install  
  
cd ../ddf-catalog  
mvn clean install  
  
cd ../ddf-content  
mvn clean install  
  
cd ../ddf-spatial  
mvn clean install  
  
cd ../ddf-solr  
mvn clean install  
  
cd ../ddf-ui  
mvn clean install  
  
cd ../ddf  
mvn clean install  
  
# This will fully compile each individual  
app. From here you may hot deploy the  
necessary apps on top of the DDF Kernel.
```

To use the updated apps in a DDF distribution, update the versions referenced in the "ddf" repository.

The zip distribution of DDF is contained in the DDF app in the distribution/ddf/target directory after the DDF app is built.

- Optionally, create a reactor pom that will allow you to perform the entire build process by calling a build on one pom rather than all of them. This pom must reside in the top-level folder that holds all the repositories. An example of the file would be:

```
<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">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.codice.ddf</groupId>
  <artifactId>reactor</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>DDF Reactor</name>
  <description>Distributed Data Framework (DDF) is an open source, modular integration framework</description>

  <modules>
    <module>ddf-support</module>
    <module>ddf-parent</module>
    <module>ddf-platform</module>
      <module>ddf-admin</module>
    <module>ddf-security</module>
    <module>ddf-catalog</module>
    <module>ddf-content</module>
    <module>ddf-spatial</module>
    <module>ddf-solr</module>
    <module>ddf-ui</module>
    <module>ddf</module>
  </modules>

  <build>
    <plugins>
      <plugin>

        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.4</version>
        <configuration>
          <!-- Do not deploy the reactor pom -->
          <skip>true</skip>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

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.

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 -Xmx512m -XX:MaxPermSize=256m before running the build.

Example on Linux system with the bash shell: export MAVEN_OPTS='-Xmx512m -XX:MaxPermSize=256m'

Troubleshoot 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.

```
bower cache clean  
npm cache clean
```

3. Reinstall bower. An occasional reinstall may solve the issue.

```
npm uninstall -g bower && npm install -g  
bower
```

4. 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.

How to Run

- * Unzip the distribution.
- * Run the executable at <distribution_home>/bin/ddf.bat or <distribution_home>/bin/ddf

Additional Information

The [wiki](#) is the right place to find any documentation about DDF.

Discussions can be found on the [Announcements forum](#), [Users forum](#), and [Developers forum](#).

For a DDF binary distribution, please read the release notes on the wiki for a list of supported and unsupported features.

If you find any issues with DDF, please submit reports with JIRA: <https://tools.codice.org/jira/browse/DDF>

For information on contributing to DDF see: <http://www.codice.org/contributing>.

The Website contains additional information at <http://ddf.codice.org>

Many thanks for using DDF.

-- The Codice DDF Development Team

Recently Updated

-  [Configuring via the Admin Console](#)
Jul 07, 2015 • updated by [Rick Larsen](#) • [view change](#)
-  [Installing DDF](#)
Jul 07, 2015 • updated by [Rick Larsen](#) • [view change](#)
-  [Troubleshooting DDF Standard Search UI](#)
Mar 20, 2015 • updated by [William Miller](#) • [view change](#)
-  [Standalone Solr Server](#)
Mar 16, 2015 • updated by [Jason Kilmer](#) • [view change](#)
-  [DDF Catalog Solr Embedded Provider](#)
Mar 16, 2015 • updated by [Jason Kilmer](#) • [view change](#)
-  [DDF Catalog Solr External Provider](#)
Mar 16, 2015 • updated by [Jason Kilmer](#) • [view change](#)
-  [Integrating DDF Solr Catalog Application](#)
Mar 12, 2015 • updated by [Jason Kilmer](#) • [view change](#)
-  [Security STS LDAP Claims Handler](#)
Mar 04, 2015 • updated by [Jason Kilmer](#) • [view change](#)
-  [Extending Federation](#)
Mar 04, 2015 • updated by [Stephen Swartz](#) • [view change](#)
-  [Using DDF Standard Search UI](#)
Jan 27, 2015 • updated by [William Miller](#) • [view change](#)
-  [Quick Start](#)
Jan 24, 2015 • updated by [Shaun Morris](#) • [view change](#)
-  [Release Version: spatial-2.6.1](#)
Jan 24, 2015 • updated by [Shaun Morris](#) • [view change](#)

-  DDF 2.6.X
Jan 24, 2015 • created by [Shaun Morris](#)
-  DDF26X
Jan 24, 2015 • attached by [Shaun Morris](#)
-  Release Version: content-2.4.2
Jan 24, 2015 • created by [Shaun Morris](#)

Documentation Guide

Overview

This document serves to guide users, administrators and developers through the DDF.

Documentation Updates

The most current Distributed Data Framework (DDF) documentation is available at <https://tools.codice.org/wiki/display/DDF/>.

Questions

Questions about DDF or this documentation should be posted to the ddf-users forum (<https://groups.google.com/d/forum/ddf-users>), ddf-announcements forum (<https://groups.google.com/d/forum/ddf-announcements>), or ddf-developers forum (<https://groups.google.com/d/forum/ddf-developers>) where they will be responded to quickly by a member of the DDF team.

Conventions

The following conventions are used within this documentation:

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

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

This is an **Emphasized Note**, used to inform of important information.

This is a **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: `ddf.catalog.CatalogFramework`

Hyperlinks

Some hyperlinks (e.g., </system/console>) within the documentation assume a locally running installation of DDF. Simply change the hostname if accessing a remote host.

Introduction

Overview

This documentation

Standards Supported

The following Standards are currently supported:

Content by label

There is no content with the specified labels



Available Endpoints

The following Endpoints are currently available:

Applications

Overview

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

Applications

Applications

- **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 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 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 Catalog Application** — 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.
- **DDF Content Application** — The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data.
- **DDF Content Application** — The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data.
- **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. These building blocks include subsets of:
 - Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
 - Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
 - Camel (<http://camel.apache.org/>).

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

- **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. These building blocks include subsets of:
 - Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
 - Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
 - Camel (<http://camel.apache.org/>).

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

- **DDF Security Application** — 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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.
- **DDF Security Application** — 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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.
- **DDF Solr Catalog Application** — The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store.
- **DDF Solr Catalog Application** — The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/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 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 in HTML format and are displayed on a globe, providing a visual representation of where the records were found.

Choosing a Guide

Overview

The documentation is segmented 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.

- [Using DDF Catalog Application](#)
- [Using DDF Catalog Application](#)
- [Using DDF Standard Search UI](#)
- [Using DDF Standard Search UI](#)

Administrators

Administrators will be installing, maintaining, and supporting existing applications.

- [Managing DDF Platform Application](#) — provides the fundamental building blocks that the DDF distribution needs in order to run, including subsets of Karaf, CXF, Cellar, and Camel
- [Managing DDF Standard Search UI](#) — The Standard Search UI is a user interface that enables users to search a catalog and associated sites for content and metadata.
- [DDF User's Guide](#)
- [Managing DDF Content Application](#) — The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data. This guide documents the installation, maintenance, and support of this application.
- [DDF Administrator's Guide](#)
- [Managing DDF Security Application](#) — provides authentication, authorization, and auditing services for the DDF
- [Managing DDF Solr Catalog Application](#) — provides an implementation of the CatalogProvider interface using Apache Solr as a data store
- [Managing DDF Catalog Application](#) — describes the DDF Catalog application and available options for extending its capabilities

- [Managing DDF Administrative Application](#)
- [Managing 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.
- [Managing DDF Standard Search UI](#) — The Standard Search UI is a user interface that enables users to search a catalog and associated sites for content and metadata.
- [Managing 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.
- [Managing DDF Platform Application](#) — provides the fundamental building blocks that the DDF distribution needs in order to run, including subsets of Karaf, CXF, Cellar, and Camel
- [Managing DDF Administrative Application](#)
- [Managing DDF Solr Catalog Application](#) — provides an implementation of the CatalogProvider interface using Apache [Solr](#) as a data store

Integrators

Integrators will use the existing applications to support their external frameworks.

- [DDF Integrator's Guide](#) — This guide provides instructions for configuring, maintaining and operation components of the Distributed Data Framework.
- [DDF Integrator's Guide](#) — This guide provides instructions for configuring, maintaining and operating components of the Distributed Data Framework.
- [Integrating DDF Catalog Application](#)
- [Integrating DDF Catalog Application](#)
- [Integrating DDF Content Application](#) — The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data. This guide provides instructions for configuring, maintaining and operation components of the Distributed Data Framework.
- [Integrating DDF Content Application](#) — The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data. This guide provides instructions for configuring, maintaining and operating components of the Distributed Data Framework.
- [Integrating 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. These building blocks include subsets of:
 - [Karaf](#) (<http://karaf.apache.org/>), [CXF](#) (<http://cxf.apache.org/>),
 - [Cellar](#) (<http://karaf.apache.org/index/subprojects/cellar.html>), and
 - [Camel](#) (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

This guide supports integration of this application with external frameworks.

- [Integrating 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. These building blocks include subsets of:
 - [Karaf](#) (<http://karaf.apache.org/>), [CXF](#) (<http://cxf.apache.org/>),
 - [Cellar](#) (<http://karaf.apache.org/index/subprojects/cellar.html>), and
 - [Camel](#) (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

This guide supports integration of this application with external frameworks.

- [Integrating DDF Security Application](#) — 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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide supports integration of this application with external frameworks.

- [Integrating DDF Security Application](#) — 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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide supports integration of this application with external frameworks.

- [Integrating DDF Solr Catalog Application](#) — The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store.

This guide supports integration of this application with external frameworks.

- [Integrating DDF Solr Catalog Application](#) — The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store.

This guide supports integration of this application with external frameworks.

- [Integrating DDF Spatial Application](#)
- [Integrating DDF Spatial Application](#)
- [Integrating DDF Standard Search UI](#)

Developers

Developers will build or extend the functionality of the applications.

- [DDF Developer's Guide](#)
- [DDF Developer's Guide](#)
- [Extending DDF Administrative Application](#)
- [Extending DDF Administrative Application](#)
- [Extending DDF Catalog Application](#)
- [Extending DDF Catalog Application](#)
- [Extending DDF Content Application](#)
- [Extending DDF Content Application](#)
- [Extending DDF Platform Application](#)
- [Extending DDF Platform Application](#)
- [Extending DDF Security Application](#)
- [Extending DDF Security Application](#)
- [Extending DDF Solr Catalog Application](#)
- [Extending DDF Solr Catalog Application](#)
- [Extending DDF Spatial Application](#)

Security

- [Securing DDF Administrative Application](#)
- [Securing DDF Administrative Application](#)
- [Securing DDF Catalog Application](#)
- [Securing DDF Catalog Application](#)
- [Securing DDF Content Application](#)
- [Securing DDF Content Application](#)
- [Securing DDF Security Application](#) — 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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide covers implementations and protocols for enhancing security.

- [Securing DDF Security Application](#) — 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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide covers implementations and protocols for enhancing security.

- [Securing DDF Solr Catalog Application](#) — The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store. This guide covers implementations and protocols for enhancing security.

- [Securing DDF Solr Catalog Application](#) — The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store. This guide covers implementations and protocols for enhancing security.
- [Securing DDF Spatial Application](#)
- [Securing DDF Spatial Application](#)
- [Securing DDF Standard Search UI](#)
- [Securing DDF Standard Search UI](#)

Documentation Downloads

Overview

This page contains complete documentation for each of the DDF application in a downloadable, printable pdf format. Get [Adobe Reader](#).

Downloads

- [Admin.pdf](#)
- [Catalog.pdf](#)
- [Content.pdf](#)
- [Platform.pdf](#)
- [Registry.pdf](#)
- [Security.pdf](#)
- [Solr-Catalog.pdf](#)
- [Spatial.pdf](#)
- [Standard-Search-UI.pdf](#)

DDF User's Guide

Overview

This guide provides instructions to install, start, and stop the DDF .

Distributed Data Framework (DDF) is an agile and modular integration framework. It is primarily focused on data integration, enabling clients to insert, query and transform information from disparate data sources via the DDF Catalog. A Catalog API allows integrators to insert new capabilities at various stages throughout each operation. DDF is designed with the following architectural qualities to benefit integrators.

Contents

- [Understanding Metadata and Metacards](#)
- [Searching Metadata](#) — This section provides a very high level overview of introductory concepts of searching with ApplicationName. These concepts are expanded upon in later sections.

Understanding Metadata and Metacards

Overview

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.

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.

Searching Metadata

Overview

DDF provides the capability to search the 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.

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.

Search Interfaces

DDF Search UI Application

The DDF Search UI application provides a graphic interface to return results in HTML format and locate them on an interactive globe or map. For more details on using this application, go to [DDF Search UI User's Guide](#).

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.

Catalog Search Result Objects

Data is returned from searches as Catalog Search Result objects. This is a subtype of Catalog Entry that also contains additional data based on what type of sort policy was applied to the search.

Because it is a subtype of Catalog Entry, a Catalog Search Result has all Catalog Entry's fields such as metadata, effective time, and modified time. It also contains some of the following fields, depending on type of search, that are populated by DDF when the search occurs:

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

Search Programmatic Flow

Searching the catalog involves three basic steps:

1. Define the search criteria (contextual, spatial, 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

These steps are performed in the same basic order but using different classes depending on whether the Web services or Search UI interfaces are used.

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.

Sort Policy	Sorts By	Default Order	Available for
Temporal	The catalog search result's <code>effectiveTime</code> field	Newest to oldest	All Search Types
Distance	The catalog search result's <code>distance</code> field	Nearest to farthest	Point-Radius Spatial searches
Relevance	The catalog search result's <code>relevance</code> field	Most to least relevant	Contextual

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

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.*

DDF Administrator's Guide

Introduction

Overview

This guide provides instructions to install, start, and stop the DDF .

Distributed Data Framework (DDF) is an agile and modular integration framework. It is primarily focused on data integration, enabling clients to insert, query and transform information from disparate data sources via the DDF Catalog. A Catalog API allows integrators to insert new capabilities at various stages throughout each operation. DDF is designed with the following architectural qualities to benefit integrators.

Contents

- [Installing DDF](#) — steps to install the ApplicationName.
- [Configuring DDF](#)
- [Configuring Catalog Provider](#)

- **Configuring Notifications** — Notifications are messages that are sent to clients to inform them of some significant event happening in ApplicationName. Clients must subscribe to a ApplicationName notification channel to receive these messages.
- **Managing Web Service Security** — The Web Service Security (WSS) functionality that comes with DDF is integrated throughout the system. This page acts as a central point to show how all of the pieces work together and point out where they live inside of the system.
- **Starting DDF** — steps to start and stop DDF.
- **Console Commands**
- **Ingesting Data**
- **Appendix** — Supplemental information for ApplicationName appendix.

Installing DDF

On This Page

- Overview
- Prerequisites
- Installation Guide
 - Use the DDF Distribution Zip to Install
 - (Option 1/Preferred Method) Continue Setup and Installation Using the Installer Module of the Admin UI
 - (Option 2/Part 1) Custom Installation Using the DDF Kernel Distribution Zip
 - (Option 2/Part 2) Configuration
 - Verification
- Troubleshooting
- Exception Starting DDF
 - Problem:
 - Solution:

Overview

Follow the below steps to install the DDF.

Prerequisites

- Supported platforms are *NIX - Unix/Linux/OSX, Solaris, and Windows.
- JDK7 must be installed (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>).
- The JAVA_HOME environment variable must be set to the location where the JDK is installed.

Example of Setting Up on *NIX

```
JAVA_HOME=/usr/java/jdk1.7.0
export JAVA_HOME
```

Example of Setting Up JAVA_HOME on Windows

```
set JAVA_HOME=C:\Program Files\Java\jdk1.7.0
```

*NIX

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

```
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
```

Restart

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

Restart Command

```
init 6
```

The Administration Web Console is not compatible with Internet Explorer.

Installation Guide

The *NIX commands listed next to the steps were performed on a default installation of Red Hat Enterprise Linux 5.4. Permission maintenance mentioned in this article, but all files should be owned by the running user, regardless of platform.

DDF Installation Types

DDF can be installed using a single distribution zip file that contains all of the DDF applications already installed,

OR

A custom DDF installation can be performed by using the DDF kernel distribution zip file then hot deploying the desired DDF apps into the <INSTALL_DIRECTORY>/deploy directory.

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

Use the DDF Distribution Zip to Install

1. 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**

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-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
```

2. 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:\
```

3. Copy ddf-X.Y.zip to <INSTALL_DIRECTORY>.

***NIX**

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

Windows

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

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

***NIX or Windows**

```
cd <INSTALL_DIRECTORY>
```

5. 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\jdk1.7.0\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
```

- Wait for the console prompt to appear.

Command Prompt when Initially Loaded

```
ddf@local>
```

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

View Status

```
ddf@local>list
```

- Proceed to Configuration.

(Option 1/Preferred Method) Continue Setup and Installation Using the Installer Module of the Admin UI

Installer Module.

(Option 2/Part 1) Custom Installation Using the DDF Kernel 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>.

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
```

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

Example: Where the Zip File may be Located

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

Windows (Example Assumes DDF has been Downloaded to the D Drive)

```
cd D:\
```

3. Copy ddf-kernel-X.Y.zip to <INSTALL_DIRECTORY>.

***NIX**

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

Windows

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

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

***NIX or Windows**

```
cd <INSTALL_DIRECTORY>
```

5. The DDF kernel zip is now located within the <INSTALL_DIRECTORY>. Unzip ddf-kernel-X.Y.zip.

***NIX**

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

Example: Using Java to Unzip in Windows

```
"C:\Program Files\Java\jdk1.7.0\bin\jar.exe" xf ddf-kernel-X.Y.zip
```

6. If the DDF Standalone Solr Server will be installed later, an additional configuration step is required for the DDF kernel. Add the following line to 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
```

7. Run the DDF kernel using the appropriate script.

*NIX

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

Windows

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

8. Wait for the console prompt to appear.

Command Prompt when Initially Loaded

```
ddf@local>
```

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

View Status

```
ddf@local>list
```

The list of bundles should look similar to this:

DDF Kernel 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)  
 [ 112] [Resolved]    ] [          ] [          ] [    80] DDF :: Distribution ::  
 Console (2.3.0)                                     Hosts: 76  
 [ 113] [Active]      ] [Created]    ] [          ] [    80] DDF :: Distribution ::  
 Branding Plugin (2.3.0)
```

9. Verify that the following DDF kernel's features are installed by executing the `features:list` command and filtering for kernel distribution.

DDF Kernel's Installed Features List

```
ddf@local>features:list | grep kernel
[uninstalled] [2.3.1-SNAPSHOT ] custom-karaf-bundles           kernel-2.3.1-SNAPS
Customized KARAF Bundles
[installed   ] [2.3.1-SNAPSHOT ] kernel-webconsolebranding kernel-2.3.1-SNAPS
Web Admin Console branding
```

DDF Application Installation Dependencies

Please read the installation instructions carefully for each DDF application because some of the applications depend on other DDF been previously installed.

The order the DDF applications are listed below is the recommended order of installation. Each application must be active before i application is installed. The app:list console command can be used to verify the state of each app as it is installed.

10. Install the DDF Platform application by following the [Platform Application Installation instructions](#).
11. Install any optional DDF applications that may be needed for the desired DDF configuration.
 - a. [DDF Catalog Application Installation instructions](#)
 - b. [DDF Security Application Installation instructions](#)
 - c. [DDF Content Application Installation instructions](#)
 - d. [DDF Spatial Application Installation instructions](#)
 - e. DDF Solr Catalog Application Installation instructions are included in each of the Solr Catalog configurations. Refer to the appropriate desired Solr Catalog Provider configuration's installation instructions.
 - i. [Embedded Solr Catalog Provider](#)
 - ii. [External Solr Catalog Provider](#)
 - iii. [Standalone Solr Server](#)
12. Proceed to Configuration.

(Option 2/Part 2) Configuration

1. Open a compatible web browser and log in to the Administrator Console (<http://localhost:8993/admin>) with username "admin" and password "admin".
2. Select **Platform App**
3. Select the **Configuration** tab.
 - a. Select **Platform Global Configuration**.
 - i. In the **Host** field, enter the IP address or hostname of the installation.
 - ii. In the **Port** field, enter 8181. The port can be changed later, if necessary.
 - iii. Enter the site name (e.g., the name DDF will return in federation and web service responses).
 - iv. Select the **Save** button.
 - b. Select **Catalog Sorted Federation Strategy**.
 - i. In the **Maximum start index** field, enter the maximum query offset number or keep the default setting. Refer to [Standard Federation](#) for additional information.
 - ii. Select the **Save** button.

Verification

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
```

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

osts, 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 ::  
Console (2.2.0) Hosts: 76
```

For a complete list of installed features/bundles see the [DDF Included Features](#) document.

Troubleshooting

Exception Starting DDF

Problem:

An exception is thrown starting DDF on a Windows machine (x86).

If using an unsupported terminal, `java.lang.NoClassDefFoundError: Could not initialize class org.fusesource.jansi.internal.Kernel32` is thrown.

Solution:

Install missing Windows libraries.

Some Windows platforms are missing libraries that are required by DDF. These libraries are provided by the Microsoft Visual C++ 2008 Redistributable Package x64 (<http://www.microsoft.com/en-us/download/details.aspx?id=15336>).

Installing and Uninstalling Applications

On This Page

- [Overview](#)
- [Install](#)
- [Verify](#)
- [Uninstall](#)
 - [Revert the Uninstall](#)
- [Upgrade](#)

Overview

This page provides the general instructions for installing, uninstalling, and upgrading a DDF application.

Specific instructions are provided for each DDF application under its wiki page, e.g., [DDF Applications](#) [DDF Catalog Application](#) [DDF Catalog Application Install/Uninstall](#).

Install

1. Before installing a DDF application, verify that its prerequisites have been met. For example, the DDF Catalog application cannot be installed until the DDF Kernel has been unzipped and installed, and the DDF Platform application has been installed. These prerequisites are identified in each of the specific application's install instructions.
2. Copy the DDF application's KAR file to the `<INSTALL_DIRECTORY>/deploy` directory.

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

Verify

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.

Refer to the individual application install/uninstall pages for details on which features and bundles should be installed specifically for that application.

Uninstall

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

```
features:removeurl -u <DDF application's feature repository URL>
```

Example: `features:removeurl -u mvn:ddf.catalog/catalog-app/2.3.0/xml/features`

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

The repository URLs for installed applications can be obtained by entering:

```
features:listrepositories -u
```

Revert 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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf.catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

Upgrade

Complete the following procedure to upgrade the DDF application.

1. Uninstall the application following the Uninstall instructions above.
2. Install the new DDF application's KAR file following the Install directions above.
3. Restart DDF.
4. Follow the Verification section above to determine if the upgrade was successful.

Configuring DDF

Overview

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

Configuration Methods

- Configuring via the Admin Console
- Configuring DDF Using the Web Console
- Configuring DDF Using the System Console
- Configuring DDF Using Configuration (.cfg) Files
- Configuring a Java Keystore for Secure Communications
- Configuring Solr Catalog Provider

Configuring via the Admin Console

Overview

This page supports updating configuration of DDF through the Administrator console.

On this page

- Overview
- Accessing the Admin Console
- Initial Configuration
- Viewing Currently Active Applications
 - Tile View
 - List View
 - Configuration
 - Details
 - Features
- Managing Applications
 - Activating / Deactivating Applications
 - Adding Applications
 - Removing Applications
 - Upgrading Applications
- System Settings Tab

Accessing the Admin Console

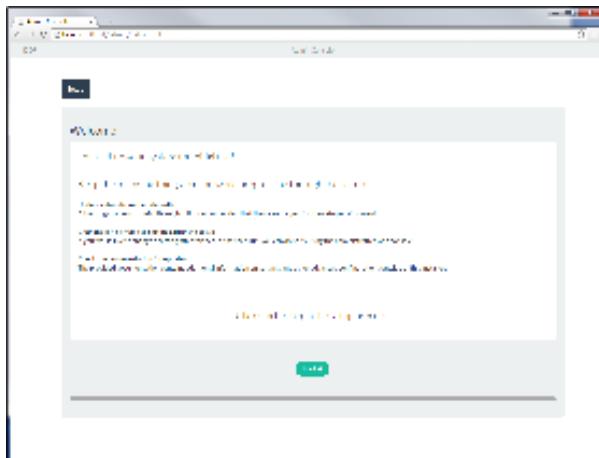
Open the admin portal.

- <https://localhost:8993/admin>
 - Enter Username and Password.

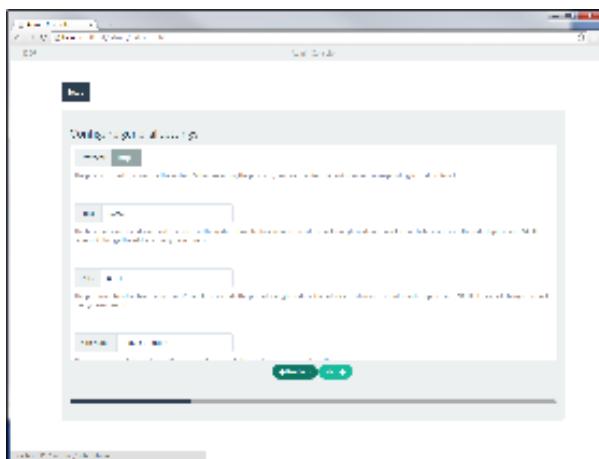
The default username/password is admin/admin. To change this, refer to [Password Management](#) page.

Initial Configuration

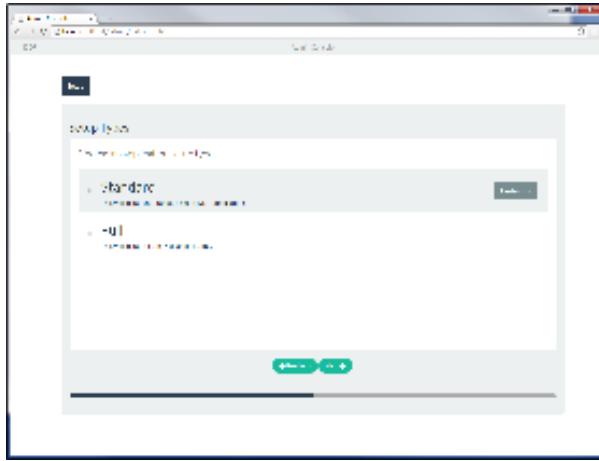
The first time the DDF administrator portal runs, the initial configuration steps appear. Click Start to begin.



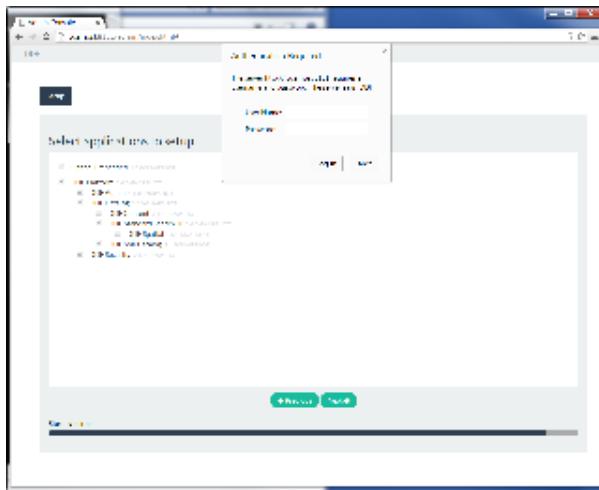
On the next screen, general configuration settings such as host address, port and site name can all be configured.



Next, choose between a standard installation and a full installation. Individual applications can be added, removed or deactivated later.



The final step of initial configuration is a display of all applications installed and their current status. This tree structure demonstrates how several applications depend on other applications.



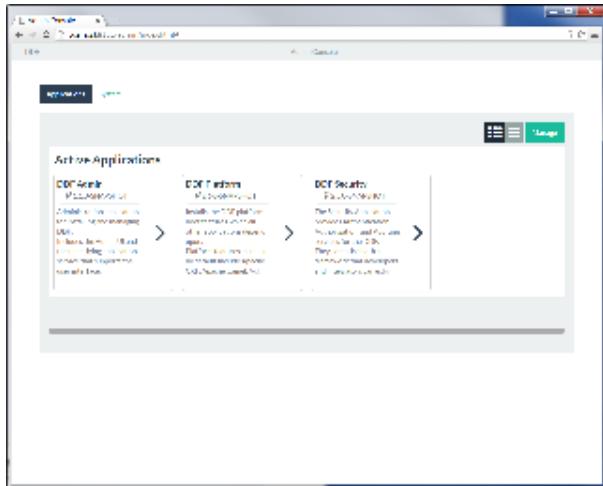
Platform App, Admin App, and Security Services App CANNOT be selected or unselected as it is installed by default and can cause errors if removed.

**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.

Viewing Currently Active Applications

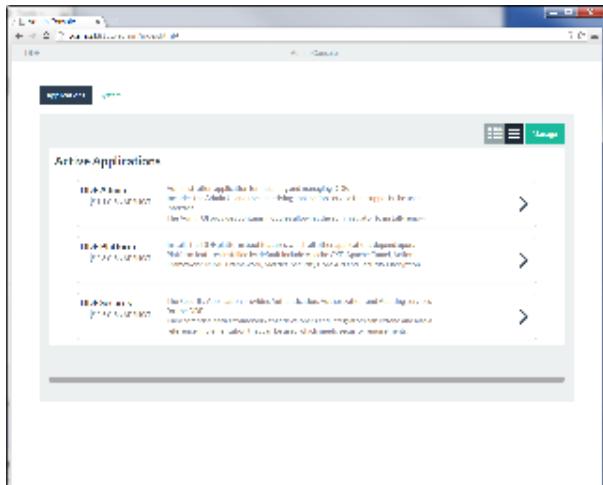
Tile View

The first view presented is the Tile View, displaying all active applications as individual tiles.



List View

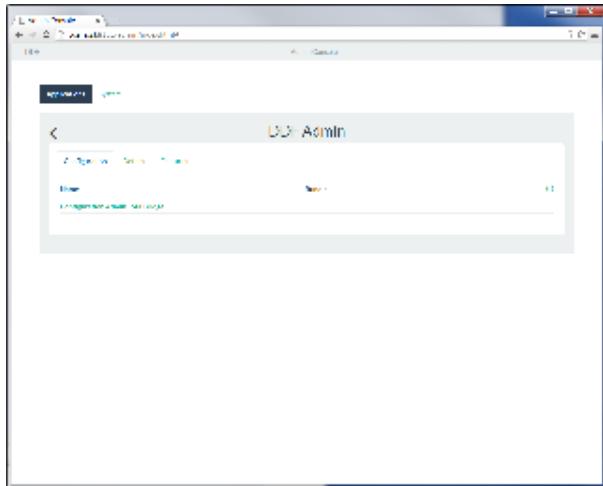
Optionally, active applications can be displayed in a list format by clicking the list view button.



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

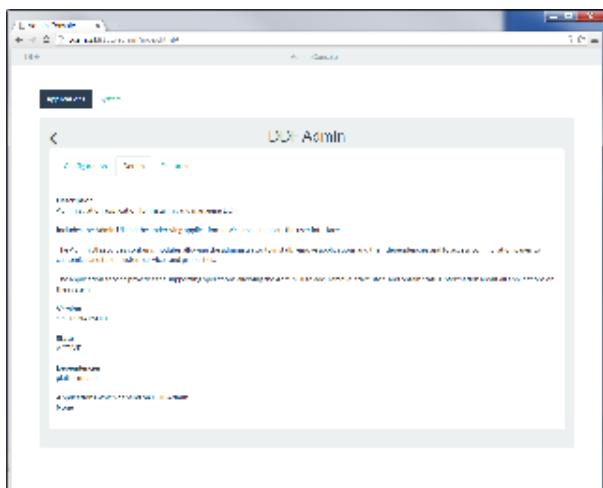
Configuration

The Configuration tab lists all bundles associated with the application as links to configure any configurable properties of that bundle.



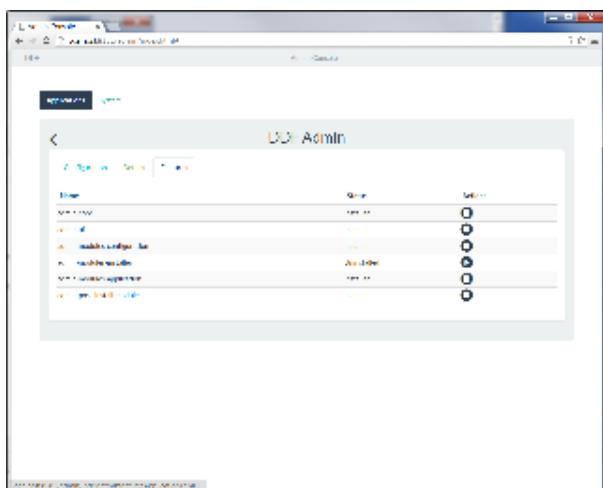
Details

The Details tab gives a description, version, status, and list of other applications that either required for , or rely on, the current application.



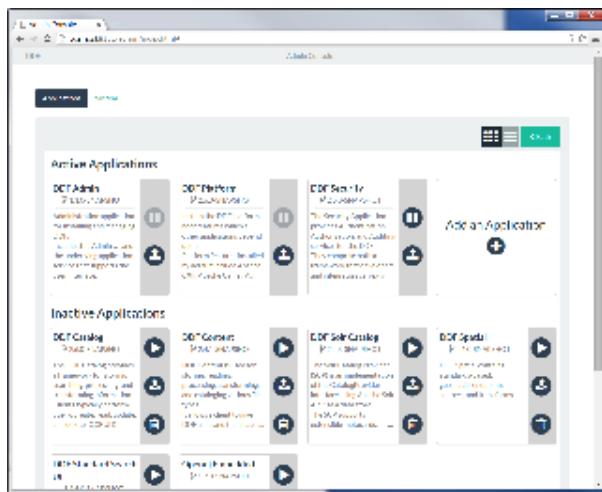
Features

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.



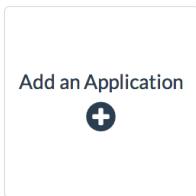
Activating / Deactivating Applications

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.

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

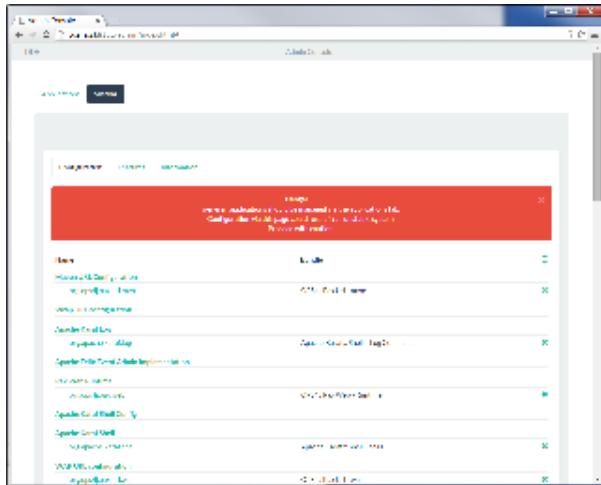
Each application tile includes an upgrade but to select a new version to install.

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.

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

Proceed with caution!



Configuring DDF Using the Web Console

On This Page

- Overview
- Access the Web Console
- Configure DDF Platform Global Settings
 - Configurable Properties
- Manage Features
 - Install Features Using the Web Console
 - Uninstall Features
 - Add Feature Repositories
- Known Issues
- Additional Information

Overview

Follow these steps to configure DDF from the web console.
configure DDF from the web console.

Access the Web Console

Open the web administration console.

- <http://localhost:8181/system/console>
- Enter Username and Password

The default username/password is admin/admin. To change this refer to the Hardening section.

Configure DDF Platform Global Settings

1. Open the web administration console.
 - <http://localhost:8181/system/console>
 - Enter Username and Password

The default username/password is admin/admin. To change this refer to the Hardening section.

2. Select the Configuration tab.

3. Select the **Platform Global Configuration** bundle.
4. Enter the DDF configuration values per the Configurable Properties table below.
5. Select the **Save** button.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Protocol	protocol	String	Default protocol that should be used to connect to this machine.	http	yes
Host	host	String	The hostname or IP address used to advertise the system. Do not enter localhost. Possibilities include the address of a single node or that of a load balancer in a multi-node deployment. NOTE: Does not change the address the system runs on.		yes
Port	port	String	The port used to advertise the system. Possibilities include the port of a single node or that of a load balancer in a multi-node deployment. NOTE: Does not change the port the system runs on.		yes
Trust Store	trustStore	String	The trust store used for outgoing SSL connections. Path is relative to ddf.home.	etc/keystores/clientTruststore.jks	no
Trust Store Password	trustStorePassword	String	The password associated with the trust store.	changeit (encrypted)	no
Key Store	keyStore	String	The key store used for outgoing SSL connections. Path is relative to karaf.home.	etc/keystores/clientKeystore.jks	no
Key Store Password	keyStorePassword	String	The password associated with the key store.	changeit (encrypted)	no
Site Name	id	String	The site name for DDF.	ddf.distribution	yes
Version	version	String	The version of DDF that is running. This value should not be changed from the factory default.	2.3.0	yes
Organization	organization	String	The organization responsible for this installation of DDF.	Codice Foundation	yes

Manage 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. For more information on the features that come with DDF, including a list of the ones included, consult the [DDF Included Features](#) page in the Software Version Description Document (SVDD).

Transitive Dependencies

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

Install Features Using the Web Console

1. Open the web administration console.
 - <http://localhost:8181/system/console>
 - Enter Username and Password

The default username/password is admin/admin. To change this refer to the [Hardening](#) section.

2. Select the Features tab.



3. Select the play arrow under the **Actions** column for the feature that should be installed.
4. Wait for the **Status** to change from **Uninstalled** to **Installed**.
5. Optional: Check the bundle status.
 - a. Select the Bundles tab.
 - b. Scroll down to the associated bundles and verify they were installed with a status of **Active**.

Uninstall Features

1. Open the web administration console.
 - <http://localhost:8181/system/console>
 - Enter Username and Password

The default username/password is admin/admin. To change this refer to the [Hardening](#) section.

2. Select the Features tab.



3. Select the eject icon under the **Actions** column for the feature that should be uninstalled.
4. Wait for the **Status** to change from **Installed** to **Uninstalled**.
5. Optional: Check the bundle status.
 - a. Select the Bundles tab.
 - b. Verify the absence of the associated bundle(s).

Add Feature Repositories

1. Open the web administration console.
 - <http://localhost:8181/system/console>
 - Enter Username and Password

The default username/password is admin/admin. To change this refer to the [Hardening](#) section.

2. Select the Features tab.
3. Enter the URL of the feature repository (see below) to be added.
4. Select the **Add URL** button.
5. The new feature repository is added to the list of **Feature Repositories** above the URL field.

There are several ways a new feature repository can be discovered based on the URL entered:

- The URL can contain the fully qualified path to the feature repository, e.g., mvn:<https://tools.codice.org/artifacts/content/groups/public/ddf-standard/2.2.0/xml/features>. This URL can include the username and password credentials if necessary. e.g., mvn:<https://user/password@tools.codice.org/artifacts/content/groups/public/ddf-standard/2.2.0/xml/features>. Note that the password will be in clear text.
- The URL can only be the mvn URL, e.g., mvn:ddf.features/ddf-standard/2.2/xml/features ddf-standard/2.2.0/xml/features. The feature repositories configured for DDF will be searched.

Repositories to be searched by DDF can be configured in one of several ways:

- Set the org.ops4j.pax.url.mvn.repositories property in the <DDF_INSTALL_DIR>/etc/org.ops4j.pax.url.mvn.cfg file (most common).
- Set the org.ops4j.pax.url.mvn.settings property in the <DDF_INSTALL_DIR>/etc/org.ops4j.pax.url.mvn.cfg file to the maven settings.xml file that specifies the repository(ies) to be searched. A settings.xml template file for this configuration is provided in <DDF_INSTALL_DIR>/etc/templates/settings.xml
- Create a maven settings.xml file in the <USER_HOME_DIR>/ .m2 directory of the user running DDF that specifies the repository(ies) to be searched (this method is typical for a developer).
- The simplest approach is to specify the fully qualified URL to the feature repository.

Known Issues

Blank Web Console

DDF uses Pax Web as part of its HTTP support. Modifying the Pax Web runtime configuration in the web console may cause the web console to freeze.

Solution

Use the configuration instructions according to the [hardening](#) instructions.

Additional Information

For more information on the Web Console refer to <http://felix.apache.org/site/apache-felix-web-console.html>

Configuring DDF Using the System Console

On This Page

- Overview
- Manage Features
 - Install Features
 - Uninstall Features

Overview

Follow these steps to configure DDF using the system console.

System Console instructions are provided in the [Console Commands](#) section.

Manage 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. For more information on the features that come with DDF, including a list of the ones included, consult the [DDF Included Features](#) page in the Software Version Description Document (SVDD).

Transitive Dependencies

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

Install Features

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

```
ddf@local>features:list
```

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 based on the version of DDF being run):

State	Version	Name
Repository	Description	
[installed]	[2.0.1]] ddf-core
[uninstalled]	[2.0.1]] ddf-sts
[installed]	[2.0.1]] ddf-security-common
[installed]	[2.0.1]] ddf-resource-impl
[uninstalled]	[2.0.1]] ddf-source-dummy

2. Install the desired feature.

```
ddf@local>features:install ddf-source-dummy
```

3. Check the feature list to verify the feature was installed.

```
ddf@local>features:list
```

State	Version	Name
Repository	Description	
[installed]	[2.0.1]] ddf-core ddf-2.1.0
[uninstalled]	[2.0.1]] ddf-sts ddf-2.1.0
[installed]	[2.0.1]] ddf-security-common ddf-2.1.0
[installed]	[2.0.1]] ddf-resource-impl ddf-2.1.0
[installed]	[2.0.1]] ddf-source-dummy ddf-2.1.0

- 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

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

```
ddf@local>features:list
```

State	Version	Name
Repository	Description	
[installed]	[2.0.1]] ddf-core ddf-2.1.0
[uninstalled]	[2.0.1]] ddf-sts ddf-2.1.0
[installed]	[2.0.1]] ddf-security-common ddf-2.1.0
[installed]	[2.0.1]] ddf-resource-impl ddf-2.1.0
[installed]	[2.0.1]] ddf-source-dummy ddf-2.1.0

- Uninstall the feature.

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

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

- Verify that the feature has uninstalled properly.

```
ddf@local>features:list
```

State	Version	Name
Repository	Description	
[installed]	[2.0.1]] ddf-core ddf-2.1.0
[uninstalled]	[2.0.1]] ddf-sts ddf-2.1.0
[installed]	[2.0.1]] ddf-security-common ddf-2.1.0
[installed]	[2.0.1]] ddf-resource-impl ddf-2.1.0
[uninstalled]	[2.0.1]] ddf-source-dummy ddf-2.1.0

Configuring DDF Using Configuration (.cfg) Files

On This Page

- Overview
- HTTP Port Configuration
 - Multiple Local DDF Nodes
- Enable SSL for Clients

Overview

The DDF can also be configured with configuration (.cfg) files.

HTTP Port Configuration

Do not use the Web Administration Console to change the HTTP port. While the Web Administration Console's Pax Web Runtime offers this configuration option, it has proven to be unreliable and may crash the system

Multiple Local DDF Nodes

Edit the port numbers in the files in the DDF install folder. The line numbers relate to 2.1.X releases.

File to Edit	Line Number	Original Value	Example of New Value
bin/karaf.bat	99	5005	i.e. 5006
etc/org.apache.karaf.management.cfg	27	1099	i.e. 1199
" "	32	44444	i.e. 44445
etc/org.ops4j.pax.web.cfg	9	8181	i.e. 8281
" "	22	8993	i.e. 8994

Be sure to note the port number that replaced 8181 then enter that number in the Web Console under the Configuration tab for the **Platform Global Configuration DDF Port** entry. Also edit the sitename so that there are no duplicates on your local machine.

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?](#).

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/serverKeystore.jks	The java keystore that contains the trusted public certificates for Certificate Authorities (CA's) 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/serverTruststore.jks	The keystore that contains the private key for the local server that can be used to signing and encryption. This must be set if establishing outgoing 2-way (mutual) SSL connections where the local server must also present it's certificate for the remote server to verify.
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

Enabling SSL

On This Page

- Enable SSL for Clients
- Enable SSL for Services
 - Pax Web Configuration Settings

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/serverKeystore.jks	The java keystore that contains the trusted public certificates for Certificate Authorities (CA's) 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/serverTruststore.jks	The keystore that contains the private key for the local server that can be used to signing and encryption. This must be set if establishing outgoing 2-way (mutual) SSL connections where the local server must also present it's certificate for the remote server to verify.

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

Enable SSL for Services

Do not use the Web Administration Console to SSL enable the DDF services. While the Web Administration Console's Pax Web Runtime 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.

Pax Web Configuration Settings

Property	Sample Value	Description
org.osgi.service.http.enabled	false	Set this to false to disable HTTP without SSL .
org.osgi.service.http.secure.enabled	true	Set this to true to SSL enable the DDF services.
org.osgi.service.http.port.secure	8993	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).
org.ops4j.pax.web.ssl.keystore.type	\${javax.net.ssl.keyStoreType}	These values should not be modified directly, as they should always be inherited from the system.properties setting for the associated properties. Modify the system.properties file to change these values.
org.ops4j.pax.web.ssl.keystore	\${javax.net.ssl.keyStore}	
org.ops4j.pax.web.ssl.keypassword	\${javax.net.ssl.keyStorePassword}	
org.ops4j.pax.web.ssl.password	\${javax.net.ssl.keyStorePassword}	

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

# They keystore and passwords pull from the Java System Properties which
are set
# in the system.properties file. It is recommended that the keystore
values are changed
# in that file if there are updates to the location or password of the
keystore.
# Fully-qualified path to your SSL keystore
org.ops4j.pax.web.ssl.keystore=${javax.net.ssl.keyStore}

# SSL Keystore Type
org.ops4j.pax.web.ssl.keystore.type=${javax.net.ssl.keyStoreType}

# Keystore Integrity Password
org.ops4j.pax.web.ssl.password=${javax.net.ssl.keyStorePassword}

# Keystore Password
org.ops4j.pax.web.ssl.keypassword=${javax.net.ssl.keyStorePassword}

```

All .cfg files follow a strict formatting structure: every entry is a key=value pair. There should be no whitespace before the key, around the equals sign (=), or after the value to prevent misinterpretation of the key or value.

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 occurs often when a DDF zip install file from a Unix operating system is transferred to a Windows operating system and installed.

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 need to pass the appropriate certificates.

Reference

Configuring a Java Keystore for Secure Communications

Additional Pax-Web SSL configuration info: <http://team.ops4j.org/wiki/display/paxweb/SSL+Configuration>

HTTP Port Configuration

On This Page

- [HTTP Port Configuration](#)
 - [Multiple Local DDF Nodes](#)

HTTP Port Configuration

Do not use the Web Administration Console to change the HTTP port. While the Web Administration Console's Pax Web Runtime offers this configuration option, it has proven to be unreliable and may crash the system

Multiple Local DDF Nodes

Edit the port numbers in the files in the DDF install folder. The line numbers relate to 2.1.X releases.

File to Edit	Line Number	Original Value	Example of New Value
bin/karaf.bat	99	5005	i.e. 5006
etc/org.apache.karaf.management.cfg	27	1099	i.e. 1199
" "	32	44444	i.e. 44445
etc/org.ops4j.pax.web.cfg	9	8181	i.e. 8281
" "	22	8993	i.e. 8994

Be sure to note the port number that replaced 8181 then enter that number in the Web Console under the Configuration tab for the **Platform Global Configuration DDF Port** entry. Also edit the sitename so that there are no duplicates on your local machine.

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?](#)

Use Port 80 as a Non-Root User

On This Page

- [Overview](#)
- [Use Jetty's setuid \(and setumask\) Feature](#)
- [Use ipchains](#)
- [Use iptables](#)
- [Use usermod](#)
- [Use xinetd](#)
- [Points to Note](#)

This documentation has been reproduced from the Jetty website for easy reference.

Overview

On Unix based systems, port 80 is protected and can usually be opened by only the superuser root. As it is not desirable to run the server as root (for security reasons), the possible solutions are:

- Start Jetty as the root user, and use Jetty's `setuid` mechanism to switch to a non-root user after startup,
OR
- Configure the server to run as a normal user on port 8080 (or some other non protected port). Then, configure the operating system to redirect port 80 to 8080 using `ipchains`, `iptables`, `ipfw` or a similar mechanism.

The latter has traditionally been the solution. However, Jetty 6.1 has added the new `setuid` feature.

If using Solaris 10, this feature may not be needed as Solaris provides a User Rights Management framework that can permit users and processes superuser-like abilities. Refer to the [Solaris documentation](#) for more information.

Use Jetty's `setuid` (and `setumask`) Feature

1. Create a jetty config file like the following:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN"
"http://jetty.mortbay.org/configure.dtd">
<Configure id="Server" class="org.mortbay.setuid.SetUIDServer">
    <Set name="umask">UMASK</Set>
    <Set name="uid">USERID</Set>
</Configure>
```

Where:

- `UMASK` is replaced with the umask setting the process needs, or optionally remove this line if the process umask setting does not need to be changed at runtime.
- `USERID` is replaced with the id of the user process needs to execute as once the ports have been opened.

Hint

An existing Jetty configuration file like the above is located in the `$jetty.home/extras/setuid/etc/jetty-setuid.xml`.

2. Build the `setuid` feature for the specific operating system, as it requires native libraries. Go to the `$jetty.home/extras/setuid` directory and follow the instructions in the `README.txt` file, summarized here as:

```
> mvn install

> gcc -I$JDK_HOME/include/ -I$JDK_HOME/include/linux/ \
      -shared src/main/native/org_mortbay_setuid_SetUID.c \
      -o ../../lib/ext/libsetuid.so

> cp target/jetty-setuid-6.1-SNAPSHOT.jar ../../lib/ext/
> cp etc/jetty-setuid.xml ../../etc
```

Where:

- `$JDK_HOME` is same as `$JAVA_HOME`.
- `linux` should be replaced by the name of DDF's operating system.

On Solaris

Omit the `-shared` argument.

3. Run jetty as the root user by switching to the userid (and setting the umask, if configured):

```
sudo java -Djava.library.path=lib/ext -jar start.jar etc/jetty-setuid.xml  
etc/jetty.xml
```

The `etc/jetty-setuid.xml` file must be first in the list of config files.

Use ipchains

On some Linux systems, the `ipchains` REDIRECT mechanism can be used to redirect from one port to another inside the kernel:

```
/sbin/ipchains -I input --proto TCP --dport 80 -j REDIRECT 8080
```

This basically means, "Insert into the kernel's packet filtering the following as the first rule to check on incoming packets: If the protocol is TCP and the destination port is 80, redirect the packet to port 8080." The kernel must be compiled with support for `ipchains`. (virtually all stock kernels are.) The `ipchains` command-line utility must be installed. (On RedHat the package is named "ipchains".) This command can run at any time, preferably just once since it inserts another copy of the rule every time it is executed.

Once this rule is set up, a Linux 2.2 kernel will redirect all data addressed to port 80 to a server, such as Jetty running on port 8080. This includes all RedHat 6.x distros. Linux 2.4 kernels, e.g., RedHat 7.1+, have a similar `iptables` facility.

Use iptables

A command similar to the following must be added to the startup scripts or the firewall rules:

```
/sbin/iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-port  
8080
```

The underlying model of `iptables` is different to that of `ipchains`, so the forwarding normally only happens to packets originating off-box. Incoming packets must be routed to port 8080 if `iptables` is used as a local firewall.

Be careful to place rules like this early in the "input" chain. Such rules must precede any rule that would accept the packet, otherwise the redirection won't occur. As many rules as needed can be inserted if the server needs to listen on multiple ports, as for HTTPS.

Use usermod

On Solaris 10 (possibly including earlier versions), the OS allows privileges to be granted to ports binding to "normal" users:

```
usermod -K defaultpriv=basic,net_privaddr myself
```

Now the `myself` user will be able to bind to port 80.

Use xinetd

With modern Linux flavors, `inetd` has an improved command `xinetd`. There are existing Unix man pages that describes its functionality in detail.

`xinetd` can be used to redirect network traffic, requiring configuration with only a text editor.

`xinetd` is driven by text files. There are two ways to give `xinetd` instructions:

1. Add a new service to `etc/xinetd.conf`.
2. Add a new file to the directory `etc/xinetd.d`.

For both approaches, the format is the same.

The following entry will redirect all inward TCP traffic on port 80 to port 8888 on the local machine. Redirection to other machines is also possible for gimp proxying.

```
service my_redirector
{
    type = UNLISTED
    disable = no
    socket_type = stream
    protocol = tcp
    user = root
    wait = no
    port = 80
    redirect = 127.0.0.1 8888
    log_type = FILE /tmp/somefile.log
}
```

Points to Note

- A space must be on either side of the =, or the line is ignored.
- `type = UNLISTED` means that the name of the service does not have to be in `/etc/services`, but the port and protocol must be specified. If an existing service name, e.g., http, is used:

```
service http
{
    disable = no
    socket_type = stream
    user = root
    wait = no
    redirect = 127.0.0.1 8888
    log_type = FILE /tmp/somefile.log
}
```

- Examine the `/etc/services` for more info on this configuration.
- Logging may present certain security problems, hence it can be omitted if necessary
- RHEL5 does not contain `xinetd` by default. Doing a `yum install xinetd` will fix this.

Enable HTTP Access Logging

On This Page

- Enable Access Logs for our Current DDF
- Additional Information

Enable Access Logs for our Current DDF

1. Update the file (`org.ops4j.pax.web.cfg`) located in `etc/` by adding the following text to the bottom:

```
org.ops4j.pax.web.config.file=etc/jetty.xml
```

This tells pax-web to look at the external jetty.xml file when performing the initial jetty configuration.

2. 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/" />/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>

```

3. Change the location of the logs to the desired location. In the settings above, the default is data/log (same place where the log is located). The log is using NCSA 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

```

Additional Information

<http://team.ops4j.org/wiki/display/paxweb/Advanced+Jetty+Configuration>

<http://wiki.eclipse.org/Jetty/Tutorial/RequestLog>

Configuring a Java Keystore for Secure Communications

On This Page

- Create a Client Keystore
- Create a Truststore
- Add a Certificate to an Existing Keystore

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>.

Create a Client Keystore

The following steps define the procedure for using a PKCS12 certificate. This is the most popular format that is used when exporting from a web browser.

1. Obtain a personal ECA cert (client certificate).
 - a. Open **Internet Explorer Tools Options**.
 - b. Select the Content tab.
 - c. Select **Certificates**.
 - d. Select the Personal tab.
 - e. Select the certificate to be exported. Choose the certificate without a "Friendly Name" and is **not** the "Encryption Cert".
 - f. Select the **Export** button.
 - g. Follow the steps in the Certificate Export Wizard.
 - h. When a prompt requests to export the private key, select the **Yes** button.
2. Download a jetty 6.1.5 distribution from <http://dist.codehaus.org/jetty/jetty-6.1.5/jetty-6.1.5.zip>.
3. Unpack the jetty distribution and place the client certificate (the one just exported) in the lib directory.
4. Navigate to the lib directory of the jetty distribution in a command console.
5. Add a cert to a new Java keystore, replacing cert with the name of the PKCS12 keystore to be converted.
6. Replace clientKeystore with the desired name of the Java keystore:

```
java -cp jetty-6.1.5.jar org.mortbay.jetty.security.PKCS12Import  
cert.p12 clientKeystore.jks
```

7. Enter the two passwords when prompted.
 - a. Input keystore passphrase is the passphrase that is used to protect cert.p12.
 - b. Output keystore passphrase is the passphrase that is set for the new Java keystore clientKeystore.jks.
8. It is recommended to set the private key password to the same as the keystore password due to limitations in Java.
9. Run the following command to determine the alias name of the added current entry. It is listed after Alias Name:

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

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

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

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

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

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

Create a Truststore

1. Import the certificate into a Java keystore as a trusted ca certificate.

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

2. Enter in a keystore password when prompted.

Add a Certificate to an Existing Keystore

1. Import the certificate into a Java keystore as a certificate.

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

2. Enter in the keystore password, if prompted.

Configuring Solr Catalog Provider

On This Page

- Configure the Solr Catalog Provider Data Directory
- Changing the Data Directory after DDF has ingested data

Configure the 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 Catalog Provider

On This Page

- [Overview](#)
- [Reconfigure](#)

Overview

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

This scenario assumes DDF is already running.

Use of the Dummy Catalog Provider

This scenario uses the Dummy Catalog Provider as the catalog provider DDF is being reconfigured to use. This is because the Dummy Catalog Provider is the only other catalog provider shipped with DDF out of the box. The Dummy Catalog Provider should **never** be used in a production environment. It is only used for testing purposes.

Reconfigure

1. Uninstall a Catalog Provider (if installed) by completing the procedure in the [Uninstalling Features](#) section.
2. Install the new [Catalog Provider](#), which will be the , by installing its feature `ddf-provider-dummy` by completing the instructions in the [Installing Features](#) section.
3. Verify DDF is running with the Dummy Provider as its [Catalog Provider](#).
 - a. Select the Services tab in the Web Console.
 - b. Locate the column labeled **Bundle** on the right. If DDF is running with the Dummy Provider, there is an entry labeled `ddf.providers.provider-dummy`, as shown below.

New catalog provider Dummy Provider installed

Id	Type(s)	Bundle
325	[<code>ddf.catalog.source.CatalogProvider</code>]	
<code>ddf.providers.provider-dummy</code> (175)		<code>osgi.service.blueprint.compname DummyProvider</code>

Configuring Notifications

On This Page

- [Overview](#)
- [Usage](#)
- [Receive Notifications](#)
- [Publish Notifications](#)

Overview

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.

Usage

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 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 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/dib.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1wlql079j6tscii19jszwp9s2i55 notifications contain the following information:

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

Receive Notifications

- If interested in retrieve resource notifications, a client must subscribe to the CometD channel `/ddf/notification/catalog/downloads`.
- If interested in all notification types, a client must subscribe to the CometD channel `/ddf/notification/**`
- A client will only receive notifications for resources they have requested.
- DDF 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.

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.

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);
```

Managing Web Service Security

On This Page

- Overview
- Security Framework
 - Subject
 - Security Manager
 - Authentication Tokens
 - Realms
 - Auditing
 - Authentication (AuthN)
 - Authorization (AuthZ)
- Security Token Service

Overview

The Web Service Security (WSS) functionality that comes with DDF is integrated throughout the system. This page acts as a central point to show how all of the pieces work together and point out where they live inside of 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.

Security Framework

The DDF Security Framework utilizes Apache Shiro (<http://shiro.apache.org/>) as the underlying security framework. The classes mentioned in this section will have their full package name listed, so it is 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.

Implementations of Subject:

Classname	Description
ddf.security.impl.SubjectImpl	Extends org.apache.shiro.subject.support.DelegatingSubject

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.

Implementations of Security Managers:

Classname	Description
ddf.security.service.SecurityManagerImpl	This implementation of the Security Manager handles taking in both org.apache.shiro.authc.AuthenticationToken and org.apache.cxf.ws.security.tokenstore.SecurityToken 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
ddf.security.service.impl.cas.CasAuthenticationToken	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.

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
ddf.security.realm.sts.StsRealm	This realm delegates authentication to the STS. It creates a RequestSecurityToken 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/Redaction. 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).

Implementations of Authorizing Realms that come with DDF:

Classname	Description

ddf.security.service.AbstractAuthorizingRealm	This is an Abstract Authorizing Realm that takes care of caching and parsing the Subject's AuthorizingInfo and should be extended to allow the implementing realm focus on making the decision.
ddf.security.pep.realm.XACMLRealm	This realm delegates the authorization decision to a XACML-based Policy Decision Point (PDP) backend. It creates a XACML 3.0 request and looks on the OSGi framework for any service implementing ddf.security.pdp.api.PolicyDecisionPoint.
ddf.security.pdp.realm.SimpleAuthZRealm	This realm performs the authorization decision without delegating to an external service. It uses the incoming permissions to create a decision.

Auditing

Refer to the [Auditing](#) section of this documentation.

Authentication (AuthN)

Central Authentication Server (CAS)

Refer to the [CAS SSO Configuration](#) section of this documentation.

Authorization (AuthZ)

Service Authorization

Refer to the [XACML Policy Decision Point \(PDP\)](#) section of this documentation.

Resource Authorization

Refer to the [Redaction and Filtering](#) section of this documentation.

Security Token Service

Refer to the [Security Token Service](#) section of this documentation.

Auditing

On This Page

- Overview
- CAS (SSO) Authentication
 - Username and Password
 - PKI Certificate
- STS Authentication
 - Username and Password
 - PKI Certificate
 - Binary Security Token (CAS)

Overview

This page details how to find and read the security audit logs.

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

CAS (SSO) Authentication

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

Sample – Successful login

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

```

2013-04-24 15:13:14,388 INFO
[org.jasig.cas.adaptors.x509.authentication.handler.support.X509Credential
sAuthenticationHandler] - <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.X509Credential
sAuthenticationHandler 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.X509Credential
sAuthenticationHandler@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.X509Credential
sAuthenticationHandler] - <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.X509Credential
sAuthenticationHandler] - <.* 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.X509Credential
sAuthenticationHandler] - <CN=Hogwarts Certifying Authority.+ matches
EMAILADDRESS=goss-support@owfgooss.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.X509Credential
sAuthenticationHandler] - <Found valid client certificate>
2013-04-25 14:15:47,478 INFO
[org.jasig.cas.adaptors.x509.authentication.handler.support.X509Credential
sAuthenticationHandler] - <Failed to authenticate
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCreden
tials@1795f1cc>
2013-04-25 14:15:47,478 INFO
[org.jasig.cas.authentication.AuthenticationManagerImpl] -
<org.jasig.cas.adaptors.x509.authentication.handler.support.X509Credential
sAuthenticationHandler failed to authenticate
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCreden
tials@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.X509CertificateCreden
tials@1795f1cc
WHAT: supplied credentials:
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCreden
tials@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=L
ockheed 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=L
ockheed 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=L  
ockheed 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.  
APPLIESTO: <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-wssecurity-
secext-1.0.xsd"
xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
ity-utility-1.0.xsd">U1QtMTctQmw0aGRRS05jaTV3cE82Zm11VE0tY2FzfGh0dHBzOi8vd
G9rZW5pc3N1ZXI6ODk5My9zZXJ2aNlcy9TZN1cm10eVRva2VuU2VydmljZQ==</BinarySec
urityToken>
Request IP: 0:0:0:0:0:0:0:1%0, 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-wssecur
ity-utility-1.0.xsd">U1QtMjctOU43RulkNHkzVFoxQmZCb0RIdkItY2Fz</BinarySecur
ityToken>
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.

```

CAS SSO Configuration

On This Page
<ul style="list-style-type: none"> • Overview • General Server Setup and Configuration <ul style="list-style-type: none"> • Install using DDF CAS WAR • Configure an Existing CAS Installation • Configure CAS for DDF <ul style="list-style-type: none"> • CAS Client • CAS Token Validator • Additional Configuration • Example Workflow • External Links

Overview

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

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.

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 [<http://tomcat.apache.org/download-70.cgi>]. 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

```
copy 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
```

- d. 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

```
cp <DDF_INSTALLATION_DIR>/etc/keystores/casKeystore.jks  
<TOMCAT_INSTALLATION_DIR>/certs/keystore.jks
```

Windows

```
copy <DDF_INSTALLATION_DIR>\etc\keystores\casKeystore.jks  
<TOMCAT_INSTALLATION_DIR>\certs\keystore.jks
```

4. Start Tomcat.

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

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.

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.

External Links

Official CAS documentation: <https://wiki.jasig.org/display/CASUM/Home>

Configure CAS for LDAP

On This Page

- [Install and Configure LDAP](#)
- [Add cas-server-support-ldap-3.3.1_1.jar to CAS](#)
- [Add spring-ldap-1.2.1_1.jar to CAS](#)
- [Modify developerConfigContext.xml](#)
- [Configure Ozone](#)

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 `${ozone-widget-framework}/apache-tomecat-${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 `${ozone-widget-framework}/apache-tomecat-${version}/webapps/cas/WEB-INF/lib/spring-ldap-1.2.1_1.jar`.

Modify developerConfigContext.xml

1. In `${ozone-widget-framework}/apache-tomecat-${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-tomecat-\${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-tomecat-\${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
 - a. In \${ozone-widget-framework}/apache-tomecat-\${version}/lib/OWFsecurityContext.xml, change the sec:x509 stanza to the following:

OWFsecurityContext.xml

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

- b. In \${ozone-widget-framework}/apache-tomecat-\${version}/lib/OWFsecurityContext.xml, remove the following import:

OWFsecurityContext.xml

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

- c. In \${ozone-widget-framework}/apache-tomecat-\${version}/lib/OWFsecurityContext.xml, add the following import:

OWFsecurityContext.xml

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

2. Modify LdapBeans.xml

- a. In \${ozone-widget-framework}/apache-tomecat-\${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.DefaultSpringSecurityCo  
nTEXTSource">  
    <!-- 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-tomecat-\${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.DefaultLdap  
AuthoritiesPopulator">  
    <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-tomecat-\${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.FilterBasedLdapU
serSearch">
    <!-- 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-tomecat-\${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.OWFUserDetailsCo
nverterMapper">
    <constructor-arg ref="contextSource" />
    <!-- search base for finding OFW 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

- a. In \${ozone-widget-framework}/apache-tomecat-\${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.CasAuthen
ticationProvider">
    <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>
```

On This Page

- Overview
- Add the cas-server-support-x509-3.3.1.jar to CAS
- Configure Web Flow
- Configure the Authentication Handler
- Configure the Credentials to the Principal Resolver
- Default Certificates
- External Links

Overview

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-tomecat-${version}/webapps/cas/WEB-INF/lib/cas-server-support-x509-3.3.1.jar`.

Configure Web Flow

1. In `apache-tomecat-${version}/webapps/cas/WEB-INF/login-workflow.xml`, make the following modifications:
 - a. 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:
- Define the x509Check bean.

x509Check

```
<bean
    id="x509Check"

    p:centralAuthenticationService-ref="centralAuthenticationService"

    class="org.jasig.cas.adaptors.x509.web.flow.X509CertificateCrede
ntialsNonInteractiveAction" >
    <property name="centralAuthenticationService"
        ref="centralAuthenticationService"/>
</bean>
```

Configure the Authentication Handler

In apache-tomcat-\${version}/webapps/cas/WEB-INF/deployerConfigContext.xml, make the following modifications:

- 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.X50
9CredentialsAuthenticationHandler">
                <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.X509CertificateCredentialsToIdentifierPrincipalResolver">
                    <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
X509CertificateCredentialsToDistinguishedNamePrincipalResolver	Retrieve the complete distinguished name and use that as the identifier.
X509CertificateCredentialsToIdentifierPrincipalResolver	Transform some subset of the identifier into the ID for the principal.
X509CertificateCredentialsToSerialNumberPrincipalResolver	Use the unique serial number of the certificate.
X509CertificateCredentialsToSerialNumberAndIssuerDNPrincipalResolver	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: <https://wiki.jasig.org/display/CASUM/X.509+Certificates>.

Certificate Management

On This Page
<ul style="list-style-type: none">• Overview• Default Certificates• File Management• Configuration Management

Overview

DDF uses certificates in two ways:

1. To transmit and receive encrypted messages.
2. To perform 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".

Alias	Keystore	Truststore	Configuration Location	Usage
localhost	serverKeystore.jks	serverTruststore.jks	<p>File: <i>etc/org.ops4j.pax.web.cfg</i></p> <p>File: <i>etc/ws-security/server/encryption.properties</i></p> <p>File: <i>etc/ws-security/server/signature.properties</i></p> <p>File: <i>etc/ws-security/issuer/encryption.properties</i></p> <p>File: <i>etc/ws-security/issuer/signature.properties</i></p> <p>File: <i>etc/ddf.platform.config.cfg</i></p>	Used to secure (SSL) all of the endpoints for DDF, to perform outgoing SSL requests, sign STS SAML assertions . This also includes the admin console and any other web service that is hosted by DDF.

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.

[Cert File Management](#)

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 Configuration Management](#)

[Cert File Management](#)

On This Page

- Overview
- General Certificates
 - Create a CA Key and Certificate
 - Use the CA to Sign Certificates
- Java Keystore (JKS)
 - Create a New Keystore/Truststore with an Existing Certificate and Private Key
 - Import into a Java Keystore (JKS)
- Certificate Revocation List (CRL)
 - Create a Certificate Revocation List (CRL)
 - Revoke a Certificate and Create a New CRL that Contains the Revoked Certificate
 - View a CRL

Overview

The following tools are used:

- openssl
 - Windows users can use: openssl for windows (https://code.google.com/p/openssl-for-windows/downloads/detail?name=openssl-0.9.8k_X64.zip&can=2&q=)
- The standard Java **keytool** certificate management utility (<http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html>)
- Portecle (<http://portecle.sourceforge.net/>) 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 define the procedure for 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
```

Use the CA to Sign Certificates

The following steps define the procedure for 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
clientKeystore.jks -srcstoretype pkcs12 -alias 1
keytool -changealias -alias 1 -destalias client -keystore
clientKeystore.jks
keytool -importcert -file ca.crt -keystore clientKeystore.jks -alias "ca"
keytool -importcert -file ca-root.crt -keystore clientKeystore.jks -alias
"ca-root"
```

2. Create the truststore using just 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 it is the format that some applications use.

```
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 define the procedure for 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)

Create 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
```

View 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
```

Certificate Configuration Management

On This Page

- Overview
- Certificate Revocation Configuration
 - Enable Revocation
 - Add Revocation to a New Endpoint
 - Verify Revocation Is Taking Place

Overview

This page details how to revoke a certificate, if it becomes necessary to do so.

Certificate Revocation Configuration

Enable Revocation

1. Place the CRL in <ddf.home>/etc keystores.
2. Uncomment the following line in <ddf.home>/etc/ws-security/server/encryption.properties and replace the file name with the CRL file used in step 1.

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

Add Revocation to a New Endpoint

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. The CXF WS-Security page (<http://cxf.apache.org/docs/ws-securitypolicy.html>) contains additional information and samples.

1. Add the following property to the jaxws 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>
```

Verify 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
| ecurity.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:83
8)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.validate.SignatureTrustValidator.verifyTrustInCert(
SignatureTrustValidator.java:213)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.validate.SignatureTrustValidator.validate(Signature
TrustValidator.java:72)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.validate.SamlAssertionValidator.verifySignedAsserti
on(SamlAssertionValidator.java:121)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.validate.SamlAssertionValidator.validate(SamlAssert
ionValidator.java:100)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.processor.SAMLTokenProcessor.handleSAMLToken(SAMLTo
kenProcessor.java:188)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.processor.SAMLTokenProcessor.handleToken(SAMLTokenP
rocessor.java:78)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.ws.security.WSSecurityEngine.processSecurityHeader(WSSecurityEn
gine.java:396)[161:org.apache.ws.security.wss4j:1.6.9]
at
org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor.handleMessage(WSS4JInI
```

```
nterceptor.java:274)[164:org.apache.cxf.cxf-rt-ws-security:2.7.3]
    at
org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor.handleMessage(WSS4JInI
nterceptor.java:93)[164:org.apache.cxf.cxf-rt-ws-security:2.7.3]
    at
org.apache.cxf.phase.PhaseInterceptorChain.doIntercept(PhaseInterceptorCha
in.java:271)[123:org.apache.cxf.cxf-api:2.7.3]
    at
org.apache.cxf.transport.ChainInitiationObserver.onMessage(ChainInitiation
Observer.java:121)[123:org.apache.cxf.cxf-api:2.7.3]
    at
org.apache.cxf.transport.http.AbstractHTTPDestination.invoke(AbstractHTTPD
estination.java:239)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.ServletController.invokeDestination(Servl
etController.java:218)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.ServletController.invoke(ServletControll
er.java:198)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.ServletController.invoke(ServletControll
er.java:137)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.CXFNonSpringServlet.invoke(CXFNonSpringSe
rvlet.java:158)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.AbstractHTTPServlet.handleRequest(Abstract
HTTPServlet.java:243)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.AbstractHTTPServlet.doPost(AbstractHTTPSe
rvlet.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(AbstractHTTPS
ervlet.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.jetty.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.doHandl
e(HttpServiceServletHandler.java:70)[73:org.ops4j.pax.web.pax-web-jetty:1.
0.11]
    at
org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:1
19)[61:org.eclipse.jetty.server:7.5.4.v20111024]
    at
org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:520
)[62:org.eclipse.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.java: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.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:875)[61:org.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:17)[61:org.eclipse.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.eclipse.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.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.jetty.server.HttpConnection$RequestHandler.content(HttpConnection.java:936)[61:org.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.jetty.http.HttpParser.parseNext(HttpParser.java:893)[57:org.eclipse.jetty.http:7.5.4.v20111024]
        at
org.eclipse.jetty.http.HttpParser.parseAvailable(HttpParser.java:218)[57:org.eclipse.jetty.http:7.5.4.v20111024]
        at
org.eclipse.jetty.server.BlockingHttpConnection.handle(BlockingHttpConnection.java:50)[61:org.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.jetty.server.bio.SocketConnector$ConnectorEndPoint.run(SocketConnector.java:245)[61:org.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.jetty.server.ssl.SslSocketConnector$SslConnectorEndPoint.run(SslSocketConnector.java:663)[61:org.eclipse.jetty.server:7.5.4.v20111024]
        at
org.eclipse.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:
```

```
a:598)[55:org.eclipse.jetty.util:7.5.4.v20111024]
    at
org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java
:533)[55:org.eclipse.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(PKIXMa
sterCertPathValidator.java:139)[:1.6.0_33]
    at
sun.security.provider.certpath.PKIXCertPathValidator.doValidate(PKIXCertPa
thValidator.java:330)[:1.6.0_33]
    at
sun.security.provider.certpath.PKIXCertPathValidator.engineValidate(PKIXCe
rtPathValidator.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:81  
4)[161:org.apache.ws.security.wss4j:1.6.9]  
... 45 more
```

Encryption Service

On This Page

- Overview
- Encryption Command

Overview

The encryption service and encryption command, which are based on Jasypt (<http://www.jasypt.org/>), provide an easy way for developers to add encryption capabilities to DDF.

Encryption Command

An encrypt security command is provided with DDF that allows plain text to be encrypted. This is useful when displaying password fields in a GUI.

Below is an example of the security:encrypt command used to encrypt the plain text "myPasswordToEncrypt". The output, bR9mJpDV08bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=, is the encrypted value.

```
ddf@local>security:encrypt myPasswordToEncrypt  
  
bR9mJpDV08bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

Redaction and Filtering

On This Page

- Overview
- How it Works
- Redaction Policies
- Redact a New Type of Metocard

Overview

Redaction and filtering are performed in a Post Query plugin that occurs after a query has been performed.

How it Works

Each metocard result will contain security attributes that are pulled from the metadata record after being processed by a PostQueryPlugin 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/redaction 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/redact* the metocard eventually relies on the installed PDP (features:install security-pdp-java OR features:install security-pdp-xacml). The PDP that is being used returns a decision, and the metocard will either be filtered/redacted or allowed to pass through.

*The default setting is to redact records.

The security attributes populated on the metocard are completely dependent on the type of the metocard. Each type of metocard must have its own PostQueryPlugin that reads the metadata being returned and populates the metocard's security attribute. If the subject or resource (metocard) permissions are missing during redaction, that resource is redacted.

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.

Redaction Policies

The procedure for setting up a policy differs depending on which PDP implementation installed. The security-pdp-java implementation is the simplest PDP to use, so it will be covered here.

1. Open the <http://localhost:8181/system/console/configuration>.*
2. Click on the **Authz Security Settings** configuration.
3. Add any roles that are allowed to access protected services.
4. Add any SOAP actions that are not to be protected by the PDP.
5. Add any attribute mappings necessary to map between subject claims and metocard values.
 - a. For example, the above example would require two Match All mappings of claim1=entry1 and claim2=entry2
 - b. Match One mappings would contain claim3=entry3 and claim4=entry4.

*See the [Security PDP AuthZ Realm](#) section of this documentation for a description of the configuration page.

With the security-pdp-java feature configured in this way, the above Metocard would be displayed to the user.

The XACML PDP is explained in more detail in the [XACML Policy Decision Point \(PDP\)](#) section of this documentation. It the administrator's responsibility to write a XACML policy capable of returning the correct response message. The Java-based PDP should perform adequately in most situations. It is possible to install the security-pdp-java and security-pdp-xacml features at the same time. The system could be configured in this way in order to allow the Java PDP to handle most cases and only have XACML policies to handle more complex situations than what the Java PDP is designed for. Keep in mind that this would be a very complex configuration with both PDPs installed, and this should only be performed if you understand the complex details.

Redact a New Type of Metocard

To enable redaction/filtering on a new type of record, implement a PostQueryPlugin that is able to read the string metadata contained within the metocard record. The plugin must set the security attribute to a map of list of values extracted from the metocard. Note that in DDF, there is no default plugin that populates the security attribute on the metocard. A plugin must be created to populate these fields in order for redaction/filtering to work correctly.

Example redacted record:

```
<?xml version="1.0" encoding="UTF-8"?>
<metocard xmlns="urn:catalog:metocard"
  xmlns:ns2="http://www.opengis.net/gml"
  xmlns:ns3="http://www.w3.org/1999/xlink"
  xmlns:ns4="http://www.w3.org/2001/SMIL20/"
  xmlns:ns5="http://www.w3.org/2001/SMIL20/Language"
  ns2:id="99f494b22f4341e9a3ba3ef6a5fe8734">
  <type>ddf.metocard</type>
  <source>ddf.distribution</source>
  <stringxml name="metadata">
    <value>
      <meta:Resource xmlns:meta="...">
        <meta:identifier meta:qualifier="http://metadata/noaccess"
          meta:value="REDACTED" />
        <meta:title>REDACTED</meta:title>
        <meta:creator>
          <meta:Organization>
            <meta:name>REDACTED</meta:name>
          </meta:Organization>
        </meta:creator>
        <meta:subjectCoverage>
          <meta:Subject>
            <meta:category meta:label="REDACTED" />
          </meta:Subject>
        </meta:subjectCoverage>
        <meta:security SEC:controls="A B" SEC:group="X" SEC:origin="USA" />
      </meta:Resource>
    </value>
  </stringxml>
  <string name="resource-uri">
    <value>catalog://metadata/noaccess</value>
  </string>
  <string name="title">
    <value>REDACTED</value>
  </string>
  <string name="resource-size">
    <value>REDACTED</value>
  </string>
</metocard>
```

On This Page

- Overview
- Using the Security Token Service (STS)
 - Standalone Installation
- STS Claims Handlers
 - Add a Custom Claims Handler
- STS WS-Trust WSDL Document
- Example Request and Responses for a SAML Assertion
 - RequestSecurityToken

Overview

The Security Token Service (STS) is a service running in DDF that allows clients to request 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-Security policies. 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.

Using the Security Token Service (STS)

Once installed, the STS can be used to request 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.

Standalone Installation

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

Unnecessary Features
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.

Add a Custom Claims Handler

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.

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, a developer may have to add the root CA of the external site to the DDF trustStore and add a valid certificate into the DDF keyStore. Doing so will allow the SSL to encrypt messages that will be accepted by the external service. For more information on certificates, refer to the [Configuring a Java Keystore for Secure Communications](#) page.

STS WS-Trust WSDL Document

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>
        <xs:schema elementFormDefault="qualified"
targetNamespace="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
            <xs:element name="RequestSecurityToken"
type="wst:AbstractRequestSecurityTokenType"/>
            <xs:element name="RequestSecurityTokenResponse"
type="wst:AbstractRequestSecurityTokenType"/>
            <xs:complexType name="AbstractRequestSecurityTokenType">
                <xs:sequence>
                    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="Context" type="xs:anyURI" use="optional"/>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:complexType>
            <xs:element name="RequestSecurityTokenCollection"

```

```

type="wst:RequestSecurityTokenCollectionType"/>
  <xs:complexType name="RequestSecurityTokenCollectionType">
    <xs:sequence>
      <xs:element name="RequestSecurityToken"
type="wst:AbstractRequestSecurityTokenType" minOccurs="2"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="RequestSecurityTokenResponseCollection"
type="wst:RequestSecurityTokenResponseCollectionType"/>
  <xs:complexType name="RequestSecurityTokenResponseCollectionType">
    <xs:sequence>
      <xs:element ref="wst:RequestSecurityTokenResponse" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
</xs:schema>
</wsdl:types>
<!-- WS-Trust defines the following GEDs -->
<wsdl:message name="RequestSecurityTokenMsg">
  <wsdl:part name="request" element="wst:RequestSecurityToken" />
</wsdl:message>
<wsdl:message name="RequestSecurityTokenResponseMsg">
  <wsdl:part name="response" element="wst:RequestSecurityTokenResponse" />
</wsdl:message>
<wsdl:message name="RequestSecurityTokenCollectionMsg">
  <wsdl:part name="requestCollection"
element="wst:RequestSecurityTokenCollection" />
</wsdl:message>
<wsdl:message name="RequestSecurityTokenResponseCollectionMsg">
  <wsdl:part name="responseCollection"
element="wst:RequestSecurityTokenResponseCollection" />
</wsdl:message>
<!-- This portType an example of a Requestor (or other) endpoint that
     Accepts SOAP-based challenges from a Security Token Service -->
<wsdl:portType name="WSSecurityRequestor">
  <wsdl:operation name="Challenge">
    <wsdl:input message="tns:RequestSecurityTokenResponseMsg" />
    <wsdl:output message="tns:RequestSecurityTokenResponseMsg" />
  </wsdl:operation>
</wsdl:portType>
<!-- This portType is an example of an STS supporting full protocol -->
<wsdl:portType name="STS">
  <wsdl:operation name="Cancel">
    <wsdl:input
wsam:Action="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Cancel"
message="tns:RequestSecurityTokenMsg" />
    <wsdl:output
wsam:Action="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/CancelF
inal" message="tns:RequestSecurityTokenResponseMsg" />
  </wsdl:operation>
  <wsdl:operation name="Issue">

```

```

<wsdl:input
wsam:Action="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue"
message="tns:RequestSecurityTokenMsg" />
<wsdl:output
wsam:Action="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueF
inal" 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/RenewFi
nal" 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/Validat
eFinal" 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/KETFina
l" message="tns:RequestSecurityTokenResponseMsg" />
</wsdl:operation>
<wsdl:operation name="RequestCollection">
<wsdl:input message="tns:RequestSecurityTokenCollectionMsg" />
<wsdl:output message="tns:RequestSecurityTokenResponseCollectionMsg" />
</wsdl:operation>
</wsdl:portType>
<!-- This portType is an example of an endpoint that accepts
    Unsolicited RequestSecurityTokenResponse messages -->
<wsdl:portType name="SecurityTokenResponseService">
<wsdl:operation name="RequestSecurityTokenResponse">
<wsdl:input message="tns:RequestSecurityTokenResponseMsg" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="STS_Binding" type="wstrust:STS">
<wsp:PolicyReference URI="#STS_policy" />
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="Issue">
<soap:operation
soapAction="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue" />
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>

```

```
<wsdl:output>
  <soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Validate">
  <soap:operation
    soapAction="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Validate"
  />
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="Cancel">
  <soap:operation
    soapAction="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Cancel"
  />
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="Renew">
  <soap:operation
    soapAction="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Renew"
  />
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="KeyExchangeToken">
  <soap:operation
    soapAction="http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/KeyExchan
geToken"
  />
  <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/RequestCo
llection"
  />
  <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>
    
```

```
</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>
```

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.

RequestSecurityToken

Refer to the [UsernameToken Bearer SAML Security Token Request/Response](#) section of this documentation.

Refer to the [X.509 PublicKey SAML Security Token Request/Response](#) section of this documentation.

Refer to the [BinarySecurityToken \(CAS\) SAML Security Token Request/Response](#) section of this documentation.

BinarySecurityToken (CAS) SAML Security Token Request/Response

On This Page

- [BinarySecurityToken \(CAS\) Sample Request/Response](#)
 - 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-a
      4a7-8a5ce243a7cb</MessageID>
    <To
      xmlns="http://www.w3.org/2005/08/addressing">https://server:8993/services/
      SecurityTokenService</To>
    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
      <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
    </ReplyTo>
    <wsse:Security
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
      rity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
      ity-utility-1.0.xsd" soap:mustUnderstand="1">
```

```

<wsu:Timestamp wsu:Id="TS-1">
  <wsu:Created>2013-04-29T18:35:10.688Z</wsu:Created>
  <wsu:Expires>2013-04-29T18:40:10.688Z</wsu:Expires>
</wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body>
  <wst:RequestSecurityToken
    xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">

    <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
    <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
      <wsa:EndpointReference
        xmlns:wsa="http://www.w3.org/2005/08/addressing">

        <wsa:Address>https://server:8993/services/SecurityTokenService</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:Claims
        xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
        xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
        Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
        <ic:ClaimType
          xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
          Optional="true"
          Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
        <ic:ClaimType
          xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
          Optional="true"
          Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
        <ic:ClaimType
          xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
          Optional="true"
          Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
        <ic:ClaimType
          xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
          Optional="true"
          Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
        <ic:ClaimType
          xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
          Optional="true"
          Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
      </wst:Claims>
      <wst:OnBehalfOf>
        <BinarySecurityToken ValueType="#CAS"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ns1:Id="CAS"
          xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
          xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">U1QtMTQtYUtmcDYxcFRtS0FxZG1pVDMzOWMtY2FzfGh0dHBzOi8vdG9rZW5pc3N1ZXI6ODk5My9zZXJ2aWN1cy9TZWNIcm10eVRva2VuU2VydmljZQ==</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</w
st:KeyType>
  <wst:UseKey>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:X509Data>
        <ds:X509Certificate>
MIIC5DCCAk2gAwIBAgIJAKj7ROPHjolyMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUz
EQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZH1lYXIxGDAWBgNVBAoMD0xvY2toZWVkIE
1h
cnRpbjENMAsGA1UECwwESTRDRTEPMA0GA1UEAwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNG
N1
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTAl
VT
MRAwDgYDVQQIDAkBcm16b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2h1ZW
Qg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbG11bnQxHDAaBgkqhkiG9w0BCQEWDW
k0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9Ss
PG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHeyx18Egd
SJ
vvACgGVChEmHndkf9bWU1AOfNaxW+vZwljUkRUvdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAg
MB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnh1NW9+lxBDAfBgNVHSMEGDAwgbQx11VLtY
XL
vFGpFdHnh1NW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUA4GBAHYs2OI0K6yVXz
YS
sKcv2fmfw6XCICGTnyA7BOdAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcD
TR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/O5tywXRT1+freI3bwA
N0
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-b
      08b-0689ffa9c93e</MessageID>
    <To
      xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/add
      ressing/anonymous</To>
    <RelatesTo
      xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:60652909-faca-4e4a-a
      4a7-8a5ce243a7cb</RelatesTo>
    <wsse:Security
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
      rity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
      ity-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-wssecu
      rity-utility-1.0.xsd"
      xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
      rity-secext-1.0.xsd" xmlns:ns4="http://www.w3.org/2005/08/addressing"
      xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200802">
      <RequestSecurityTokenResponse>

        <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1
        #SAMLV2.0</TokenType>
        <RequestedSecurityToken>
          <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            ID="_BDC44EB8593F47D1B213672605113671"
            IssueInstant="2013-04-29T18:35:11.370Z" Version="2.0"
```

```

xsi:type="saml2:AssertionType">
    <saml2:Issuer>tokenissuer</saml2:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                <ds:Reference URI="#_BDC44EB8593F47D1B213672605113671">
                    <ds:Transforms>
                        <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                        <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" >
                            <ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs" />
                            </ds:Transform>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                            <ds:DigestValue>6wnWbft6Pz5XOF5Q9AG59gcGwLY=</ds:DigestValue>
                        </ds:Reference>
                    </ds:SignedInfo>

<ds:SignatureValue>h+NvkgXGdQtca3/eKebhAKgG38tHp3i2n5uLLy8xXXIg02qyKgEP0FC
owp2LiYlsQU9YjKfSwCUbH3WR6jhbAv9zj29CE+ePfEny7MeXvgN13wId+vChqtI/DGGhhgtO2
Mbx/tyX1BhHQUwKRlcHajxHeecwmvV7D85NMdV48tI=</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>

<ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9w0BAQQFADB1MQswCQYDVQ
QGEwJVUzEQMA4GA1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZHl1YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0
V4
YW1wbGUxEDAOBgNVBAstB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxMVoXDT
Iz
MDQwNzE4MzcxMVowgaYxCzAJBgNVBAYTA1VTMRAwDgYDVQQIEwdBcm16b25hMREwDwYDVQQHEw
hH
b29keWvhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRX
hh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaNzdWVyQG
V4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtx
N9
uB44diIIqq3JOzDGfDhGLu6mjpuHO1hrKITv42hBohhmH7ls9ipiaQCIPVfgIG63MB7fa5dBrf
GF
G69vFrU1Lfj7IvsVVNsrtAEQ1jOMmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqPv8Kb0COOGwIDAQ
AB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBhvCAQ0EHxYdT3Blb1NTTCBHZW51cmF0ZWQgQ2
Vy
dG1maWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgnVHSMEgZ8wgZyAFB
cn
en6/j05DzaVwORwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYT

```

ER
MA8GA1UEBxMIR29vZH11YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxED
AO
BgNVBAstB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk70cw07gwwDQYJKoZIhvcNAQEEBQADgY
EA
PiTX5kYXwdhmi justSkrObKpRbQkvkkzcyZlO6VrAxRQ+eFeN6NyuyhgYy5K61/sIWdaGou5iJO
Qx
2pQYWx1v8Klyl0W22IfEAXYv/epiO89hpdACryuDjpioXI/X8TAwvRwLKL21Dk3k2b+eyCgA0O
++
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>MIIC5DCCAk2gAwIBAgIJAKj7ROPHjo1yMA0GCSqGSIb3DQEBCwUAMI
GKMQswCQYDVQQGEwJVUzEQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZH11YXIxGDAWBgNVBAoMD0xvY2toZWVkie
1h
cnRpbjENMASGA1UECwwESTRDRTEPMa0GA1UEAwGY2xpZW50MRwwGgYJKoZIhvcNAQkBfgleNG
N1
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTAl
VT
MRAwDgYDVQQIDAdbcm16b25hMREwDwYDVQQHDAhHb29keWVhcjeYMBYGA1UECgwPTG9ja2h1ZW
Qg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbGllbnQxHDAabgkqhkiG9w0BCQEWDW
k0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAiIpHxCBLYE7xfDLcITS9Ss
PG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHeyx18Egd
SJ
vvACgGVChEmHndkf9bwU1AoFNaXW+vZwljUkRUVDkhPbPdPwOcMdKg/SsLSNjZfsQIjowd4rAg
MB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnh1NW9+lxBDAfBgNVHSMEGDAwgBQx11VLtY
XL
vFGpFdHnh1NW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4GBAHYs2OI0K6yVXz
yS
sKcv2fmfw6XCICGTnyA7BOdAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcD
TR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/O5tywXRT1+freI3bwA
N0
L6tQ</ds:X509Certificate>
 </ds:X509Data>
 </ds:KeyInfo>

```
        </saml2:SubjectConfirmationData>
    </saml2:SubjectConfirmation>
</saml2:Subject>
<saml2:Conditions NotBefore="2013-04-29T18:35:11.407Z"
NotOnOrAfter="2013-04-29T19:05:11.407Z">
    <saml2:AudienceRestriction>

<saml2:Audience>https://server:8993/services/SecurityTokenService</saml2:Audience>
    </saml2:AudienceRestriction>
</saml2:Conditions>
<saml2:AuthnStatement AuthnInstant="2013-04-29T18:35:11.392Z">
    <saml2:AuthnContext>

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml2:AuthnContextClassRef>
    </saml2:AuthnContext>
</saml2:AuthnStatement>
<saml2:AttributeStatement>
    <saml2:Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
xsi:type="xs:string">srogers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
xsi:type="xs:string">srogers@example.com</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
xsi:type="xs:string">srogers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
xsi:type="xs:string">avengers</saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
```

```
xsi:type="xs:string">admin</saml2:AttributeValue>
    </saml2:Attribute>
    </saml2:AttributeStatement>
    </saml2:Assertion>
    </RequestedSecurityToken>
    <RequestedAttachedReference>
        <ns3:SecurityTokenReference
            xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1
            .1.xsd"
            wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profil
            e-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-profil
                    e-1.1#SAMLV2.0">
                    <ns3:KeyIdentifier
                        ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1
                        #SAMLID">_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                    </RequestedUnattachedReference>
                    <wsp:AppliesTo
                        xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
                        xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                        <wsa:EndpointReference
                            xmlns:wsa="http://www.w3.org/2005/08/addressing">

                            <wsa:Address>https://server:8993/services/SecurityTokenService</wsa:Address
                        </wsa:EndpointReference>
                    </wsp:AppliesTo>
                    <Lifetime>
                        <ns2:Created>2013-04-29T18:35:11.444Z</ns2:Created>
                        <ns2:Expires>2013-04-29T19:05:11.444Z</ns2:Expires>
                    </Lifetime>
                </RequestSecurityTokenResponse>
```

```
</RequestSecurityTokenResponseCollection>
</soap:Body>
</soap:Envelope>
```

UsernameToken Bearer SAML Security Token Request/Response

On This Page

- Overview
- Request
 - Explanation
- Response

Overview

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>
```

```

a:Action>
    <wsa:MessageID>uuid:albba87b-0f00-46cc-975f-001391658cbe</wsa:MessageID>
    <wsa:To>https://server:8993/services/SecurityTokenService</wsa:To>
</soap:Header>
<soap:Body>
    <wst:RequestSecurityToken
        xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
        <wst:SecondaryParameters>
            <t:TokenType
                xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oas
is-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</t:TokenType>
                <t:KeyType
                    xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oas
is-open.org/ws-sx/ws-trust/200512/Bearer</t:KeyType>
                    <t:Claims
                        xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
                        xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
                        Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
                        <!--Add any additional claims you want to grab for the service-->
                        <ic:ClaimType Optional="true"
                            Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/uid"/>
                            <ic:ClaimType Optional="true"
                                Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
                                <ic:ClaimType Optional="true"
                                    Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"
                                    />
                                    <ic:ClaimType Optional="true"
                                        Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
                                        <ic:ClaimType Optional="true"
                                            Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
                                            <ic:ClaimType Optional="true"
                                                Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
                                                </t:Claims>
                                                </wst:SecondaryParameters>

<wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</w
st: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-a
      53c-13d960e3b6e8</MessageID>
    <To
      xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/add
      ressing/anonymous</To>
    <RelatesTo
      xmlns="http://www.w3.org/2005/08/addressing">uuid:a1bba87b-0f00-46cc-975f-
      001391658cbe</RelatesTo>
    <wsse:Security
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
      rity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
      ity-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-wssecu
      rity-utility-1.0.xsd"
      xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecu
      rity-secext-1.0.xsd" xmlns:ns4="http://www.w3.org/2005/08/addressing"
      xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200802">
      <RequestSecurityTokenResponse>
```

```

<TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1
#SAMLV2.0</TokenType>
  <RequestedSecurityToken>
    <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      ID="_7437C1A55F19AFF22113672577526132"
      IssueInstant="2013-04-29T17:49:12.613Z" Version="2.0"
      xsi:type="saml2:AssertionType">
      <saml2:Issuer>tokenissuer</saml2:Issuer>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="#_7437C1A55F19AFF22113672577526132">
              <ds:Transforms>
                <ds:Transform
                  Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                <ds:Transform
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                    <ec:InclusiveNamespaces
                      xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs" />
                  </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>ReOqEbGZlyplW5kqiynXOjPnVEA=</ds:DigestValue>
                </ds:Reference>
              </ds:SignedInfo>

              <ds:SignatureValue>X5Kzd54PrK1lGVV2XxzCmWFRzHRoybF7hU6zxbEhSLMR0AWS9R7Me3e
                pq91XqeOwvIDDbwmE/oJNC7vI0fIw/rqXkx4aZsY5a5nbAs7f+aXF9TGdk82x2eNhNGYpViq0Y
                ZJfsJ5WSyMtG8w5nRekmDMy9oTLsHG+Y/OhJDEwq58=</ds:SignatureValue>
                <ds:KeyInfo>
                  <ds:X509Data>

                  <ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9w0BAQQFADB1MQswCQYDVQ
                    QGEwJVUzEQMA4GA1UECBMH
                    QXJpem9uYTERMA8GA1UEBxMIR29vZHl1YXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0
                    V4
                    YW1wbGUxEDAOBgNVBAstB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxMVoXDT
                    Iz
                    MDQwNzE4MzcxMVowgaYxCzAJBgNVBAYTA1VTMRAwDgYDVQQIEwdBcm16b25hMREwDwYDVQQHEw
                    hH
                    b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRX
                    hh
                    bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaNzdWVvQG
                    V4
                    YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtx
                    N9
                    uB44diilqq3JOzDGfDhGLu6mjpuHO1hrKITv42hBOhhmH71S9ipiaQCIpVfgIG63MB7fa5dBrf

```

GF
G69vFrU1Lf17IvsVVNsNrtAEQ1jOMmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqPv8Kb0COOGwIDAQ
AB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBvhCAQ0EHxYdT3Blb1NTTCBHZW51cmF0ZWQgQ2
VY
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgNVHSMEgZ8wgZyAFB
cn
en6/j05DzaVwORwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYT
ER
MA8GA1UEBxMIR29vZH1lYXIxEDAObgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxED
AO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk70cw07gwwDQYJKoZIhvcNAQEEBQADgY
EA
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZl06VrAxRQ+eFeN6NyuyhgYy5K61/sIWdaGou5iJO
Qx
2pQYWx1v8Klyl0W22IfEAXYv/epi089hpdACryuDjpioXI/X8TAwvRwLKL21Dk3k2b+eyCgA0O
++
HM0dPfiQLQ99ElWkv/0=</ds:X509Certificate>
 </ds:X509Data>
 </ds:KeyInfo>
 </ds:Signature>
 <saml2:Subject>
 <saml2:NameID
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
NameQualifier="http://cxf.apache.org/sts">srogers</saml2:NameID>
 <saml2:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
 </saml2:Subject>
 <saml2:Conditions NotBefore="2013-04-29T17:49:12.614Z"
NotOnOrAfter="2013-04-29T18:19:12.614Z">
 <saml2:AudienceRestriction>

 <saml2:Audience>https://server:8993/services/QueryService</saml2:Audience>
 </saml2:AudienceRestriction>
 </saml2:Conditions>
 <saml2:AuthnStatement AuthnInstant="2013-04-29T17:49:12.613Z">
 <saml2:AuthnContext>

 <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspeci
fied</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:wss11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1
.1.xsd"
wss11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profil
e-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:wss11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1
.1.xsd"
wss11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profil
e-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

On This Page

- Overview
- Request
 - Explanation
- Response
 - Explanation

Overview

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-wssecur  
ity-secext-1.0.xsd"  
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
```

```

ity-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-tok
en-profile-1.0#PasswordText">password1</wsse:Password>
        <wsse:Nonce>
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap
-message-security-1.0#Base64Binary">LCTD+5Y7hlWIP6SpsEg9XA==</wsse:Nonce>
        <wsu:Created>2014-02-19T17:30:37.355Z</wsu:Created>
        </wsse:UsernameToken>
    </wsse:Security>
</soapenv:Header>
<soapenv:Body>
    <wst:RequestSecurityToken
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">

    <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile
-1.1#SAMLV2.0</wst:TokenType>

    <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey</w
st:KeyType>

    <!-- OPTIONAL: Only required if the endpoint that the SAML
assertion will be sent to requires claims. -->
    <wst:Claims
Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity"
xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity">
        <ic:ClaimType Optional="true"
Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
        <ic:ClaimType Optional="true"
Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"
/>
        <ic:ClaimType Optional="true"
Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
        <ic:ClaimType Optional="true"
Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
        <ic:ClaimType Optional="true"
Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
    </wst:Claims>

    <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</w
st: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>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1
UEBhMCVVMxGDAWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBqNVBAMTDk
RP
RCBKSVRDIEBLTI3MB4XDTEzMDUwNzAwMjU0OVoXDTE2MDUwNzAwMjU0OVowaTELMAkGA1UEBh
MC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0
kx
EzARBgNVBAsTCkNPT1RSQUNUT1IxDzANBgNVBAMTBmNsawWVudDCCASIwDQYJKoZIhvcNAQEBBQ
AD
ggEPADCCAQoCggEBAOq6L1/jjZ5cyhjhHEbOHr5WQpbokACYbrsn8lg85LGNoAfewImr9KBmOx
Gb
ZCxHYIhkW7pJ+kppH8DDMviIvvdkvrAIU018OBn2wReCBGQ01Imdc3+WzFF2svW75d6wi2Z
Vd
eMvU015p/pAD/sdIfXmAfyu8+tqtio8KVZGkTnlg3AMzfeSrkci5UHMVWj0qUSuzLk9SAg/9ST
gb
Kf2xBpHUYecWFSB+dTpdzN2pC85tj9xIoWGH5dFWG1fPcYRgzGPxsybiGOylbJ7rHDJuL7IIiy
x5
EnkCuxmQwoQ6XQAhWRGyP1Y08w1LZixI2v+Cv/ZjUFIHv49I9P4Mt8CAwEAAAOCAdUwggHRMB
8G
A1UdIwQYMBaAFCMUNCBNxy43NZLBBlnDjDplNZJoMB0GA1UdDgQWBGRPGiX6zzKTqQSx/tjg6
hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2Nybc5nZH
Mu
bml0LmRpC2EubWlsL2Nybc9ET0RKSVDQ0FfMjcuY3JsMIGUoIGRoIGOhGLbGRhcDovL2Nybc
5n
ZHMubml0LmRpC2EubWlsL2NuJTNkRE9EJTIwSk1UQyUyMENBLTI3JTJjb3U1M2RQS0k1MmNvdS
Uz
ZERvRCUyY28lM2RVlMuJTIwR292ZXJubWVudCUyY2M1M2RVUz9jZXJ0awZpY2F0ZXJ1dm9jYX
Rp
b25saXN0O2JpbmFyeTAjBgNVHSAEHDAAcMAcGCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKw
YB
BQUHAQEEcTBvMD0GCCsGAQUFBzACHjFodHRwOi8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi
9E
T0RKSVDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzcC5uc24wLnJjdnMubml0Lm
Rp
c2EubWlsMA0GCSqGSIb3DQEBCQUAA4IBAQCGUJPGh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvu
p1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIs
im
EhmzSgzxnt4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinI
Hr
hwtE3xo6qQRWl03/c1C4RnTev1crFVJQVBF3yfpRu8udJ2SOGdqU0vjUSu1h7aMkYJMHIu08W
hj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCxg1Ljc4dmbjkVr9s4QK+/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-a
      e07-b9e10c7911f3</MessageID>
    <To
      xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/add
      dressing/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-wssecu
      rity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
      ity-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-wssecur
      ity-utility-1.0.xsd"
      xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
```

```

ity-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+UuYlGOuTwDYSUPcS2qI0nAgMD1VsDP7y1fDJxeqsq7HY
hFKsnqRfebMM4WLH1D/lJ4rD4UO+i913tuiHml7SN24WM1/bOqfDUCoDqmwG8afUJ3r4vmTNPW
TfOss8BZ/8ODgZzm08ndlkhxDfvcN7OrExbV/3/45JwF/MMPZoqvi2MJGfx56E9fErJNuzezpWn
RqP0lWPxyffKMA1VaB9zF6gvVnUqcW2k/Z8X91N705jouBI281ZnIfsIPuBJERFtYNVDHsIXM1
pjnrY6FlKIaOsi55LQu3Ruirl/n82pU7BT5aWtxwrn7akBg==</ds:SignatureValue>
            <ds:KeyInfo>
                <ds:X509Data>
```

<ds:X509Certificate>MIIFHTCCBAwGAwIBAgICJe8wDQYJKoZIhvCNQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT

D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP

RCBKSVRDIEBLTI3MB4XDTEzMDUwNzAwMjYzN1oXDTE2MDUwNzAwMjYzN1owbjELMAkGA1UEBhMC

VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0

kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxFDASBgNVBAMTC3Rva2VuaXNzdWVyMIIBIjANBgkqhkiG9w
OB
AQEFAAOCQ8AMIIIBCgKCAQEAx01/U4M1wG+wL1JxX2RL1glj101FkJXMk3Kft3zD//N8x/Dcww
vs
ngCQjXrV6YhbB2V7scHwnThPv3RSwYYiO62z+g6ptfBbKGGBLSZOzLe3fyJR4RxblFKsELFgPH
fx
vgUHS/keG5uSRk9S/Okqps/yxKB7+ZlxefxsIz5QywXvBpMiXtc2zF+M7BsbSIDSx5LcPcDFBw
jF
c66rE3/y/25VMht9EZx1QoKr7f8rWD4xgd5J6DYMFWEcniCz4BDJH9sfTw+n1P+CYgrhwslWGq
xt
cDME9t6SWR3GLT4Sdtr8ziIM5uUteEhPIV3rVC3/u23JbYEeS8mpnp0bxt5eHQIDAQABo4IB1T
CC
AdEwHwYDVR0jBBgwFoAUIxQ0IE1fLjc1ksEGWcOMOmU1kmgwHQYDVR0OBByEFGBjdkdey+bMHM
hC
Z7gwiQ/mJf5VMA4GA1UdDwEB/wQEAWIFoDCB2gYDVR0fBIHSMIHPMDagNKAYhjBodHRwOi8vY3
Js
Lmdkcy5uaXQuZGlzYS5taWwvY3JsL0RPRepJVENDQV8yNy5jcmwwgZSggZGggY6GgYtsZGFwOi
8v
Y3JsLmdkcy5uaXQuZGlzYS5taWwvY241M2RET0Q1MjBKSVRDJTIwQ0EtMjclMmNvdSUzzFBLSS
Uy
Y291JTnkRG9EJTJjbyUzzFUuUy41MjbHb3z1cm5tZW50JTJjYyUzzFVTp2N1cnRpZmljYXRlc
v2
b2NhdG1vbmxpc3Q7YmluYXJ5MCMGA1UdIAQcMBowCwYJYIZIAWUCAQsFMAsgCWCGSAFlAgELEj
B9
BggrBgEFBQcBAQRxMG8wPQYIKwYBBQUHMAKGMWh0dHA6Ly9jcmwuZ2RzLm5pdC5kaXNhLm1pbC
9z
aWduL0RPRepJVENDQV8yNy5jZXIwLgYIKwYBBQUHMAGGImh0dHA6Ly9vY3NwLm5zbjAucmN2cy
5u
aXQuZGlzYS5taWwwDQYJKoZIhvcNAQEFBQADggEBAlHZQTINU3bMpJ/PkwTYLWPmwCqAYgEUzS
Yx
bNcVY5MWD8b4XCdw5nM3GnFl0qr4IrHeyyOzsEbIebTe3bv011pHx0Uyj059nAhx/AP8DjVtuR
U1
/Mp4b6uJ/4yaoMjIGceqBzHqhHIJinG0Y2azua7eM9hvWZsa912ihbiupCq22mYuHFP7NUNzB
vv
j03YUcsy/sES5sRx9Rops/CBN+LUUYOdJOxYWxo8oAbtF8ABE5ATLAwqz4ttsToKPUYh1sxidx5
Ef
APeZ+wYDmMu4OfLckwnCKZgkEtJOxXpdIJHY+VmyZtQSB0LkR5toeH/ANV4259Ia5ZT8h2/vIJ
Bg
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>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT  
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBqNVBAMTDk  
RP  
RCBKSVDIENBLTI3MB4XDTEzMDUwNzAwMjU0OVoXDTE2MDUwNzAwMjU0OVowaTELMAkGA1UEBh  
MC  
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0  
kxF  
EzARBgNVBAsTCkNPT1RSQUNUT1IxZANBgNVBAMTBmNsawVudDCCASIwDQYJKoZIhvcNAQEBBQ  
AD  
ggEPADCCAQoCggEBAOq6L1/jjZ5cyhjhHEbOHr5WQpbokACYbrsn8lg85LGNoAfcwImr9KBmOx  
Gb  
ZCxHYIhkW7pJ+kppyH8DDMviIvvdkvrAIU0180BRn2wReCBGQ01Imdc3+WzFF2svW75d6wi2Z  
Vd  
eMyU015p/pAD/sdIfXmAfyu8+tqtio8KVZGkTnlg3AMzfeSrkci5UHMVWj0qUSuzLk9SAg/9ST  
gb  
Kf2xBpHUYecWFSB+dTpZn2pC85tj9xIoWGH5dFWG1fPcYRgzGPxsybiGOylbJ7rHDJuL7IIy  
x5  
EnkCuxmQwoQ6XQAhWRGyPlY08w1LZixI2v+Cv/ZjUFIHv49I9P4Mt8CAwEAAAOCAdUwggHRMB  
8G  
A1UdIwQYMBaAFCMUNCBNXy43NZLBBlnDjDplNZJoMB0GA1UdDgQWBFRPGiX6zzKTqQSx/tjg6  
hx  
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2Nybc5nZH  
Mu  
bm10LmRpc2EubWlsL2Nybc9ET0RKSVRDQ0FfMjcuY3JsMIGUoIGRoIGOhogLbGRhcDovL2Nybc  
5n  
ZHMubm10LmRpc2EubWlsL2NuJTNkRE9EJTIwSk1UQyUyMENBLTI3JTJjb3U1M2RQS0k1MmNvdS  
Uz  
ZERvRCUyY281M2RVllMuJTIwR292ZXJubWVudCUyY2M1M2RVUz9jZXJ0aWZpY2F0ZXJ1dm9jYX  
Rp  
b25saXN0O2JpbmFyeTAjBgNVHSAEHDAAmAsgCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKw  
YB  
BQUHAQEeCTBvMD0GCCsGAQUFBzAChjFodHRwOi8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi  
9E  
T0RKSVRDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzCC5uc24wLnJjdnMubm10Lm  
Rp  
c2EubWlsMA0GCSqGSIb3DQEBBQUAA4IBAQCGUJPGh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvu  
p1  
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtFWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIs  
im  
EhmzSgznT4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinI  
Hr  
hwtE3xo6qQRRL03/c1C4RnTev1crFVJQVF3yfpRu8udJ2SOGdqU0vjUSulh7aMkYJMHIu08W  
hj  
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">dodusers</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>

```

XACML Policy Decision Point (PDP)

On This Page

- Overview
- Creating a Policy
 - Actions
 - Attributes
- Example Requests and Responses
 - Policy
 - Service Authorization
 - Metocard Authorization

Overview

After unzipping the DDF distribution, place the desired XACML policy in the <distribution root>/etc/pdp/policies directory. 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.

Creating a Policy

This document assumes familiarity with the XACML schema and does not go into detail on the XACML language. There are some DDF-specific items that need to be considered when creating a policy, so it is compatible with the XACMLRealm. 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. Generally targets are geared toward the actions that are set in the request.

Actions

For DDF, these actions are populated by various components in the security API. The actions and their population location are identified in the following table.

Operation	Action-id Value	Component Setting the action	Description
Filtering / Redaction	filer	security-pdp-xacmlrealm	When performing any redaction or filtering, the XACMLRealm will set the action-id to "filter".
Service	<SOAPAction>	security-pep-interceptor	If the PEP Interceptor is added to any SOAP-based web services for service authorization, the action-id will be the SOAPAction of the incoming request. This allows the XACML policy to have specific rules for individual services within the system.

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

In the examples below, the policy has specified targets for the above type of calls. For the Filtering/Redaction 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 XACMLRealm 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 XACMLRealm retrieves a list of permissions that should be checked against the subject. These permissions are populated outside of the realm and should be populated with the security attributes located in the metocard security property. When the permissions are of a key-value type, the key being used is populated as the AttributId 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. For the XACML PDP, the request is made by the XACML realm (security-pdp-xacmlrealm), passed to the XACML processing engine (security-pdp-xacmlprocessor), which 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 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</Attribut
```

```

eValue>
    <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="http://www.opm.gov/fedata/CountryOfCitizenship"
Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
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</Attribut
                eValue>
            <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"/>
</Policy>

```

Service Authorization

Allowed Query

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">query</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="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://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">TestUser</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>
</Request>
```

Response

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

Denied Query

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">query</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 USA</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="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://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">TestUser</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/feddata/CountryOfCitizenship" IncludeInResult="false">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">USA</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>
```

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">TestUser</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/feddata/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">TestUser</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>

```

Expansion Service

On This Page

- Overview
- Instances and Configuration
- Expansion Commands
 - Expansion Commands
 - Command Descriptions
- Expansion Command Examples and Explanation
 - security:expansions
 - security:expand

Overview

The Expansion Service and its corresponding expansion-related commands provide an easy way for developers to add expansion capabilities to DDF during user attributes and metadata card processing. In addition to these two defined uses of the expansion service, developers are free to utilize the service in their own implementations.

Each instance of the expansion service consists of a collection of rule sets. Each rule set consists of a key value and its associated set of rules. Callers of the expansion service provide a key and an original value to be expanded. The expansion service then looks up the set of rules for the specified key. The expansion service then cumulatively applies each of the rules in the set starting with the original value, with the resulting set of values being returned to the caller.

Key (Attribute)	Rules (original new)	
key1	value1	replacement1
	value2	replacement2
	value3	replacement3
key2	value1	replacement1
	value2	replacement2

The examples below use the following collection of rule sets:

Key (Attribute)	Rules (original new)	
Location	Goodyear	Goodyear AZ
	AZ	AZ USA
	CA	CA USA

Title	VP-Sales	VP-Sales VP Sales
	VP-Engineering	VP-Engineering VP Engineering

Note that the rules listed for each key are processed in order, so they may build upon each other, i.e., a new value from the new replacement string may be expanded by a subsequent rule.

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	security.user.attribute.mapping	This instance is configured with rules that expand the user's attribute values for security checking.
mapping	security.metocard.attribute.mapping	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
```

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.
------------------------------------	----------	---

Expansion Commands

security:expand	security:expansions
-----------------	---------------------

Command Descriptions

Command	Description
expand	Runs the expansion service on the provided data returning the expanded value.
expansions	Dumps the ruleset for each active expansion service.

Expansion Command Examples and Explanation

security:expansions

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

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

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

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

security:expand

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

```

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

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

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

```

Configure WSS Using Standalone Servers

On This Page

- Overview
- Configure LDAP, CAS, and STS
 - LDAP
 - CAS
 - STS
- Configure DDF

Overview

DDF uses CAS as its single sign-on service. DDF uses LDAP and STS to keep track of users and user attributes. CAS, LDAP, and STS are integral, interconnected components of the DDF security scheme, and all can be installed on a local DDF instance with only a few feature installs (with the exception of the [CAS installation](#), which requires Apache Tomcat to run). 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.

This page assumes that there is a keystore for each of the services/servers. 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
CAS	keystore.jks	Used on the CAS Tomcat server.
STS	stsKeystore.jks	Used to sign SAML and as incoming connections.
DDF	serverKeystore.jks	Server used for incoming connections.
	clientKeystore.jks	Client used for outgoing connections.

[Distributed WSS.pptx](#)

Login Authentication Scheme

Configure LDAP, CAS, and STS

It is implied 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

LDAP is used to maintain a list of trusted DDF users and the attributes associated with them. It interacts with both CAS and the STS. The former uses LDAP to create session information, and the latter queries LDAP for user attributes and converts them to SAML claims.

1. Obtain and unzip the DDF kernel (`ddf-distribution-kernel-<VERSION>.zip`).
2. Start the distribution.
3. Deploy the Embedded LDAP application by copying the `ldap-embedded-app-<VERSION>.kar` into the `<DISTRIBUTION_HOME>/deploy` directory. Verify that the LDAP server is installed by checking the DDF log or by performing an `la` command and verifying that the

- OpenDJ bundle is in the active state. Additionally, it should be responding to LDAP requests on the default ports, which are 1389 and 1636.
4. Copy the environment's Java keystore file into the \${DISTRIBUTION}/etc/keystores folder, making sure it overwrites the folder's existing serverKeystore.jks file.

It is very important that the keystore file used in the process is set up to trust the hostnames used by CAS and STS. If it is not, there will be certificate authentication issues for the user.

CAS

CAS is used for SSO authentication purposes. Unlike LDAP and STS, CAS cannot be run as a DDF bundle. CAS must be run through Apache Tomcat.

1. Follow the instructions on the [CAS installation](#) page to install and configure Tomcat/CAS. For example, with LDAP above, the keystore.jks file that is used must trust the hostnames used by the STS server, LDAP server, and the DDF user connecting to CAS.
2. Open the \${TOMCAT}/webapps/cas/WEB-INF/cas.properties file and modify the cas.ldap.host, cas.ldap.port, cas.ldap.user.dn, and cas.ldap.password fields with your environment's LDAP information.

STS

The Security Token Service, unlike the LDAP, cannot currently be installed on a kernel distribution of DDF. To run an STS-only DDF installation, uninstall the catalog components that are not being used. This will increase performance. A list of unneeded components can be found on the [STS page](#).

1. In the unzipped DDF distribution folder, open /etc/org.ops4j.pax.web.cfg and find the following line:

```
org.ops4j.pax.web.ssl.keystore=etc/keystores/serverKeystore.jks
```

and change it to:

```
org.ops4j.pax.web.ssl.keystore=etc/keystores/stsKeystore.jks
```

2. Update the password fields to the ones you keystore uses.
3. Verify that the stsKeystores.jks file in /etc/keystores trusts the hostnames used in your environment (the hostnames of LDAP, CAS, and any DDF users that make use of this STS server).
4. Start the distribution.
5. Enter the following commands to install the features used by the STS server:

```
features:install security-sts-server  
features:install security-cas-tokenvalidator
```

6. Open the DDF web console as an administrator. The default user is "admin" with a password of "admin" (no quotes).
7. Navigate to the Configuration tab.
8. Open the Security STS LDAP Login configuration.
9. 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 username is "cn=user", and the default password is "secret" (no quotes). In a production environment, the username and password should be changed in the LDAP data file.
10. Select the **Save** button.
11. Open the **Security STS LDAP and Roles Claims Handler** configuration.
12. Populate the same URL, user, and password fields with your LDAP server information.
13. Select the **Save** button.
14. Open the **Security STS CAS Token Validator** configuration.
15. Under **CAS Server URL**, type the URL for your CAS server.
16. Select the **Save** button.
17. Open the **Platform Global Configuration**.
 - a. Change the protocol to https.
 - b. Populate the host/port information with the STS server's host/port. For STS, the default port is 8993.
 - c. Update the **Trust Store** and **Key Store** location/password fields with your environment's .jks files.
 - d. Select the **Save** 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.

Configure DDF

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. Verify that the \${DISTRIBUTION}/etc/keystores folder is updated with the correct keystores for your operating environment.
2. Start the distribution.
3. Enter the following commands to install the CAS features:+

```
features:install security-cas-cxfservletfilter
```

4. Open the **Security CAS Client** configuration.
 - a. Under **Server Name** and **Proxy Callback URL**, replace the hostname of 'server' with your server hostname.
 - b. Under **CAS Server URL**, enter the hostname for the CAS server.
5. Open the **Security STS Client** configuration. Verify that the host/port information in the **STS Address** field points to your STS server.
6. Open the **Platform Global Configuration**.
 - a. Change the protocol to https.
 - b. Populate the host/port information with the DDF instance's host/port. For DDF, the default port is 8993.
 - c. Update the **Trust Store** and **Key Store** location/password fields with your environment's .jks files.
 - d. Select the **Save** button.

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

Hardening

On This Page

- Overview
- Disable the Web Console
- Enable SSL for Clients
 - Configure DDF without the Web Administration Console
 - Configure Using a .cfg File Template
 - Configure Using the Command Line Console

Overview

These instructions demonstrate how to harden a DDF system for a more secure installation.

The web administration console is not compatible with Internet Explorer 7.

Disable the Web Console

To harden DDF for security purposes, disable the Web Console, so users do not have access. All configuration is performed using the DDF command line console and/or .cfg configuration files.

1. Open the Web Console: <http://localhost:8181/system/console>.
2. Enter the username (default is "admin") and the password (default is "admin").
3. Select the Features tab.
4. Select the **Install/Uninstall** button to uninstall the webconsole-base feature. If attempting to access a page on the Web Console, an HTTP 404 error occurs.
5. To re-enable the Web Console, open the Karaf command line console and install the ddf-webconsole feature.

```
features:install webconsole
```

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/serverKeystore.jks	The java keystore that contains the trusted public certificates for Certificate Authorities (CA's) 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/serverTruststore.jks	The keystore that contains the private key for the local server that can be used to signing and encryption. This must be set if establishing outgoing 2-way (mutual) SSL connections where the local server must also present it's certificate for the remote server to verify.
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

Configure DDF without the Web Administration Console

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

In an environment hardened for security purposes, access to the DDF Web Console is denied. It is necessary to configure DDF (e.g., providers, Schematron rulesets, etc.) using .cfg configuration files or the Karaf command line console. The OSGi container detects the addition, updating, or deletion of .cfg files in the etc/ddf directory.

The following sections describe how to configure each DDF item using both of these mechanisms. A template file is provided for each configurable DDF item so that it can be copied/renamed then modified with the appropriate settings.

If at a later time the Web Console is enabled again, all of the configuration done via .cfg files and/or the Karaf command line console are loaded and displayed. However, note that the name of the .cfg 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 (from DDF User's Guide)
DDF Catalog Framework	ddf.catalog.impl.service.CatalogFrameworkImpl.cfg	ddf.catalog.CatalogFrameworkImpl	Standard Catalog Framework

Configure Using a .cfg File Template

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

1. Copy/rename the provided template file in the `etc/templates` directory to the `etc` directory. (Refer to the table above to determine correct template.)
 - **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.
 - 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).
 - This file is a Java properties file, hence the syntax is `<key> = <value>`.
 - Consult the inline comments in the file for guidance on what to modify.
 - 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 Web Console.

Configure Using the Command Line Console

Configuring a new source, provider, or feature using the command console follows a standard procedure. The properties and their values will change based on type of service being created, but the actual commands entered at the command line do not change. To help illustrate the commands, an example of creating a new OpenSearch federated source is shown after each step.

1. Create the federated source using the `config:edit` command.

- **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.

```
config:edit OpenSearchSource-my_federated_source
```

2. Enter the settings for the federated source's properties. These properties can be found in the template file for the specified service.

```
config:propset endpointUrl
http://ddf:8181/services/query?q={searchTerms}&src={fs:routeTo?}&
amp;mr={fs:maxResults?}&count={count?}&mt={fs:maxTimeout?}&am
p;dn={idn:userDN?}&lat={geo:lat?}&lon={geo:lon?}&radius={
geo:radius?}&bbox={geo:box?}&polygon={geo:polygon?}&dtsta
rt={time:start?}&dtend={time:end?}&dateName={cat:dateName?}&a
mp;filter={fsa:filter?}&sort={fsa:sort?}
```

3. Save the configuration updates.

```
config:update
```

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

Directory Permissions

On This Page

- Directory Permissions on Windows
- Directory Permissions on *NIX

DDF_HOME

DDF is installed in the `DDF_HOME` directory.

Directory Permissions on Windows

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

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 DDF_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 DDF_HOME/etc and DDF_HOME/deploy.

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.

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 DDF_HOME is /opt/ddf):

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

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

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

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.

Deployment Guidelines

On This Page

- Overview
- Endpoint Schema Validation

Overview

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.

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

Endpoint Schema Validation

SOAP Web Services may have WSDL validation enabled. Ensure that the bundle has WSDL schema validation enabled. These instructions assume the implementation made use of the Spring beans model. All DDF endpoint bundles follow this model.

- Prior to deploying a bundle/feature, verify the schema validation if it is a DDF endpoint.
- Modify the beans.xml file
 - In a terminal window, change directory to the feature directory under the DDF installation directory.

```
cd DDF_HOME/system/com/lmco/ddf/endpoint-bundle.jar
```

- Unzip the endpoint-bundle.jar.

```
unzip endpoint-bundle.jar
```

- Change directory to the beans.xml file.

```
cd META-INF/spring
```

- Open the beans.xml file in an editor (e.g., vi).
- Search for schema-validation-enabled and change its value to true.

```
<entry key="schema-validation-enabled" value="true"/>
```

- Save and close the file.
- Change directory to the feature directory,

```
cd ../../..
```

- Recreate the jar file (use zip or another archive tool).

```
zip endpoint-bundle.jar *
```

- Re-install the feature.
 - Open the Web browser and navigate to the DDF Web Console,
 - Select the **Install** button. Schema validation is now enabled for the endpoint.

Assuring Authenticity

On This Page

- Introduction
- Prerequisites
- Signing a JAR/KAR
- Verifying a JAR/KAR

Introduction

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 <http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html>

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:
```

Signing a JAR/KAR

Once a keystore is available, the

Additional documentation on `jarsigner` can be found at <http://docs.oracle.com/javase/6/docs/technotes/tools/windows/jarsigner.html>

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.j
ar
    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-c
ore-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-comma
nds-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/catal
og-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-reg
istry-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-c
ore-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-catalogfr
amework-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-endpo-
int-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/catalo-
g-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-ext-
ensions-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.dom
4j/
0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.dom
4j/1.6.1_5/
smk 325676 Thu Feb 13 09:33:56 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.dom
4j/1.6.1_5/org.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.jdo
m/
0 Wed Sep 17 17:14:10 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.jdo
m/1.1.2_1/
smk 160101 Thu Feb 13 09:33:56 MST 2014
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.jdo
m/1.1.2_1/org.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-httpcl
ient-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-f
ederation-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/cata
log-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/cat
alog-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/cata-
log-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/geoj-
son-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-t
ransformer-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-collectio
ns-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-secu
rity-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-secu
rity-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-sec
```

urity-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.

Security

On This Page

- Manage Users and Passwords
- Enable Password Encryption
- Known Issues

Manage Users and Passwords

The default security configuration uses a property file located in `DDF_HOME/etc/user.properties` to store users and passwords.

The default Web Console user is "admin" (no quotes) with a password of "admin" (no quotes). Change this password to a more secure password by editing this file.

user.properties

```
Format:  
USER=PASSWORD,ROLE1,ROLE2,...  
  
Current default:  
admin=admin,admin
```

Enable Password Encryption

In the DDF Text Console, enter the following commands:

Enable Password Encryption

```
ddf@local> config:edit --force org.apache.karaf.jaas  
ddf@local> config:propset encryption.enabled true  
ddf@local> config:update  
ddf@local> dev:restart
```

The passwords will then be encrypted in the `users.properties` file once DDF restarts.

Known Issues

A system administrator must block visual access to the screen when administering passwords for particular components, such as the OpenSearch source. This is a known issue and will be addressed in a future version of DDF.

Starting DDF

On This Page

- Overview
- Start DDF
 - *NIX
 - Windows
- Stop DDF
- Automatic Start on System Boot
- Karaf Documentation
- Troubleshooting
- Exception Starting DDF
 - Problem:
 - Solution:

Overview

Follow the below steps to start and stop DDF.

Start DDF

*NIX

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

```
DDF_INSTALL/bin/ddf
```

Windows

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

```
DDF_INSTALL/bin/ddf.bat
```

Stop DDF

There are two options:

- Call shutdown from the console:

Shuts down with a prompt

```
ddf@local>shutdown
```

Force Shutdown without prompt

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

- 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 Ctrl-D shortcut (Windows) from the command line console.

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> features: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 `DDF_INSTALL/bin/ddf-service`.

Change:

DDF_INSTALL/bin/ddf-service

```
#RUN_AS_USER=
```

to:

DDF_INSTALL/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.

DDF_INSTALL/etc/ddf-wrapper.conf

```
#Add the following:  
wrapper.java.additional.9=-XX:PermSize=128m  
wrapper.java.additional.10=-XX:MaxPermSize=512m  
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.

DDF_INSTALL/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.

```
DDF_INSTALL/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
```

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>

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 at <http://karaf.apache.org/index/documentation.html>.

Troubleshooting

Exception Starting DDF

Problem:

An exception is thrown starting DDF on a Windows machine (x86).

If using an unsupported terminal, `java.lang.NoClassDefFoundError: Could not initialize class org.fusesource.jansi.internal.Kernel32` is thrown.

Solution:

Install missing Windows libraries.

Some Windows platforms are missing libraries that are required by DDF. These libraries are provided by the Microsoft Visual C++ 2008 Redistributable Package x64 (<http://www.microsoft.com/en-us/download/details.aspx?id=15336>).

Console Commands

On This Page

- Overview
- Access the System Console
- Example Commands
 - View Bundle Status
 - View Installed Features

Overview

Once the distribution has started, users will have access to a powerful command line console. This text 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 Shell Console is the console that is available to the user when the distribution is started manually. It may also be accessed from the Web Console through the Gogo tab or 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 `osgi:list` on the console to view the status of the bundles loaded in the distribution.

View Installed Features

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

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

Catalog Commands

On This Page

- Catalog Commands
 - Commands
 - Command Descriptions
- Available System Console Commands
- System Console Command Help
 - Example Help
- `catalog:dump` Options
- Date Syntax
 - Examples
- Known Command Issues

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.

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 (`--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:range</code>
<code>catalog:latest</code>	<code>catalog:migrate</code>	
<code>catalog:remove</code>	<code>catalog:removeall</code>	
<code>catalog:replicate</code>	<code>catalog:search</code>	<code>catalog:spatial</code>

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.
<code>envlist</code>	<div style="border: 2px solid red; padding: 5px;"><p>Deprecated as of ddf-catalog 2.5.0. Please use <code>platform:envlist</code>.</p></div> <p>Provides a list of environment variables.</p>
<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 <code>Metocard.MODIFIED</code> date.
<code>migrate</code>	Allows two CatalogProviders 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.

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
```

Known Command Issues

Description
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.

Command Scheduler

On This Page

- Overview
- Usage
 - Schedule a Command
 - Update a Scheduled Command
 - Command Output

Overview

Command Scheduler is a capability exposed through the Web Console (<http://localhost:8181/system/console>) that allows administrators to schedule Command Line Shell 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 Web Console (<http://localhost:8181/system/console>).
2. Select the Configuration tab.
3. Find the **Platform Command Scheduler** configuration.
4. Click on the title or the + button. This will create a new configuration.
5. 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.
6. Type in a positive integer for the **Interval In Seconds** field.
7. 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.

Scheduled Commands can be updated and deleted. To delete, click on the trash bin button to the right of the page. To update, modify the fields and click Save.

Update a Scheduled Command

1. Navigate to the Web Console (<http://localhost:8181/system/console>).
2. Select the Configuration tab.
3. The scheduled commands are displayed under the **Platform Command Scheduler**. Scheduled commands use the following syntax: `ddf.platform.scheduler.Command.{GUID}`, such as `ddf.platform.scheduler.Command.4d60c917-003a-42e8-9367-1da0f822ca6e`.
4. Find the configuration to modify.
5. Update either the **Command** field, the **Interval In Seconds** field, or both fields.
6. Select the **Save** button. Once the **Save** button is selected, the command is executed immediately. Its 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.

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.

Subscriptions Commands

On This Page

- Subscriptions Commands
 - Commands
 - Command Descriptions
 - List Available System Console Commands
- System Console Command Help
 - Example Help
- `subscriptions:list` Command Usage Examples
 - List All Subscriptions
 - List a Specific Subscription by ID
 - List Subscriptions Using Wildcards
 - List Subscriptions Using an LDAP Filter
- `subscriptions:delete` Command Usage Example
 - Delete a Specific Subscription Using Its Exact ID
 - Delete Subscriptions Using Wildcards
 - Delete All Subscriptions
 - Delete Subscriptions Using an LDAP Filter

Subscriptions Commands

Title	Namespace	Description
-------	-----------	-------------

DDF :: Catalog :: Core :: PubSub Commands	subscriptions	The DDF PubSub shell commands provide functions to list the registered subscriptions in DDF and to delete subscriptions.
---	---------------	--

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

Commands

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

Command Descriptions

Command	Description
delete	Deletes the subscription(s) specified by the search phrase or LDAP filter.
list	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?WS-DL
my.contextual.id.json|http://172.18.14.169:8088/services/json-ddms/local/event/notification
```

List a Specific Subscription by ID

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

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?WS
DL
my.contextual.id.json|http://172.18.14.169:8088/services/json-ddms/local/e
vent/notification
```

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

Total subscriptions found: 1

Subscription ID
my.contextual.id.json|http://172.18.14.169:8088/services/json-ddms/local/e
vent/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?WS
DL
```

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-ddms/local/e
vent/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-ddms/local/e  
vent/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-ddms/local/  
event/notification"  
  
Deleted subscription for ID =  
my.contextual.id.json|http://172.18.14.169:8088/services/json-ddms/local/e  
vent/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?WS_DL

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-ddms/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?WS_DL
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-ddms/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?WS-DL

Deleted 2 subscriptions out of 2 subscriptions found.

```

Application Commands

On This Page

- Application Commands
 - Commands
 - Command Descriptions
 - List Available System Console Commands
- System Console Command Help
 - Example Help
- Command Usage Examples
 - List All Applications
 - View the Status of an Application
 - Application in Failure State
 - Display a Dependency Tree of Applications

Application Commands

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

The Application Commands are installed automatically with the Admin application.

Commands

```

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

```

Command Descriptions

Command	Syntax	Description
add	app:add appUri	Adds an application with the given URL.
remove	app:remove appName	Removes an application with the given name.

start	app:start appName	Starts an application with the given name.
stop	app:stop appName	Stops an application with the given name.
list	app:list	Lists the applications that are in the system and displays their current state.
status	app:status appName	Shows status of an application. Gives information on the current state, features within the application, displays the required features that are not started, and displays the required bundles that are not started.
tree	app:tree	Creates a hierarchy tree of all of the applications.

List Available System Console Commands

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

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

System Console Command Help

For details on any command type `help` then the application command. For example, `help app:list` displays the information in the following table.

Example Help

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

    Lists the applications that are in the system and gives their current
    state.

    There are four possible states:
        ACTIVE: no errors, started successfully
        FAILED: errors occurred while trying to start
        INACTIVE: nothing from the app is installed
        UNKNOWN: could not determine status.

SYNTAX
    app:list [options]

OPTIONS
    --help
        Display this help message
```

Command Usage Examples

List All Applications

```
ddf@local>app:list
State      Name
[ACTIVE] catalog-app-2.3.0
[ACTIVE] distribution-kernel-2.3.0
[ACTIVE] platform-app-2.3.0
```

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

View the Status of an Application

```
ddf@local>app:status catalog-app-2.3.0
catalog-app-2.3.0

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, there is a required feature or bundle that is not started.

```
ddf@local>app:list
State      Name
[FAILED   ] catalog-app-2.3.0
[ACTIVE    ] distribution-kernel-2.3.0
[ACTIVE    ] platform-app-2.3.0
```

In the above example, the catalog app is in a failure state. Checking the status of that application indicates what did not start correctly.

```
ddf@local>app:status catalog-app-2.3.0
catalog-app-2.3.0

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 the reason for the failure.

```
[ 261] [Resolved     ] [           ] [           ] [     80] DDF :: Catalog :: 
OpenSearch :: Endpoint (2.3.0)
```

Once that bundle is started by fixing its error, the catalog application changes to the Active state.

Display a Dependency Tree of Applications

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

Platform Commands

On This Page

- Platform Commands
- Commands
 - Command Descriptions
 - List Available System Console Commands
- System Console Command Help
 - Example Help

Platform Commands

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

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

Commands

Command Descriptions

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

Command	Description
describe	Shows the current platform configuration.
envlist	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.

Persistence Commands

On This Page

- Persistence Commands
 - Commands
 - Command Descriptions
- Available System Console Commands
- System Console Command Help
 - Example Help
- CQL Syntax
 - Examples

Persistence Commands

Title	Namespace	Description
DDF:: Persistence :: Core :: Commands	store	The Persistence Shell Commands are meant to be used with any <code>PersistentStore</code> implementation s. They provide the ability to query and delete entries from the persistence store.

Commands

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

Command Descriptions

Command	Description
delete	Delete entries from the persistence store that match a given CQL statement
list	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.

CQL Syntax

The CQL syntax used should follow the OGC CQL format. Examples and a description of the grammar is located at http://docs.geoserver.org/stable/en/user/tutorials/cql/cql_tutorial.html.

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
```

Ingesting Data

Overview

Ingesting is the process of getting metadata into the Catalog Framework (including via the Content 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.

On This Page

- Overview
- File types supported
- Methods of Ingest
 - Easy (for fewer records or manual ingest)
 - Medium
 - Advanced (more records, automated ingest)

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 use file formats, such as Microsoft office products: Word documents, Excel spreadsheets, and PowerPoint presentations as well as .pdf files, GeoJson and others.

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 Content 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](#).

Usage

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.

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 ingest data into the catalog. For further information, see [Integrating Endpoints](#).

Appendix

Overview

Supplemental information for DDF appendix.

Table of Contents

- [DDF Directory Contents after Installation — Major Directories](#)
- [Software Versioning](#)
- [Troubleshooting — This section is a compilation of successful troubleshooting steps to take while installing or maintaining ApplicationName.](#)

DDF Directory Contents after Installation

Major Directories

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

Directory Name	Description
bin	Scripts to start and stop DDF
data	The working directory of the system – installed bundles and their data
data/log/ddf.log	Log file for DDF, logging all errors, warnings, and (optionally) debug statements. This log rolls up to 10 times, frequency based on a configurable setting (default=1 MB)
deploy	Hot-deploy directory – KARs and bundles added to this directory will be hot-deployed (Empty upon DDF installation)
docs	The DDF Catalog API Javadoc
etc	Directory monitored for addition/modification/deletion of third party <code>.cfg</code> configuration files
etc/ddf	Directory monitored for addition/modification/deletion of DDF-related <code>.cfg</code> configuration files (e.g., Schematron configuration file)
etc/templates	Template <code>.cfg</code> files for use in configuring DDF sources, settings, etc., by copying to the <code>etc/ddf</code> directory.

lib	The system's bootstrap libraries. Includes the ddf-branding.jar file which is used to brand the system console with the DDF logo.
licenses	Licensing information related to the system
system	Local bundle repository. Contains all of the JARs required by DDF, including third-party JARs.

Software Versioning

On This Page

- Format
- Major
- Minor
- Patch
- Build
- Thirdparty Versioning

Format

DDF component versions take the form *major.minor[.patch[.build]]*

Major

A major release occurs when new functionality is added to DDF that requires external clients to modify their implementations that interface with DDF.

Minor

A minor release occurs when new functionality is added to DDF that may require DDF developers to modify their implementations, but does not require external clients to modify their implementations that interface with DDF.

Patch

A patch release occurs when bugs are fixed in DDF but do not result in external clients or developers having to change their implementations that interface to DDF.

Build

The build section indicates milestone or maturity (e.g., RC, SNAPSHOT).

RC = Release Candidate

Thirdparty Versioning

It may be necessary to modify jars to make them bundles that will run in the OSGi container. These third party jars have the version of the JAR that it is wrapping followed by a suffix of "_X" where X is the version of that pom. For example, if the artifact is wrapping the JTS jar of version 1.11, the version of that pom creating the bundle would be 1.11_1. If that specific pom is ever changed, the suffix should be incremented. For example, a subsequent change would make the version number 1.11_2 then 1.11_3 on the next change.

Troubleshooting

Overview

This section is a compilation of successful troubleshooting steps to take while installing or maintaining DDF.

Table of Contents

- Blank Web Console
- CXF BusException
- Distribution Will Not Start
- Exception Starting DDF
- DDF Is Unresponsive to Incoming Requests

Blank Web Console

Blank Web Console

Problem:

<http://localhost:8181/system/console> opens as a blank page.

Solution:

Restart DDF.

1. Shut down DDF from the console:

```
ddf@local>shutdown
```

2. Start DDF back up:

```
./ddf
```

3. Verify that all of the files were copied over correctly during the deploy bundles step.

CXF BusException

CXF BusException

Problem:

The following exception is thrown:

```
org.apache.cxf.BusException: No conduit initiator
```

Solution:

Restart DDF.

1. Shut down DDF:

```
ddf@local>shutdown
```

2. Start up DDF:

```
./ddf
```

Distribution Will Not Start

Distribution Will Not Start

Problem:

DDF will not start when calling the start script defined during installation.

Solution:

Complete the following procedure.

1. Verify that Java is correctly installed.

```
java -version
```

2. Should return something similar to:

```
java version "1.6.0_22" Java(TM) SE Runtime Environment (build  
1.6.0_22-b04) Java HotSpot(TM) Server VM (build 17.1-b03, mixed mode)
```

3. If running *nix, verify that bash is installed.

```
echo $SHELL
```

Should return:

```
/bin/bash
```

Exception Starting DDF

Exception Starting DDF

Problem:

An exception is thrown starting DDF on a Windows machine (x86).

If using an unsupported terminal, java.lang.NoClassDefFoundError: Could not initialize class org.fusesource.jansi.internal.Kernel32 is thrown.

Solution:

Install missing Windows libraries.

Some Windows platforms are missing libraries that are required by DDF. These libraries are provided by the Microsoft Visual C++ 2008 Redistributable Package x64 (<http://www.microsoft.com/en-us/download/details.aspx?id=15336>).

DDF Is Unresponsive to Incoming Requests

DDF Is Unresponsive to Incoming Requests

Problem:

DDF is unresponsive to incoming requests.

An example of the log file when this problem is encountered:

```
Feb 7, 2013 10:51:33 AM org.apache.karaf.main.SimpleFileLock lock
INFO: locking
Feb 7, 2013 10:51:33 AM org.apache.karaf.main.Main doLock
INFO: Waiting for the lock ...
Feb 7, 2013 10:51:33 AM org.apache.karaf.main.SimpleFileLock lock
INFO: locking
Feb 7, 2013 10:51:33 AM org.apache.karaf.main.Main doLock
INFO: Waiting for the lock ...
Feb 7, 2013 10:51:34 AM org.apache.karaf.main.SimpleFileLock lock
INFO: locking
Feb 7, 2013 10:51:34 AM org.apache.karaf.main.SimpleFileLock lock
INFO: locking
Feb 7, 2013 10:51:35 AM org.apache.karaf.main.SimpleFileLock lock
INFO: locking
Feb 7, 2013 10:51:35 AM org.apache.karaf.main.SimpleFileLock lock
INFO: locking
```

Symptoms

- Multiple java.exe processes running, indicating more than one DDF instance is running. This can be caused when another DDF is not shut down.

Solutions:

Perform one or all of the following recommended solutions, as necessary.

- Wait for proper shutdown of DDF prior to starting a new instance.
- Verify running java.exe are not DDF (e.g., kill/close if necessary).
- Utilize automated [start/stop scripts](#) to run DDF as a service.

DDF Integrator's Guide

Overview

This guide provides instructions for configuring, maintaining and operation components of the Distributed Data Framework.

Distributed Data Framework (DDF) is an agile and modular integration framework. It is primarily focused on data integration, enabling clients to insert, query and transform information from disparate data sources via the DDF Catalog. A Catalog API allows integrators to insert new capabilities at various stages throughout each operation. DDF is designed with the following architectural qualities to benefit integrators.

Contents

- [Quick Start](#) — quick reference to installation, catalog capabilities, using, and reporting steps.

- Using the CometD Endpoint for Asynchronous Communication

Quick Start

On This Page
<ul style="list-style-type: none"> • Overview • Prerequisites • Install DDF • Catalog Capabilities • Use of the Content Framework • Metrics Reporting

Overview

This quick tutorial will demonstrate:

- Installation
- Catalog Capabilities: Ingest and query using every endpoint
- Use of the Content Framework
- Metrics Reporting

Prerequisites

Review [Prerequisites](#) to ensure all system prerequisites are met.

Install DDF

1. Install DDF by unzipping the [zip file](#). 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 in place of this.
2. Start DDF by running the <DISTRIBUTION_INSTALL_DIR>/bin/ddf script (or ddf.bat on Windows).
3. Verify the distribution is running.
 - a. Go to <https://localhost:8993/admin>.
 - b. Enter the default username of "admin" (no quotes) and the password of "admin" (no quotes).
4. Follow the [install instructions](#) for more extensive install guidance, or use the command line console (which appears after the < DISTRIBUTION_INSTALL_DIR>/bin/ddf script starts) to install a few applications as mentioned below.

```
app:start catalog-app
app:start content-app
app:start solr-app
```

Other applications may be installed at a later time.

5. Go to <http://localhost:8181/services> and verify five REST services are available: admin, application, metrics, catalog, and catalog/query.
6. Click on the links to each REST service's WADL to see its interface.
7. In the Web Console (at </system/console/configMgr>), configure the system settings.
 - a. Enter the username of "admin" (no quotes) and the password "admin" (no quotes).
 - b. Select **Platform Global Configuration**.
 - c. Enter the port and host where the distribution is running.

Catalog Capabilities

1. Create an entry in the Catalog by ingesting a [valid GeoJson file](#) (attached to this page). This ingest can be performed using:
 - a. A REST client, such as Google Chrome's Advanced REST Client.

- b. OR by using the following curl command to POST to the Catalog REST CRUD endpoint.

Windows Example

```
curl.exe -H "Content-type: application/json;id=geojson" -i -X  
POST -d @"C:\path\to\geojson_valid.json"  
http://localhost:8181/services/catalog
```

*NIX Example

```
curl -H "Content-type: application/json;id=geojson" -i -X POST -d  
@geojson_valid.json http://localhost:8181/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 2013 22:02:22 GMT  
id: 44dc84da101c4f9d9f751e38d9c4d97b  
Location:  
http://localhost:8181/services/catalog/44dc84da101c4f9d9f751e38d  
9c4d97b  
Server: Jetty(7.5.4.v20111024)
```

2. 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.
3. Verify the catalog entry exists by executing a query via the OpenSearch endpoint.
4. Enter the following URL in a browser `/services/catalog/query?q=ddf`. A single result, in Atom format, should be returned.

Use of the Content Framework

Using the Content 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 [Web Console](#), select the Configuration tab.
2. Select the **Content Directory Monitor**.
3. Set the directory path to **inbox**.
4. Click the **Save** button.
5. Copy the attached `geojson_valid.json` file to the `<INSTALL_DIR>/inbox` directory.

The Content 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 that XML metadata for text searching is not automatically generated from GeoJson fields.

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

- a. Install the feature content-rest-endpoint using the [Features tab](#) in the Web Console.
- b. 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 <DISTRIBUTION_INSTALL_DIR>/content/store directory and copying the sub-directory in this folder (there should only be one).

*NIX Example

```
curl -X GET
http://localhost:8181/services/content/c90147bf86294d46a9d35ebbd44992c5
```

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

Metrics Reporting

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

1. Open the Web Console ([/system/console/metrics](#)).
2. Select the **PNG** link for **Catalog Queries** under the column labeled **1h** (one hour). A graph of the catalog queries that were performed in the last hour is displayed.
3. Select the browser's back button to return to the Metrics tab.
4. Select the **XLS** link for **Catalog Queries** under the column labeled **1d** (one day).

Handy Tip

Based on the browser's configuration, the .xls file will be downloaded or automatically displayed in Excel.

Using the CometD Endpoint for Asynchronous Communication

On This Page

- CometD Basics
 - General Operations
 - Initialization
 - Subscribe
 - Publish
 - DDF Specifics
 - Asynchronous Queries
 - Asynchronous Notifications and Activities

CometD Basics

DDF uses the CometD framework (<http://cometd.org/>) 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 (<http://docs.cometd.org/2/reference/>) that details how to interact with and best use the framework with a variety of clients (e.g., javascript, java, perl, and python). This page is used to help developers understand how the UI application uses CometD, so the developer can build off of this same interface with their own implementations.

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 (<http://docs.cometd.org/2/reference/javascript.html>) 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 DDF 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);
```

Further documentation is available at http://docs.cometd.org/2/reference/javascript_subscribe.html.

Publish

Publishing allows clients to send information to the server. This allows the UI 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: { ... } });
```

DDF Specifics

The CometD endpoint is still undergoing changes, and the following pages describing the interfaces may change during development. Although breakage changes are not anticipated, it is recommended to watch these pages when developing components that use the CometD endpoint, so any changes made will be immediately known.

DDF has defined formats for both input and output data that are specific to the operations that are being performed. Information on these data formats and the corresponding comet topics are available on the following pages:

Asynchronous Queries

Asynchronous Notifications and Activities

Asynchronous Notifications and Activities

On This Page

- Notification Events
 - Channel
 - Notification Format
- Notification Operations
 - Channel
 - Operation Format
- Activity Events
 - Channel
 - Activity Format
- Activity Operations
 - Channel
 - Activity Format
- Retrieving Persisted Notifications and Activities

Notification Events

Channel

To receive all notifications, follow the instructions on the [Using the CometD Endpoint for Asynchronous Communication](#) page and subscribe to /ddf/notifications/**

Notification Format

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

Map Key	Description	Value Type	Possible Values	Example Values
application	Name of the application that caused the notification to be sent	String	Any	"Downloads"
id	ID of the notification "thread" – Notifications about the same event should use the same id to allow clients to filter out notifications that may be outdated.	String	Any	"27ec3222af1144ff827a351b1962a236"

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

Example: Notification Message

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

Notification Operations

Channel

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

Operation Format

Map Key	Description	Value Type	Possible Values	Example Values	Comments
action	Type of action to request	String	Any	"remove"	
id	ID of the notification to which the action relates	String	Any	"27ec3222af1144ff827a351b1962a236 6"	This is the id of the notification, nothing else

Example: Notification Operation Request

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

Activity Events

Channel

To receive all activity updates, follow the instructions on the [Using the CometD Endpoint for Asynchronous Communication](#) 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
category	Category of the activity	Any String	Any String	"Product Retrieval"
event.topics		String		
id	ID that uniquely identifies the activity that sent out the update. Not required to be unique per update.	String	Any String	"b72ccdf6-8ca7-4f53-a0f6-b0ad264393b0"
message	User-readable message that explains the current activity status	String	Any String	"Resource retrieval downloading."
operations	Map of operations that can be performed on this activity	JSON Map	<p>A map of keys with populated values (that evaluate to 'true' rather than 'null' 'undefined' or 'false')</p> <p>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> <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"> • "download" • "retry" • "cancel" • "pause" • "remove" • "resume" 	<pre> "operations" : { "download" : "http://example.com/product" } </pre>
progress	Percentage value of activity completion	String	Integer between 0 - 100 followed by a %	"45%"
status	Enumerated value that displays the current state of the activity	String	"STARTED", "RUNNING", "FINISHED", "STOPPED", "PAUSED", or "FAILED"	"RUNNING"
timestamp	Time that the activity update was sent	String	Positive long value (seconds since unix epoch)	1403801920875
title	User-readable title for the activity update	String	Any String	"Download Complete"
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  
}
```

Activity Operations

Channel

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

Refer to [Using the CometD Endpoint for Asynchronous Communication](#) for instructions on how to publish to a CometD channel.

Activity Format

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

Example: Activity Operation Request Message

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

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 [Using the CometD Endpoint for Asynchronous Communication](#) for instructions on how to publish to a CometD channel.

Example: Persistence Retrieval

```
Cometd.Comet.publish("/ddf/notifications", {});  
Cometd.Comet.publish("/ddf/activities", {});
```

Asynchronous Queries

On This Page

- Query Request
 - Channel
 - Request Format
 - Examples
- Query Response
 - Channel
 - Response Format
 - Examples

Asynchronous Query

An asynchronous query is performed using the [CometD Endpoint](#), which is currently packaged with the DDF UI application.

Query Request

Channel

All query requests should be published to the `/service/query` channel following the guidance on the [Using the CometD Endpoint for Asynchronous Communication](#) page.

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 Values	Example Search UI Value	Comments
count	Number of entries to return in the response	Number	Positive integer	250	
format	Format that the results should be displayed in	String	"geojson"	"geojson"	"geojson" is the recommended format to use
id		String		"4303ba5d-21af-4878-9a4c-808e80052e6c"	
src	Comma-delimited list of federated sites to search over	String	Any list of site names.	"DDF-OS,ddf.distribution"	
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	Search Filter	String	OGC CQL formatted string	"anyText LIKE '*'"	See OpenGIS® Catalogue Services Specification for more details

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  
}
```

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 [Using the CometD Endpoint for Asynchronous Communication](#) page details how to subscribe to a CometD channel.

Response 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	Comments
id	ID that corresponds to the request	String	ID value	

hits	Total number of query hits that were found by the server	Number	Integer >= 0	This contains the total amount of items that were found by the query. Depending on the 'count' in the request, not all of the results may be returned.
results	Array of metocard results	Array of Maps	GeoJson formatted value	This format is defined by the GeoJSON Metacard Transformer .
results/metocard/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	Number of results that were returned in this response from the source	Number	Integer >= 0	
types	A Map mapping a metocard-type's name to a map about that metocard-type. Only metocard-types represented by the metacards returned in the query are represented.	Map of Maps		The Map defining a particular metocard-type maps the the fields supported by that metocard-type to the datatype for that particular field.

Examples

Example Query Response

```

"data": {
    "id": "4303ba5d-21af-4878-9a4c-808e80052e6c",
    "hits": 20,
    "results": [
        {
            "metocard": {
                "geometry": {
                    "coordinates": [
                        174.77557,
                        -41.28664
                    ],
                    "type": "Point"
                },
                "properties": {
                    "created": "2014-07-02T18:53:59.496+0000",
                    "id": "126e16d340dd47b7ad823b70662ed8ca",
                    "metocard-type": "ddf.metocard",
                    "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.metocard.resource",
            "title": "Get resource",
            "description": "Gets the Metacard
resource",
            "url":
"http://ddf.codice.org:8181/services/catalog/sources/ddf.distribution/8522
7c45d9c34fe1ad3e725a72d6b44a?transform=resource"
        },
        {
            "id": "catalog.data.metocard.html",
            "title": "Get html",
            "description": "Gets the Metacard html",
            "url":
"http://ddf.codice.org:8181/services/catalog/sources/ddf.distribution/8522
7c45d9c34fe1ad3e725a72d6b44a?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.metocard" : {
        "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",
        "resource-type" : "STRING"
    }
}
```

```
    "metadata" : "XML"
},
...other metocard types...
```

```
    }  
}  
}
```

DDF Developer's Guide

Overview

This guide discusses the several extension points and components permitted by the Distributed Data Framework (DDF) Catalog API. Using code examples, diagrams, and references to specific instances of the Catalog API, this guide provides details on how to develop and integrate various DDF components.

Contents

- [Building DDF](#)
- [DDF Development Prerequisites](#)
- [Formatting Source Code](#)
- [Ensuring Compatibility](#)
- [Development Recommendations](#) — provides general developer tips and recommendations for OSGi bundle development
- [UI Development Recommendations](#)
- ["White Box" DDF Architecture](#) — This page describes ApplicationName at the infrastructure level, including where ApplicationName fits in an architectural stack.
- [Developing DDF Applications](#)
- [OGC Filter with DDF Frequently Asked Questions](#) — This section answers frequently asked questions about the use of the OGC Filter in DDF Catalog Queries and Subscriptions.
- [Utilities](#)

Building DDF

On This Page

- [Prerequisites](#)
- [Procedures](#)
 - [Run the Build](#)
 - [Troubleshoot Build Errors on ddf-admin and ddf-ui on a Windows Platform](#)

Prerequisites

- Install J2SE 7 SDK (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>).
- 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 Maven 3.0.4 or later (<http://maven.apache.org/download.cgi>). 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 (<http://maven.apache.org/settings.html>).

Sample settings.xml file

```
<settings>
  <!-- If proxy is needed
  <proxies>
    <proxy>
    </proxy>
  </proxies>
  -->
</settings>
```

Handy Tip on Encrypting Passwords

See this Maven [guide](#) on how to encrypt the passwords in your settings.xml.

Procedures

Run the Build

In order to run through a full build, be sure to have a clone for all repositories in the same folder:

- ddf (<https://github.com/codice/ddf.git>)
- ddf-admin (<https://github.com/codice/ddf-admin.git>)
- ddf-catalog (<https://github.com/codice/ddf-catalog.git>)
- ddf-content (<https://github.com/codice/ddf-content.git>)
- ddf-parent (<https://github.com/codice/ddf-parent.git>)
- ddf-platform (<https://github.com/codice/ddf-platform.git>)
- ddf-security (<https://github.com/codice/ddf-security.git>)
- ddf-solr (<https://github.com/codice/ddf-solr.git>)
- ddf-spatial (<https://github.com/codice/ddf-spatial.git>)
- ddf-support (<https://github.com/codice/ddf-support.git>)
- ddf-ui (<https://github.com/codice/ddf-ui.git>)

- 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.
```

- Build command example to build all repositories. This must be performed at the top level folder that contains all the repositories. A command list would look like this:

```
# Build is run from the top level folder that contains all the
repositories in a command line prompt or terminal.

cd ddf-support
mvn clean install

cd ../ddf-parent
mvn clean install

cd ../ddf-platform
mvn clean install

cd ../ddf-admin
mvn clean install

cd ../ddf-security
mvn clean install

cd ../ddf-catalog
mvn clean install

cd ../ddf-content
mvn clean install

cd ../ddf-spatial
mvn clean install

cd ../ddf-solr
mvn clean install

cd ../ddf-ui
mvn clean install

cd ../ddf
mvn clean install

# This will fully compile each individual app. From here you may hot
deploy the necessary apps on top of the DDF Kernel.
```

To use the updated apps in a DDF distribution, update the versions referenced in the "ddf" repository.

The zip distribution of DDF is contained in the DDF app in the distribution/ddf/target directory after the DDF app is built.

- Optionally, create a reactor pom that will allow you to perform the entire build process by calling a build on one pom rather than all of them. This pom must reside in the top-level folder that holds all the repositories. An example of the file would be:

```

<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">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.codice.ddf</groupId>
  <artifactId>reactor</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>DDF Reactor</name>
  <description>Distributed Data Framework (DDF) is an open source,
  modular integration framework</description>

  <modules>
    <module>ddf-support</module>
    <module>ddf-parent</module>
    <module>ddf-platform</module>
      <module>ddf-admin</module>
    <module>ddf-security</module>
    <module>ddf-catalog</module>
    <module>ddf-content</module>
    <module>ddf-spatial</module>
    <module>ddf-solr</module>
    <module>ddf-ui</module>
    <module>ddf</module>
  </modules>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.4</version>
        <configuration>
          <!-- Do not deploy the reactor pom -->
          <skip>true</skip>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>

```

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.

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 -Xmx512m -XX:MaxPermSize=256m before running the build.

Example on Linux system with the bash shell: export MAVEN_OPTS='-Xmx512m -XX:MaxPermSize=256m'

Troubleshoot 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.

```
bower cache clean  
npm cache clean
```

3. Reinstall bower. An occasional reinstall may solve the issue.

```
npm uninstall -g bower && npm install -g bower
```

4. 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.

DDF Development Prerequisites

Overview

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 (<http://www.osgi.org>). 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. For specific code examples from DDF, source code can be seen in the [OSGi Services](#) section.

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 (<http://books.sonatype.com/m2eclipse-book/reference/>) on developing with Eclipse.

Additional Documentation

Additional documentation on developing with the core technologies used by DDF can be found on their respective websites. Notably:

- Karaf (<http://karaf.apache.org/manual/latest-2.2.x/developers-guide/index.html>)
- CXF (<http://cxf.apache.org/docs/overview.html>)
- Geotools (<http://docs.geotools.org/latest/developer/>)

Major Directories

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

Directory Name	Description
bin	Scripts to start and stop DDF
data	The working directory of the system – installed bundles and their data
data/log/ddf.log	Log file for DDF, logging all errors, warnings, and (optionally) debug statements. This log rolls up to 10 times, frequency based on a configurable setting (default=1 MB)
deploy	Hot-deploy directory – KARs and bundles added to this directory will be hot-deployed (Empty upon DDF installation)
docs	The DDF Catalog API Javadoc
etc	Directory monitored for addition/modification/deletion of third party .cfg configuration files
etc/ddf	Directory monitored for addition/modification/deletion of DDF-related .cfg configuration files (e.g., Schematron configuration file)
etc/templates	Template .cfg files for use in configuring DDF sources, settings, etc., by copying to the etc/ddf directory.
lib	The system's bootstrap libraries. Includes the ddf-branding.jar file which is used to brand the system console with the DDF logo.
licenses	Licensing information related to the system
system	Local bundle repository. Contains all of the JARs required by DDF, including third-party JARs.

Formatting Source Code

On This Page

- Load the Code Formatter Into the Eclipse IDE
- Load the Code Formatter Into IntelliJ IDEA
- Format Your Source Code Using Eclipse
- Set Up Save Actions in Eclipse
- Format Source Code Using IntelliJ

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.

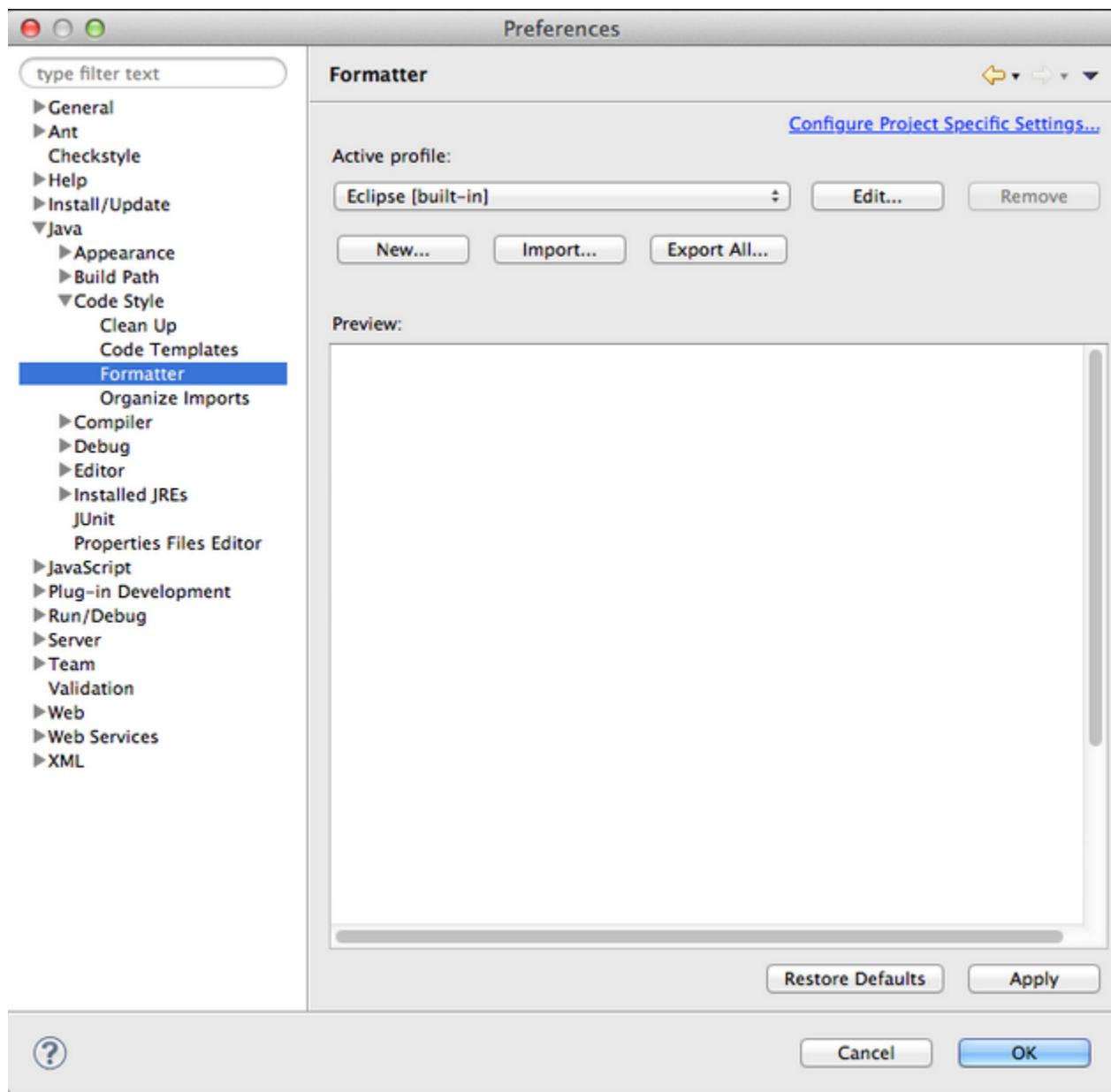
DDF uses an updated version of the Apache ServiceMix Code Formatter (<http://servicemix.apache.org/developers/building.html>) for code formatting.

DOWNLOAD THIS FILE: [ddf-eclipse-code-formatter.xml](#)

1. Follow the link.
2. Right-click on **Raw**.
3. Select **Save As**.

Load the Code Formatter Into the Eclipse IDE

1. In Eclipse, select **Window Preferences**. The Preferences window opens.
2. Select **Java Code Style Formatter**.
3. Select the **Edit...** button and load the attached `ddf-eclipse-code-formatter.xml` file.
4. 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. Open **IntelliJ IDEA**.
2. Select **File Settings Code Style Java**.
3. Select **Manage**.
4. Select the **Import** button to Import `ddf-eclipse-code-formatter.xml` file.

Format Your Source Code Using Eclipse

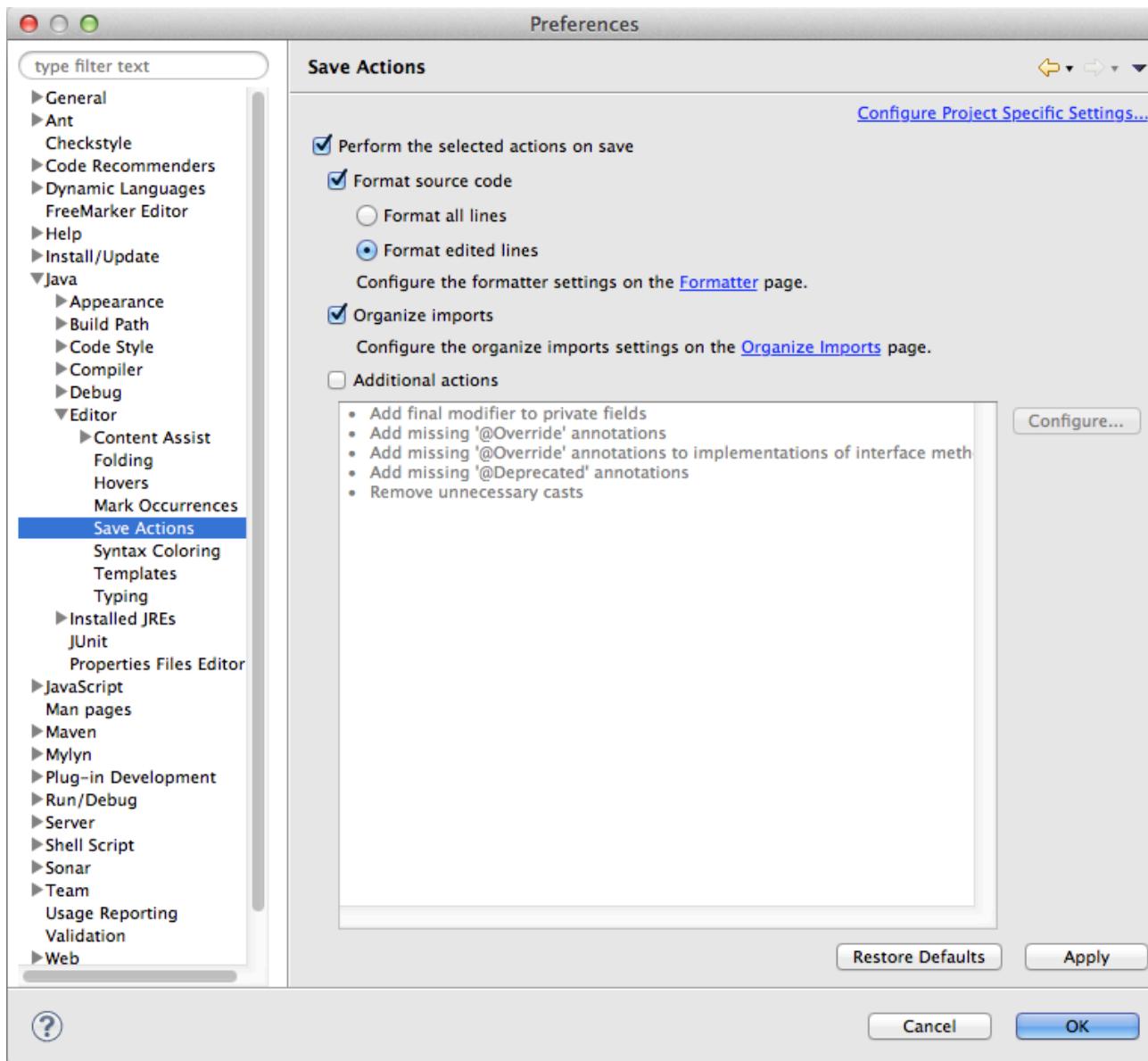
A developer may write code and format it before saving.

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**.

Ensuring Compatibility

On This Page

- Compatibility Goals
- Guidelines for Maintaining Compatibility
 - DDF Framework
 - DDF Catalog API
- DDF Software Versioning
- Third Party and Utility Bundles
- Best Practices

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 FileInstall and ConfigurationAdmin and Web Console (for Metatype support)

Avoid developing dependencies on the implementations directly, as compatibility in future releases is not guaranteed.

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.

DDF Software Versioning

DDF follows the [Semantic Versioning](#) White Paper for bundle versioning (see [Software Versioning](#)).

Third Party and Utility Bundles

It is recommended to avoid building directly on included third party and utility bundles. These components do provide utility (e.g., JScience) 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 like commons-geospatial, 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).

Best Practices

- Always use a version number when exporting a package. In the following example, the \${project.artifactId} represents the project and artifactId of the package being exported. The \${project.version} represents the version of the project.

Export Example

```
<Export-Package>
    ${project.artifactId}*;version=${project.version}
</Export-Package>
```

- Try to avoid deploying multiple versions of a bundle. Although OSGi is designed to support multiple versions, other developers may not include the versions of the packages that are being imported. If the bundle is versioned and designed appropriately, typically, having

multiple versions of the bundle will not be an issue. However, if each bundle is competing for a specific resource, race conditions may occur. Third party and utility bundles (often denoted by commons in the bundle name) are the general exception to this rule, as these bundles will likely function as expected with multiple versions deployed.

Development Recommendations

On This Page

- Javascript
- Package Names
- Author Tags
- Unit Testing
- Logging
- Dependency Injection Frameworks
- Basic Security

OSGi Development Recommendations Table of Contents

- Fat Bundles
- OSGi Services — provides quick tutorials of OSGi basic concepts

Javascript

Avoid using Console.log

Package Names

Use singular package names.

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.

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

The recommended framework to use with DDF is Mockito: <https://github.com/mockito/mockito>. This test-level dependency is managed by the ddf-parent pom and is used to standardize the version being used across DDF.

Logging

There are many logging frameworks available for Java.

Recommended Framework

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);
```

Dependency Injection Frameworks

It is highly recommended to use a dependency injection framework, such as Blueprint, Spring-DM, or iPojo for non-advanced OSGi tasks. Dependency injection frameworks allow for more modularity in code, keep the code's business logic clean of OSGi implementation details, and take the complexity out of the dynamic nature of OSGi. In OSGi, services can be added and removed at any time, and dependency injection frameworks are better suited to handle these types of situations. Allowing the code to be clean of OSGi packages also makes code easier to reuse outside of OSGi. These frameworks provide code conveniences of service registration, service tracking, configuration property management, and other OSGi core principles.

Basic Security

(Provided by Pierre Parrend (<http://www.slideshare.net/kaihackbarth/security-in-osgi-applications-robust-osgi-platforms-secure-bundles>))

- Bundles should
 - Never use **synchronized** statements that rely on third party code. Keep multi-threaded code in mind when using synchronized statements in general, as they can lead to performance issues.
 - Only have dependencies on bundles that are trusted.
- Shared Code
 - Provide only final static non-mutable fields.
 - Set security manager calls during creation in all required places at the beginning of methods.
 - All Constructors
 - *clone()* method if a class implements *Cloneable*
 - *readObject(ObjectInputStream)* if the class implements *Serializable*
 - Have security check in **final** methods only.
- Shared Objects (OSGi services)
 - Only have basic types and **serializable final** types as parameters.
 - Perform copy and validation (e.g., null checks) of parameters prior to using them.
 - Do not use **Exception** objects that carry any configuration information.

Fat Bundles

- Decision
 - Status [NOT VOTED, APPROVED, DENIED, TABLED]
 - Date of Decision
 - Contributions by
- Description
 - Decision
 - Why use a fat bundle?
 - Whitelist
 - When to use a fat bundle?
 - Migration
- Pros and Cons
 - Pros
 - Cons or Risks
- Unresolved

Decision

Status [NOT VOTED, APPROVED, DENIED, TABLED]

Approved

Date of Decision

Nov 13, 2013

Contributions by

- Ashraf Barakat (editor)
- Bruce Beyeler
- Phillip Klinefelter
- Michael Menousek

Description

Decision

Decision is to adopt the best practice of creating applications as fat jars with plans to move to OSGi subsystems when the standard and implementation is mature.

What is a fat bundle?

A fat bundle, defined for the purposes of our community, is a single OSGi bundle jar which contains both the business logic of a new application as well as the dependencies of that application. It should contain all files and resources the application needs with the exception of shared-use interfaces and system files (found on the whitelist). It is not necessarily executable on its own. It does not export any packages except for necessary shared API packages if it is known that others will use them across apps.

Why use a fat bundle?

A proper fat bundle allows for application isolation and alleviates problems with exposing implementation details and compatibility issues between applications on the OSGi container. See Pros and Cons of using fat bundles.

Whitelist

DDF shall keep a whitelist of packages that are allowed to be used by other apps. The DDF whitelist will be eventually phased out once corrections are made to the existing DDF exported packages. Each DDF app should be evaluated for what packages are intentionally exposed to the container, therefore providing their own whitelist. For third party developers, ideally a newly developed application's whitelist should be equivalent to the exported packages of the new application's bundle(s) if designed properly.

Preliminarily, packages developers can rely on from the container include DDF APIs, packages from the system bundle (bundle 0) such as JRE packages, OSGi components, logging (slf4j, log4j, etc) support, and certain xml apis that were not written for OSGi. Other items will be identified in a later whitelist.

When to use a fat bundle?

A fat bundle should be used when creating new applications to be deployed in the OSGi container. Fat bundles are recommended to only import packages that are on the DDF whitelist or the whitelist of other applications. All external (non-DDF) developers shall use fat bundles until OSGi subsystems have been adopted.

Migration

DDF bundles and applications shall be migrated to fat bundles as time and resources permit and when applicable. DDF applications shall be migrated to fat bundles at the feature level as to allow for easy configuration of the DDF system.

As for DDF implementation packages exported to the container and on which others have previously depended, those classes in those packages will be deprecated. The packages will continue to be exported for backwards compatibility, but since they are deprecated, they can be removed at the DDF team's discretion in later versions. The deprecated classes in those packages will not be maintained. Instead, those classes will be moved to new packages and should be embedded within application's fat bundles. Those who want to use DDF implementation libraries do so at their own risk. The recommendation is to continue to code to interfaces and not implementations.

Pros and Cons

Pros

Degree (S,M,L)	Description
L	Since exports are prohibited and dependencies are embedded, a fat bundle hides the internals of the application, which should reduce complexity to container and clients.
L	Hiding details means freedom to change implementation with more ease.
L	Upgrades to other apps are less likely to affect a self-contained jar
L	Installation and upgrades of new app is less likely to affect others on container
L	Does not require developers to "bundlify" or make dependencies "OSGi-ready" depending on the application.
M	Provides more control to the developer over their dependencies. Developers don't have to worry what versions of their dependencies will have to work in the container since they bring their own dependencies.
S	Fat bundles have similarities to OSGi subsystems in that code can run in isolation. Since the concepts of isolation are similar, migration from fat bundles to subsystems should not cause great difficulty.
S	Since it is self-contained, the mechanism of an upgrade would only include replacing a single jar in most cases. No need to run scripts or identify other dependencies during installation, which should make installation/upgrades easier.
S	Encourages better interface design since most details should be hidden from developers.

Cons or Risks

Degree (S,M,L)	Description
L	(Risk) Requires major rework to existing DDF apps.
L	Cannot disable/enable certain portions of the business logic as easily as if those pieces were in separate bundles. If they were in bundles, you could use features or manually just remove the bundles.
M	Duplication/Redundancy of jars across apps. Loss of benefits of OSGi classloader and package sharing / bundle reuse. Less sharing of code across the container could mean more classloading done by container. Possible increase in memory usage. (Metrics needed for this argument.)
M	Bundles become more complex because all the dependencies are within the jar. Previously bundles mostly contained the necessary business logic packages.
M	Sharing of bundles allows for automatic bug fixes / new functionality across apps. Requires developers to upgrade their own app dependencies and keep track of their own transitive dependencies.
S	Doesn't use Versioning of OSGi for implementation bundles.

Unresolved

OSGi Services

On This Page

- OSGi Service Registry
 - Spring DM - Retrieving a Service Instance
 - Blueprint - Retrieving a Service Instance
 - Blueprint - Registering a Service into the Registry
- Packaging Capabilities as Bundles
 - Creating a Bundle
 - Bundle Development Recommendations
 - Maven Bundle Plugin
 - Deploying a Bundle
 - Verifying Bundle State
- Additional Resources

OSGi Service Registry

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. There are many tutorials and guides available on the Internet for both of these frameworks. Refer to the [Additional Resources](#) section for details not covered in this guide. Links to some of these guides are given in the External Links section of this page.

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

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
6	Instantiates a new object, injecting the retrieved Service as a constructor argument

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

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.

Alternative Bundle Creation Methods

Using Maven is not necessary to create bundles. Alternative tools exist, and OSGi manifest files can also be created by hand although hand editing should be avoided by most developers.

See external links (below) for resources that give in-depth guides on creating bundles.

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.

Maven Bundle Plugin

Below is a code snippet from a Maven `pom.xml` for creating an OSGi Bundle using the Maven Bundle plugin.

Maven pom.xml

```
...
<packaging>bundle</packaging>
...
<build>
...
<plugin>
    <groupId>org.apache.felix</groupId>
    <artifactId>maven-bundle-plugin</artifactId>
    <configuration>
        <instructions>
            <Bundle-Name>${project.name}</Bundle-Name>
            <Export-Package />

            <Bundle-SymbolicName>${project.groupId}.${project.artifactId}</Bundle-SymbolicName>
            <Import-Package>
                ddf.catalog,
                ddf.catalog.*
            </Import-Package>
        </instructions>
    </configuration>
</plugin>
...
</build>
...
```

Deploying a Bundle

A bundle is typically installed in two ways:

- As a feature
- 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 the CAB Client is deployed and running.

Verifying with grep

```
ddf@local>list | grep -i cab
[ 162] [Active     ] [          ] [    ] [ 80] DDF :: Registry :: CAB Client
(2.0.0)
```

The state is `Active`, indicating that the bundle is ready for program execution.

Additional 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>

UI Development Recommendations

On This Page

- Overview

Overview

Recommendations for developing UI components.

"White Box" DDF Architecture

On This Page

- Architecture Diagram
- Nomenclature
- OSGi Core
- Built on Apache Karaf
- Additional Upstream Dependencies
- Recommended Hardware

Architecture Diagram

 Unknown macro: 'plantuml'

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.

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.

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/>.

Built on Apache Karaf

Apache Karaf is a FOSS product that includes an OSGi framework and adds extra functionality, including:

- **Web Administration Console** - Useful for configuring bundles, installing/uninstalling features, and viewing services.
- **System Console** - Provides command line administration of the OSGi container. All functionality in the Web Administration Console can also be performed via this command line console.
- **Logging** - Provides centralized logging to a single log file (data/logs/ddf.log) utilizing log4j.
- **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 <http://aries.apache.org/modules/blueprint.htm>.
- **Spring DM** - An alternative dependency injection framework. DDF is not dependent on specific dependency injection framework. Blueprint is recommended.

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).

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.

Developing DDF Applications

On This Page

- Overview
- Creating a KAR File
- Including Data Files in a KAR File
- Installing a KAR File
- Installing a KAR File for DDF 2.0.0 Release

Overview

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 Web Console or command line 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 (Karaf ARchive). 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.

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 (available as of Karaf v2.2.5, which DDF 2.X is based upon). This goal reads all of the features specified in the features 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.

An example of using the `create-kar` goal is shown below:

create-kar goal

```
<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
-->
                <!--
                    Location of the features.xml file. If it references
properties that need to be filtered, e.g., ${project.version}, it will need
to be
                    filtered by the maven-resources-plugin.
-->
            <featuresFile>${basedir}/target/classes/features.xml</featuresFile>
                <!-- Name of the kar file (.kar extension added by
default). If not specified, defaults to ${project.build.finalName} -->
                <finalName>ddf-ifis-${project.version}</finalName>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Examples of how KAR files are created for DDF components can be found in the DDF source code under the `ddf/distribution/ddf-kars` directory.

The .kar file generated should be deployed to the application author's maven repository. The URL to the application's KAR file in this Maven repository should be the installation URL that is used.

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.

Installing a KAR File

When the user downloads an application by clicking on the **Installation** link, the application's KAR file is downloaded. This KAR file should 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. The user can then go to the Web Console's Features tab and verify the new feature(s) is installed.

Installing a KAR File for DDF 2.0.0 Release

DDF 2.0.0 is based on Karaf v2.2.2, which has a bug in the KAR deployer. Therefore, a more manual approach to install a KAR file is required rather than simply dropping the KAR file in the `<DDF_INSTALL_DIR>/deploy` directory.

DDF 2.2.0+ should have this issue resolved by upgrading Karaf to v2.3.x+. Refer to

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration. (DDF-46)and KARAF-771 (<https://issues.apache.org/jira/browse/KARAF-771>) for more details.

After downloading the application's KAR file:

1. Copy the KAR file to a temp directory.
2. Unjar the KAR file in this temp directory.
3. Edit the `etc/org.ops4j.pax.url.mvn.cfg` configuration file.
4. Uncomment the `org.ops4j.pax.url.mvn.repositories` property if it is commented out.
5. Append the `org.ops4j.pax.url.mvn.repositories` property with `file:<temp_dir>/<kar_root_dir>/repository`.
6. Restart DDF.
7. Add the KAR's feature repository URL using either the command line console or the Features tab in the Web Console.

Example:

Assume the application's KAR file is named `new-app-1.2.3.kar`

1. Copy `new-app-1.2.3.kar` to `/my_kars` temp directory.
2. `unjar xvf /my_kars/new-app-1.2.3.kar`
3. Edit the `<DDF_INSTALL_DIR>/etc/org.ops4j.pax.url.mvn.cfg` configuration file.
4. Append `file:/my_kars/new-app-1.2.3/repository` to the `org.ops4j.pax.url.mvn.repositories` property.
5. Restart DDF.
6. At the `ddf@local>` prompt enter `features:addurl file:/my_kars/new-app-1.2.3/repository/new-app/1.2.3/new-app-1.2.3-features.xml`
7. At the `ddf@local>` prompt enter `features:list`. The new feature(s) loaded by the KAR file should be displayed and active.

OGC Filter with DDF Frequently Asked Questions

On This Page

- Frequently Asked Questions
 - What Is an OGC Filter?
 - Why Does DDF Need an OGC Filter?
 - If the Standard Is Implemented Using XML, How Will It Be Used in the DDF Catalog, Which Is Mostly Implemented in Java?

Frequently Asked Questions

- What Is an OGC Filter?
- Why Does DDF Need an OGC Filter?
- If the Standard Is Implemented Using XML, How Will It Be Used in the DDF Catalog, Which Is Mostly Implemented in Java?

What Is an 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).

Why Does DDF Need an OGC Filter?

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.

If the Standard Is Implemented Using XML, How Will It Be Used in the DDF Catalog, Which Is Mostly Implemented in Java?

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.

Utilities

Each [DDF Application](#) is located in its own code repository. In addition to the applications, there are other utilities that are available in other code repositories on Github. These utilities are deployed into Nexus for easier accessibility.

- [DDF-libs](#) — DDF-libs is a repository for library modules in DDF. Typically the modules in this repository are re-usable across different components of DDF.
- [DDF Load Balancer](#) — provides utility that allows incoming traffic to be distributed over multiple instances of DDF, via HTTP or HTTPS
- [DDF STOMP](#) — The DDF STOMP application allows query subscription messages to be sent to the DDF server via STOMP protocol.

DDF-libs

DDF-libs is a repository for library modules in DDF. Typically the modules in this repository are re-usable across different components of DDF.

DDF HTTP Proxy

On This Page

- Overview
- Set up the HTTP Proxy
- Run DDF HTTP Proxy as an OSGi Service
- Run DDF HTTP Proxy as an Embeddable Service
- Use the DDF HTTP Proxy
- Configure External Proxy for Target URL
- Configure Target URL for SSL

Overview

The DDF HTTP Proxy provides a reverse proxy mechanism to allow a DDF service to connect with external HTTP-based services and web pages via an internal URL. The proxy creates an endpoint on DDF, which is accessible via HTTP or HTTPS. This endpoint is configured to route requests from the endpoint to a specified URL. Multiple proxy connections can be created with this service. The proxy is based on Camel and utilizes the base web container to host the proxy.

Set up the HTTP Proxy

The DDF HTTP Proxy is included in the DDF distribution, but it can also be found as a standalone DDF app or KAR file at <https://github.com/codice/ddf-http-proxy>.

Run DDF HTTP Proxy as an OSGi Service

If installing the app by dropping the KAR file into the deploy directory, the HTTP Proxy will automatically start and can be used as an OSGi service. To achieve this, complete the following procedure.

1. Clone the ddf-http-proxy repository from git: <https://github.com/codice/ddf-http-proxy>.
2. Run maven install to build ddf-http-proxy.
3. Deploy KAR file built under /ddf-http-proxy/httpproxy-app/target.

If a standard distribution of DDF is being utilized, the proxy is contained within DDF as a feature but is not active. To activate the proxy, start DDF then type the following in the console:

```
features:install codice-httpproxy
```

This command should start the proxy.

Run DDF HTTP Proxy as an Embeddable Service

The proxy can also be installed as an embeddable service. By being embeddable, the proxy bundles can be placed in the classpath of an existing or new bundle and utilized as a service. The benefit of this is the elimination of dependencies on the deployed DDF HTTP Proxy bundle. To include these bundles, they can either be embedded using a reference to the bundles in the POM of the project or by building the repository and including the bundles in the bundle classpath directly. Complete the following procedure to build the bundles.

1. Clone the ddf-http-proxy repository from git: <https://github.com/codice/ddf-http-proxy>.
2. Run maven install to build ddf-http-proxy.
3. Copy the proxy camel servlet bundle located at ddf-http-proxy/proxy-camel-servlet/target to your classpath directory.
4. Copy the proxy camel route bundle located at ddf-http-proxy/proxy-camel-route/target to your classpath directory.

Once the bundle libraries are in place, the appropriate classes can be put into place for use. The following blueprint can be used as a guideline for setting up the appropriate classes:

```

<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="camelServletCreator"
        class="org.codice.proxy.http.CamelServletCreator"
        init-method="registerServlet">
        <argument ref="blueprintBundleContext" />
    </bean>

    <bean id="httpProxyService"
        class="org.codice.proxy.http.HttpProxyServiceImpl"
        destroy-method="destroy">
        <argument ref="blueprintBundleContext" />
    </bean>

    <service ref="httpProxyService"
        interface="org.codice.proxy.http.HttpProxyService" />

</blueprint>

```

Use the DDF HTTP Proxy

All interactions for starting and stopping the proxy occur through the `HTTPProxyService`. Once the service has been retrieved from OSGi or locally, the following call can be made to start the proxy:

```

//Provide a name for the endpoint and the destination target URL
String endpointName = httpProxyService.start(endpointName, targetURL);

//Provide the destination target URL; endpoint name will be randomly
//generated
String endpointName = httpProxyService.start(targetURL);

```

Once `startProxy()` is called, the proxy will start, and the endpoint name is returned. The official URL for the proxy will be `http://localhost:81/proxy/endpointName` where `endpointName` is the string that was specified or generated. Multiple proxies can be created by calling `startProxy()` multiple times.

To stop the proxy, provide the endpoint name:

```
httpProxyService.stop(endpointName);
```

Configure External Proxy for Target URL

There may be cases where the target destination URL of the DDF HTTP Proxy requires an external proxy to allow the connection. The configuration of the parameters for the external proxy can be accomplished by using command line property parameters for Java. More information on the proxy parameters for Java can be found at <http://docs.oracle.com/javase/7/docs/technotes/guides/net/proxies.html>.

The DDF HTTP Proxy supports the following parameters for both HTTP and HTTPS: `proxyHost`, `proxyPort`, `proxyAuthMethod`, `proxyAuthUsername`, `proxyAuthPassword`, `proxyAuthDomain`, and `proxyAuthHost`. DDF can be configured with these properties by adding them to the `karaf.bat` file located in `%DDF_HOME%/bin`. As an example, the following line within `karaf.bat`:

```

"%JAVA%" %JAVA_OPTS% %OPTS% -classpath "%CLASSPATH%"
-Djava.endorsed.dirs="%JAVA_HOME%\jre\lib\endorsed;%JAVA_HOME%\lib\endorse
d;%KARAF_HOME%\lib\endorsed"
-Djava.ext.dirs="%JAVA_HOME%\jre\lib\ext;%JAVA_HOME%\lib\ext;%KARAF_HOME%\l
ib\ext" -Dkaraf.instances="%KARAF_HOME%\instances"
-Dkaraf.home="%KARAF_HOME%" -Dkaraf.base="%KARAF_BASE%"
-Djava.io.tmpdir="%KARAF_DATA%\tmp" -Dkaraf.data="%KARAF_DATA%"
-Djava.util.logging.config.file="%KARAF_BASE%\etc\java.util.logging.proper
ties" %KARAF_OPTS% %MAIN% %ARGS%

```

The following proxy flags can be added to the end of the line above:

```
-Dhttp.proxyHost=proxy.example.com -Dhttp.proxyPort=80
```

Once this has been added in karaf.bat, the DDF HTTP Proxy will utilize this configuration to configure external proxy use.

Configure Target URL for SSL

There may be cases where the target destination URL of the DDF HTTP Proxy requires SSL certificate support to access a site over HTTPS. The DDF HTTP Proxy has been configured to check the local DDF server trust store for certificates required to access HTTPS sites. By default, the server trust store can be found in %DDF_HOME%/etc/keystores. This path can be changed via the Platform Global Configuration (<https://tools.codice.org/wiki/display/DDF/Platform+Global+Settings>).

DDF Load Balancer

On This Page

- Overview
- Set up the DDF Load Balancer
 - Prerequisites
 - Install
 - Verify
 - Uninstall
- Configure the DDF Load Balancer
 - Configure the HTTP Load Balancer
 - Configure the HTTPS Load Balancer

Overview

Contained within DDF is a Load Balancer utility that allows incoming traffic to be distributed over multiple instances of DDF. The DDF Load Balancer supports two protocols: HTTP and HTTPS. The Load Balancer can be configured to run one protocol or both at the same time. The DDF Load Balancer has been configured to utilize a "Round Robin" algorithm to distribute transactions. The load balancer is also equipped with a failover mechanism. When the load balancer attempts to access a server that is non-functional, it will receive an exception and move on to the next server on the list to complete the transaction. The action will try to be replayed on every server once before failing back to the client.

Set up the DDF Load Balancer

The main method for installing the DDF Load Balancer is to install the application (.kar) file into the hot deploy folder of a full DDF distribution.

Prerequisites

Before the DDF Load Balancer can be installed:

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

Install

Complete the following procedure to install the DDF Load Balancer.

1. Download the application (.kar) file from the artifacts repo (<http://artifacts.codice.org/>).
2. Copy the KAR file into the <INSTALL_DIRECTORY>/deploy folder of a currently running DDF distribution.
3. Uninstall the included jetty feature. **Note: This is due to a bug in the current version of jetty being delivered with DDF, this step will be removed once that version is updated.**

```
features:uninstall jetty
```

Verify

1. Verify all of the Load Balancer's appropriate features have been successfully installed.

DDF Load Balancer installed features

```
ddf@local>features:list | grep -i loadbalancer-app
[installed ] [1.0.0] codice-load-balancer
loadbalancer-app-1.0.0 Load Balancer
[installed ] [7.6.12.v20130726] loadbalancer-jetty
loadbalancer-app-1.0.0 Provide Jetty engine support
```

2. Verify the DDF Load Balancer bundles are Active.

DDF Load Balancer active bundles

```
[ 223] [Active      ] [                      ] [      ] [ 50] camel-http (2.12.1)
[ 224] [Active      ] [                      ] [      ] [ 50] camel-jetty (2.12.1)
[ 258] [Active      ] [                      ] [      ] [ 80] Jetty :: Utilities
(7.6.12.v20130726)
[ 259] [Active      ] [                      ] [      ] [ 80] Jetty :: IO Utility
(7.6.12.v20130726)
[ 260] [Active      ] [                      ] [      ] [ 80] Jetty :: Http Utility
(7.6.12.v20130726)
[ 261] [Active      ] [                      ] [      ] [ 80] Jetty :: Asynchronous
HTTP Client (7.6.12.v20130726)
[ 262] [Active      ] [                      ] [      ] [ 80] Jetty :: Continuation
(7.6.12.v20130726)
[ 263] [Active      ] [                      ] [      ] [ 80] Jetty :: JMX
Management (7.6.12.v20130726)
[ 264] [Active      ] [                      ] [      ] [ 80] Jetty :: Server Core
(7.6.12.v20130726)
[ 265] [Active      ] [                      ] [      ] [ 80] Jetty :: Security
(7.6.12.v20130726)
[ 266] [Active      ] [                      ] [      ] [ 80] Jetty :: Servlet
Handling (7.6.12.v20130726)
[ 267] [Active      ] [                      ] [      ] [ 80] Jetty :: Utility
Servlets and Filters (7.6.12.v20130726)
[ 268] [Active      ] [                      ] [      ] [ 80] Jetty :: XML
utilities (7.6.12.v20130726)
[ 269] [Active      ] [                      ] [      ] [ 80] Jetty :: Webapp
Application Support (7.6.12.v20130726)
[ 270] [Active      ] [                      ] [      ] [ 80] Jetty :: JNDI Naming
(7.6.12.v20130726)
[ 271] [Active      ] [                      ] [      ] [ 80] Jetty :: Plus
(7.6.12.v20130726)
[ 272] [Active      ] [                      ] [      ] [ 80] Jetty :: Websocket
(7.6.12.v20130726)
[ 273] [Active      ] [Created          ] [      ] [ 80] Codice :: Loadbalancer :: Camel (1.0.0)
```

Uninstall

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

Complete the following procedure to uninstall the DDF Load Balancer.

1. Delete the KAR file (`loadbalancer-app-X.Y.kar`) from the `<INSTALL_DIRECTORY>/deploy` directory.
2. Re-install the jetty feature.

```
features:install jetty
```

3. Restart DDF to ensure that all of the Jetty bundles are refreshed properly.

Configure the DDF Load Balancer

The DDF Load Balancer can be configured to allow multiple DDF nodes to be balanced. It can also be configured with a port on which to accept connections. Configurations differ slightly between the HTTP- and HTTPS-based load balancers. All configurations are dynamic in that configuration settings are immediately applied, and it is not necessary to restart DDF.

Configure the HTTP Load Balancer

Complete the following procedure to access the load balancer configuration.

1. Click the Configuration tab in the DDF management console.
2. Scroll down to the configuration entry that is labeled **Platform HTTP Load Balancer**. The configuration for the load balancer contains two fields: **Load Balancer Port** and **IP Address and Port**.
3. In the **Load Balancer Port** field, enter the port number to be accessed when reaching other systems.
4. In the **IP Address and Port** field, enter a comma-delimited list of IP addresses and ports for each DDF node to be balanced. The format for IP address and port is <IP_ADDRESS>:<PORT> (e.g., 192.168.1.123:8181,192.168.1.22:8181).
5. Select the **Save** button when all configuration have been added.

At this point, the load balancer is reset and ready to accept requests. These configurations can be updated at any time without starting the host DDF instance.

Configure the HTTPS Load Balancer

It is possible to run the HTTPS load balancer by itself or run it in parallel with the HTTP load balancer. The HTTPS load balancer utilizes the centralized SSL configurations within DDF, along with the load balancer configurations. Complete the following procedure to configure the HTTPS load balancer.

1. Find the SSL configurations and verify that the values are correct.
2. Click the Configuration tab in the DDF management console.
3. Scroll down to the configuration entry that is labeled **Pax Web Runtime**.
4. Select **Pax Web Runtime**.
5. Ensure that the displayed settings match what is configured in the DDF nodes to be balanced.
6. Save the updated settings or close the window, as applicable.
7. In the configuration table, scroll down to the configuration entry labeled **Platform HTTPS Load Balancer**.
8. The configuration for the load balancer contains three fields: **Load Balancer Host**, **Load Balancer Port**, and **IP Address and Port**.
9. In the **Load Balancer Host** field, enter the host name or IP address to be used for the host load balancer machine.
10. In the **Load Balancer Port** field, enter the port number to be accessed on the load balancer to reach the other systems.
11. In the **IP Address and Port** field, enter a comma delimited list of IP address and port for each DDF node that will be balanced. The format for IP address and port is <IP_ADDRESS>:<PORT> (e.g. 192.168.1.123:8993,192.168.1.22:8993)
12. Select the **Save** button when all configuration have been added.

Since SSL requests will be coming from a client into the load balancer, it is essential that the nodes being balanced have the same security policy and settings. The client has no idea which DDF server it will be connecting with behind the load balancer. The client is responsible for connecting securely with the load balancer, and the load balancer is responsible for connecting securely and consistently with all DDF nodes.

The DDF Load Balancer cannot run on the same port as the DDF Web Console or other web services. If you would like the load balancer to run on this port, change the web console port to a different port number. This configuration parameter can be found in the **Pax Web Runtime** configuration.

DDF STOMP

On This Page

- Overview
 - STOMP
 - Common Query Language (CQL)
- Publish and Subscribe Query Subscription Message
 - Subscription Identifier (subscriptionId)
 - Action (action)
 - Subscription Time to Live (subscriptionTtlType and subscriptionTtl)
 - Query String (queryString)
 - Sources (sources)
 - Creation Date and Last Modified Date (creationDate and lastModifiedDate)
- DDF STOMP Setup
- Configuration
- Send a Subscription Message
- Subscribe or Results
- Returned Messages
- Examples
- Architecture
 - Subscription Message Schema

Overview

The DDF STOMP application allows query subscription messages to be sent to the DDF server via STOMP protocol.

Subscription query messages are defined in JSON format following a defined schema. These messages allow for the management of subscriptions using create, time to live (TTL), update, and delete functions. Catalog queries within the message are defined in CQL format. Results are sent to a STOMP-based topic, which can be subscribed to via a STOMP-based client. Content results will be delivered over time as the subscription query matches incoming data published to the DDF catalog.

STOMP

STOMP is a streaming text-based messaging protocol that supports the delivery of messages as well as publish and subscribe. STOMP mimics HTTP and can utilize TCP-IP, thus making it compatible with many different programming languages. STOMP is very simple to implement and can easily be tested. For more information, refer to <http://stomp.github.io/>.

Common Query Language (CQL)

Common Query Language or CQL is a query language the the OGC has chosen for expressing data filtering. The power of CQL is its strong integration into GeoTools, which helps to represent complex queries as text strings. For more information, refer to <http://docs.geotools.org/latest/userguide/library/cql/index.html>

Publish and Subscribe Query Subscription Message

Subscriptions are stateful and will survive when the server is restarted. Subscription messages are specified in a JSON format. The schema section specifies the values that construct this message, along with the defaults. The sections below describe the meaning of the message values.

Subscription Identifier (subscriptionId)

The subscription identifier is a unique string that is provided by the query subscription requester. The preferred value should be a generated UUID.

Example: "subscriptionId" : "faf4e8493h389fh4398f3h0040"

Action (action)

The action value tells the system what action should take place. The following actions are available:

- CREATE: Creates a new subscription
- UPDATE: Updates an existing subscription (requires subscriptionId)
- DELETE: Deletes an existing subscription (requires subscriptionId)

Example: "action" : "CREATE"

Subscription Time to Live (subscriptionTtlType and subscriptionTtl)

A time to live value can be set on a subscription by providing values for the TTL type and TTL. The subscriptionTtlType value describes the type of TTL that will be used. The following values are available for subscriptionTtlType:

- MILLISECONDS
- SECONDS
- MINUTES
- HOURS
- MONTHS
- YEARS

The subscriptionTtl value is an integer value that specifies the quantity of the subscriptionTtlType. For example, a value of 60 for subscriptionTtl along with a value of 'HOURS' for subscriptionTtlType tells the system that the subscription should last for 60 hours. If a value of 0, -9, or no value is set for subscriptionTtl, the time to live will be infinite.

Example: "subscriptionTtlType" : "HOURS", "subscriptionTtl" : 90

Query String (queryString)

The query string allows the user to specify a query that can filter targeted results. Query string values are specified in CQL format. See the CQL section for more information on using this format.

Example: "queryString" : "anyText LIKE 'Red Truck"

Sources (sources)

Source targets can be specified in the message. Sources will be passed on to all queries to retrieve the correct results from all of the correct sources. The sources value is an array.

Example: "sources" : ["source1", "source2", "source3"]

Creation Date and Last Modified Date (creationDate and lastModifiedDate)

The creation date and last modified date are values that are written into the subscription by the system. The creation date specifies the date and time that the subscription was created in the system. The last modified date specifies the date and time of the last change to the subscription. The user can specify these values in the subscription message, but it will be ignored.

DDF STOMP Setup

DDF STOMP can be found at <https://github.com/codice/ddf-stomp>. When building DDF STOMP a KAR application file is provided. This file can be added to the DDF server by placing the file in the deploy folder. Once the application has been deployed it is best to restart DDF.

Configuration

DDF STOMP has a default configuration that it utilizes. By default, the STOMP server within DDF STOMP runs on port 61613. To change this port number, modifications must be made to activemq.xml and the DDF configuration. The activemq.xml file can be found in the etc/ directory of DDF. Open the file and navigate to the bottom of the page. An entry for transportConnector reads `stomp://0.0.0.0:61613`. The port number at the end can be changed to the desired choice. Once chosen, save the file and exit.

The second half of configuring the port number and other options can be found in the DDF configuration web console. Upon selecting the Configurations tab, look for the configuration named **Publish Subscribe Subscription Query Service**. This configuration has the following options:

- Destination Topic Name: The topic destination where query subscription messages are sent.
- Subscription Topic Name: The prefix of the topic where subscription results are sent.
- STOMP Host: The host name of the STOMP server.
- STOMP Port: The port number of the STOMP server.
- Default Max Results: The maximum number of results to return from a given query.
- Default Request Timeout: The maximum time to wait before a request is timed out.
- Transformer ID: ID that specifies the format in which results are produced (default: geojson). See Extending Catalog Transformers for more information.

Once these configurations have been made, it is best to restart the DDF server.

Send a Subscription Message

Subscription messages can be sent to the system using a STOMP-based client. STOMP is a standardized publish subscribe messaging protocol that utilizes the HTTP protocol for sending data. Connections to the system are asynchronous. For more information, refer to the section on STOMP. To connect the STOMP client to the system, the following information is typically required: STOMP server host, STOMP port, user name, password, and topic name. The username, password, port, and topic name can be found in the publish subscribe query subscriptions configuration. Refer to the Configuration section for more information.

The Gozirra STOMP client (<http://www.germane-software.com/software/Java/Gozirra/>) and the Fuse Source STOMP client (<https://github.com/fusesource/stompjms>) have both been used successfully to test functionality.

Subscribe or Results

A STOMP-based client can be used to retrieve results from a query subscription. STOMP is a standardized publish subscribe messaging protocol that utilizes the HTTP protocol for receiving data. Connections to the system are asynchronous. For more information, refer to the section on STOMP. To connect the STOMP client to the system, the following information is typically required: STOMP server host, STOMP port, user name, password, and topic name. The username, password, port, and topic name can be found in the publish subscribe query subscriptions configuration. Refer to the Configuration section for more information. The topic name for subscription results, which is found in the configuration, is a partial name. The end of the topic name will include the subscription id. For example, if the partial topic name was "/topic/result", and the subscription ID for the subscription was "faf4e8493h389fh4398f3h0040", the actual topic name that would be subscribed to is: "/topic/result/faf4e8493h389fh4398f3h0040". This is the full topic name that would be used with your STOMP client to retrieve results.

The Gozirra STOMP client (<http://www.germane-software.com/software/Java/Gozirra/>) and the Fuse Source STOMP client (<https://github.com/fusesource/stompjms>) have both been used sucessfully to test functionality.

Returned Messages

Return messages will be delivered back to a subscribing STOMP client. All delivered messages are returned in GeoJSON format. When a subscription is first submitted, an initial query is executed on existing data in the catalog. All results based on this query are immediately returned to the subscribing STOMP client via the defined topic. As content is added to the catalog (when these items match the query in the subscription), the content items are immediately returned to the subscribing STOMP client via the defined topic.

Examples

Create a new subscription that will last for 90 hours and searches for any text that is matching red car:

```
{  
  "subscriptionId" : "faf4e8493h389fh4398f3h0060",  
  "action" : "CREATE",  
  "subscriptionTtlType" : "HOURS",  
  "subscriptionTtl" : 90,  
  "queryString" : "anyText LIKE 'red car'"  
}
```

Update a previous subscription to last for three months instead of 90 hours:

```
{  
  "subscriptionId" : "faf4e8493h389fh4398f3h0060",  
  "action" : "UPDATE",  
  "subscriptionTtlType" : "MONTHS",  
  "subscriptionTtl" : 3,  
  "queryString" : "anyText LIKE 'red car'"  
}
```

Delete the previous subscription:

```
{
  "subscriptionId": "faf4e8493h389fh4398f3h0060",
  "action": "DELETE"
}
```

Architecture

The API is set up to utilize STOMP as the protocol for receiving subscription management messages. External third party applications will utilize STOMP to send commands for subscription management and delivery.

Subscription Message Schema

The messages sent over STOMP will be in JSON format. The messages used for subscription management utilize the following schema:

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "id": "http://jsonschema.net",
  "required": false,
  "properties": {
    "queryString": {
      "type": "string",
      "id": "http://jsonschema.net/searchPhrase",
      "required": true
    },
    "sources": {
      "type": "array",
      "id": "http://jsonschema.net/sources",
      "required": false,
      "items": {
        "type": "string",
        "id": "http://jsonschema.net/sources/0",
        "required": false
      }
    },
    "subscriptionId": {
      "type": "string",
      "id": "http://jsonschema.net/subscriptionId",
      "required": true
    },
    "action": {
      "type": "string",
      "id": "http://jsonschema.net/subscriptionId",
      "required": true
    }
}
```

```

"required":true
},
"subscriptionTtl": {
"type": "number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
"subscriptionTtlType": {
"type": "number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
"creationDate": {
"type": "number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
"lastModifiedDate": {
"type": "number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
}
}
}

```

DDF Applications

Overview

DDF consists of applications to support a modular system that allows for installation flexibility. For a general overview of application commands and administration start with the [Administrative Application](#). To examine a more in-depth view of the core functionality of DDFstart with the [DDF Catalog Application](#).

Table of Contents

DDF Administrative Application

- Managing DDF Administrative Application
- Integrating with the Admin Application Service
- Extending DDF Administrative Application
- Securing DDF Administrative Application
- DDF Admin Release Notes

DDF Catalog Application

- Using DDF Catalog Application
- Managing DDF Catalog Application
- Integrating DDF Catalog Application
- Extending DDF Catalog Application
- Securing DDF Catalog Application
- DDF Catalog Application Release Notes

DDF Content Application

- Managing DDF Content Application
- Integrating DDF Content Application
- Extending DDF Content Application
- Securing DDF Content Application
- DDF Content Application Release Notes

DDF Platform Application

- Managing DDF Platform Application
- Integrating DDF Platform Application
- Extending DDF Platform Application
- Securing DDF Platform Application
- DDF Platform Application Release Notes

DDF Registry Application

- Managing DDF Registry Application
- Integrating DDF Registry Application
- Extending DDF Registry Application
- Securing DDF Registry Application
- DDF Registry Application Release Notes

DDF Security Application

- Managing DDF Security Application
- Integrating DDF Security Application
- Extending DDF Security Application
- Securing DDF Security Application
- DDF Security Application Release Notes

DDF Solr Catalog Application

- Managing DDF Solr Catalog Application
- Integrating DDF Solr Catalog Application
- Extending DDF Solr Catalog Application
- Securing DDF Solr Catalog Application
- DDF Solr Catalog Application Release Notes

DDF Spatial Application

- Managing DDF Spatial Application
- Integrating DDF Spatial Application
- Extending DDF Spatial Application
- Securing DDF Spatial Application
- DDF Spatial Application Release Notes

DDF Standard Search UI

- Using DDF Standard Search UI
- Managing DDF Standard Search UI
- Integrating DDF Standard Search UI
- Extending DDF Standard Search UI
- Securing DDF Standard Search UI
- DDF Standard Search UI Release Notes

DDF Administrative Application

On This Page

- Overview
- Integrating with the Admin Application Service
- Administrative User Interface
- Modules

Overview

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.

Integrating with the Admin Application Service

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.

Administrative User Interface

The Admin UI is the centralized location for administering the system. The Admin UI allows an administrator to install and remove selected applications and their dependencies and access configuration pages to configure and tailor system services and properties

Modules

The Admin UI 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 UI. More information on modules, including the ones that come with DDF, is available on the [Modules](#) page.

Managing DDF Administrative Application

On This Page

- Overview
- Prerequisites
- Install Applications
- Verify
- Uninstall Applications
 - Revert the Uninstall
- Upgrade

Overview

The DDF Admin Application contains components that are responsible for the installation and configuration of DDF and other DDF applications.

Prerequisites

Before the DDF Admin application can be installed:

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

- the DDF Catalog Application must be installed

Install Applications

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.

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

Verify

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.

Uninstall Applications

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

```
features:removeurl -u <DDF application's feature repository URL>
```

Example: `features:removeurl -u mvn:ddf.catalog/catalog-app/2.3.0/xml/features`

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

Revert 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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf.catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

Upgrade

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.

```
features:install admin-app
```

4. Complete the steps in the Verify section above to determine if the upgrade was successful.

Modules

On This Page

- Overview
- Included Modules
 - Installer Module
 - Configuration Module
 - Applications Module

Overview

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

Included Modules

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 Module

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

```
features:install admin-modules-installer
```

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

```
https://DDF_HOST:DDF_PORT/admin
```

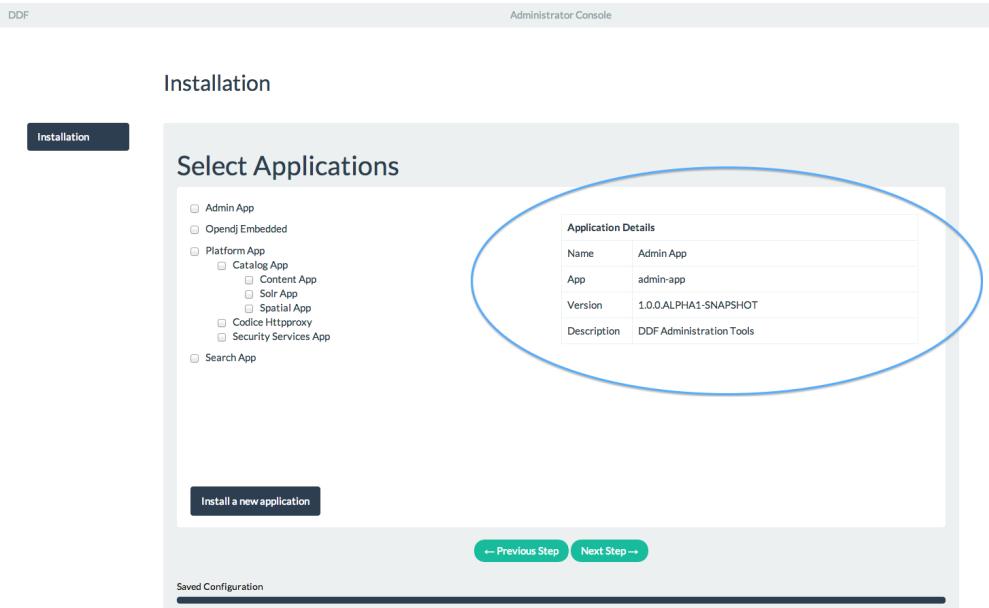
By default DDF_PORT is 8993

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

UI Basics

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.



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

Display the Features File in the Installer

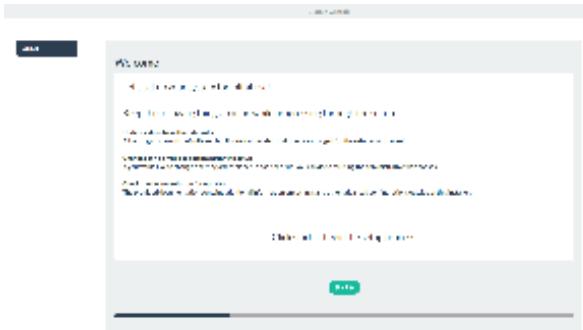
To ensure that the installer can correctly interpret and display application details, there are several guidelines that should be followed when creating the features file for the application.

- 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.
- Verify that the one feature specified to auto-install has a complete list of all of its dependencies to ensure the dependency tree can be constructed correctly.

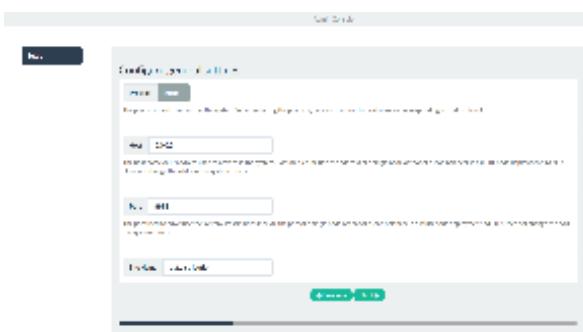
Example Screenshots

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

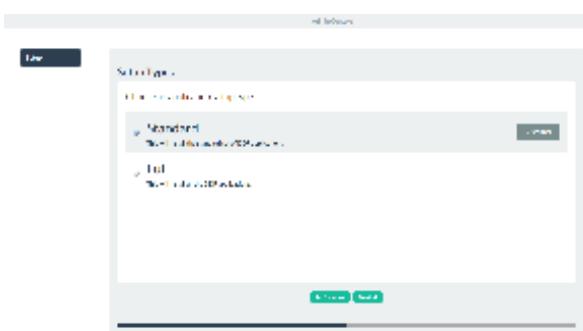
Welcome Page



General Configuration Page



Installation Profile Page



Select Applications Page



Platform App, Admin App, and Security Services App are NOT selectable as they are installed by default and can cause errors if removed.

Final Page



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.

```
features:install admin-modules-configuration
```

2. Open a web browser and navigate to the Admin UI page.

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

3. Select the Configurations tab if not already selected.

Configurations Tab

A screenshot of the DDF Admin UI showing the 'Configurations' tab. The table lists various configuration entries:

Name	Bundle	Action
Maven URL Configuration org.ops4j.pax.url.mvn	OPS4J Pax Url - mvn:	X
Wrap URL configuration		X
Apache Karaf Log org.apache.karaf.log	Apache Karaf :: Shell :: Log Commands	X
Apache Karaf IMX Management org.apache.karaf.management	Apache Karaf :: Management	X
Apache Karaf Shell org.apache.karaf.shell	Apache Karaf :: Shell :: SSH	X
Apache Felix Event Admin Implementation		X
Pax Web Runtime org.ops4j.pax.web	OPS4J Pax Web - Runtime	X
webconsole.name org.apache.karaf.webconsole	Apache Karaf :: Web Console :: Console	X
Apache Felix Web Console Event Plugin		X
Apache Karaf Shell Config		X
WAR URL Configuration org.ops4j.pax.url.war	OPS4J Pax Url - war	X
CXF Servlet Transport org.apache.cxf.osgi	Apache CXF Runtime HTTP Transport	X
Platform Global Configuration		X

Annotations highlight specific features:

- An arrow points to the 'Maven URL Configuration' entry with the text: "Click on configuration name to create NEW configuration entry."
- An arrow points to the 'Apache Karaf Log' entry with the text: "Click on existing configuration to EDIT values."
- An arrow points to the 'REFRESH' button with the text: "Click to REFRESH configuration list. List will also automatically update on regular intervals."
- An arrow points to the 'Delete' icon next to the 'Apache Felix Web Console Event Plugin' entry with the text: "Click to DELETE configuration entry."

Applications Module

The applications module allows administrators to manage application configurations, features, etc.

Toggle the View of the Module

Tile View

This view displays 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.

Tabs within an App

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.

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.

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.

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.
3. A dialogue will pop up that allows you to edit DDF access restrictions.



- a. Once you have configured your realms in your security infrastructure, you can associate them with DDF contexts.
- b. If your infrastructure supports multiple authentication methods, they may be specified on a per-context basis.
- c. Role requirements may be enforced by configuring the required attributes for a given context.

- d. The whitelist allows child contexts to be excluded from the authentication constraints of their parents.

Integrating with the Admin Application Service

On This Page
<ul style="list-style-type: none"> • Overview • API • JMX Managed Bean • Configuration Files <ul style="list-style-type: none"> • Initial Application Installation • Defining Platform Settings • Console Commands • Application Service Interfaces <ul style="list-style-type: none"> • Installing and Uninstalling • Configuring • Interface Details • Application • ApplicationStatus • ApplicationNode • Implementation Details <ul style="list-style-type: none"> • Imported Services • Exported Services

Overview

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

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](#).

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.

Configuration Files

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

Initial Application Installation

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

To minimize the chance of accidentally installing and uninstalling application, 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 after DDF has been started is to use the console commands. Similar 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. This way will be deprecated after the application module has been built for the Admin UI.

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-S  
NAPSHOT/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.

Defining Platform Settings

The platform settings can be configured by a file located in `DDF_HOME/etc/ddf.platform.config.cfg`

The settings in this file can be changed at any time during the DDF lifecycle (before installation, after, or during run-time) and will immediately update the corresponding properties for the platform global configuration.

Console Commands

The application service comes with various console commands that can be executed on the DDF system console. More information is available on the [Application Commands page](#).

Application Service Interfaces

The Application service has multiple ways of interacting with it. These methods range from being at the coding level to operations that can be performed by administrators and end users.

Installing and Uninstalling

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

Configuring

None.

Interface Details

The Application Service comes with several interfaces to use.

ApplicationService

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 that says if 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 a 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

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

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 have a requirement on this application

Implementation Details

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.

Imported Services

Registered Interface	Availability	Multiple	Notes
----------------------	--------------	----------	-------

org.apache.karaf.features.FeaturesService	required	false	Provided by Karaf Framework
org.apache.karaf.bundle.core.BundleStateService	required	true	Installed as part of Platform Status feature.

Exported Services

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

Extending DDF Administrative Application

Overview

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.

This guide supports developers creating extensions of the existing framework.

Securing DDF Administrative Application

Overview

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.

This guide covers implementations and protocols for enhancing security.

DDF Admin Release Notes

Overview

Release notes for Administration application.

Versions
<ul style="list-style-type: none"> Release Version: ddf-admin-1.0.1 Release Version: ddf-admin-1.1.1

Release Version: ddf-admin-1.0.1

Contents

- App Dependencies

App Dependencies

```
commons-collections-3.2.1.jar  
jolokia-osgi-1.2.1.jar  
json-smart-1.1.1.jar  
ops4j-base-util-property-1.4.0.jar  
org.apache.aries.jmx.core-1.1.1.jar  
org.apache.karaf.bundle.core-3.0.0.1.jar  
org.apache.karaf.bundle.springstate-3.0.0.jar
```

Release Version: ddf-admin-1.1.1

Contents

- Summary
- Resolved Issues
- Known Issues

Summary

The Administration UI was updated to include a configuration module for managing applications, features, and configurations.

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

DDF Catalog Application

Overview

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.

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.

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. For more information, see the [Software Versioning](#) section in the [Integrator's Guide Appendix](#).

DDF Catalog Application Table of Contents

- [Using DDF Catalog Application](#)
- [Managing DDF Catalog Application](#) — describes the DDF Catalog application and available options for extending its capabilities
 - [DDF Catalog Application Install and Uninstall](#)
 - [Federation UI](#) — The federation user interface is a convenient way to manage federated data sources for the DDF.
- [Integrating DDF Catalog Application](#)
 - [Integrating Endpoints](#) — Endpoints act as a proxy between the client and the Catalog Framework <https://tools.codice.org/wiki/display/DDF/Catalog+Framework>. Endpoints expose the Catalog Framework to clients using protocols and formats that they understand.
 - [DDF Data Migration](#)
 - [Integrating Catalog Framework](#) — the core of the Catalog application, routes requests and responses between all Catalog Components
 - [DDF Catalog Schematron](#) — provides a pre-ingest interceptor that validates the incoming request against a Schematron rule set (or rule sets)
 - [Developer Use of OGC Filter](#) — answers frequently asked questions about the use of the OGC Filter in DDF Catalog Queries and Subscriptions
- [Extending DDF Catalog Application](#)
 - [Catalog Application Services](#) — This page summarizes DDF internal services within the Catalog application.
 - [Catalog Development Fundamentals](#) — Introduces the fundamentals of working with the Catalog API, including Metacards and use of the OGC Filter for Queries
 - [Extending Catalog Plugins](#) — process Catalog operations, generally before and after they are executed
 - [Extending Operations](#) — represent all transactions that occur in the Catalog, including requests and responses
 - [Extending Data and Metadata Basics](#) — representations of data in the Catalog, primarily metadata represented as a Metocard.
 - [Extending Catalog Framework](#) — This section describes the core components of the Catalog app and Catalog Framework. The Catalog Framework wires all Catalog components together.
 - [Extending Sources](#) — connect Catalog components to data sources, both local and remote
 - [Extending Catalog Transformers](#) — transform data to and from various formats
 - [Extending Federation](#) — provides the capability to extend the DDF enterprise to include Remote Sources, which can include other instances of DDF
 - [Extending Eventing](#) — allows endpoints (and thus external users) to create a "standing query" and be notified when a matching metocard is created, updated, or deleted
 - [Extending Resource Components](#) — used to work with resources, i.e., the data represented by the catalogued metadata
 - [Developing Catalog Components](#) — Describes how to create Catalog components. Used in conjunction with the Javadoc to begin extending the DDF Catalog.
 - [Javadocs](#)
- [Securing DDF Catalog Application](#)
- [DDF Catalog Application Release Notes](#)
 - Release Version: catalog-2.6.1
 - Release Version: catalog-2.5.0
 - Release Version: catalog-2.5.1
 - Release Version: ddf-catalog-2.4.1

Using DDF Catalog Application

Overview

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 section supports end users of this application.

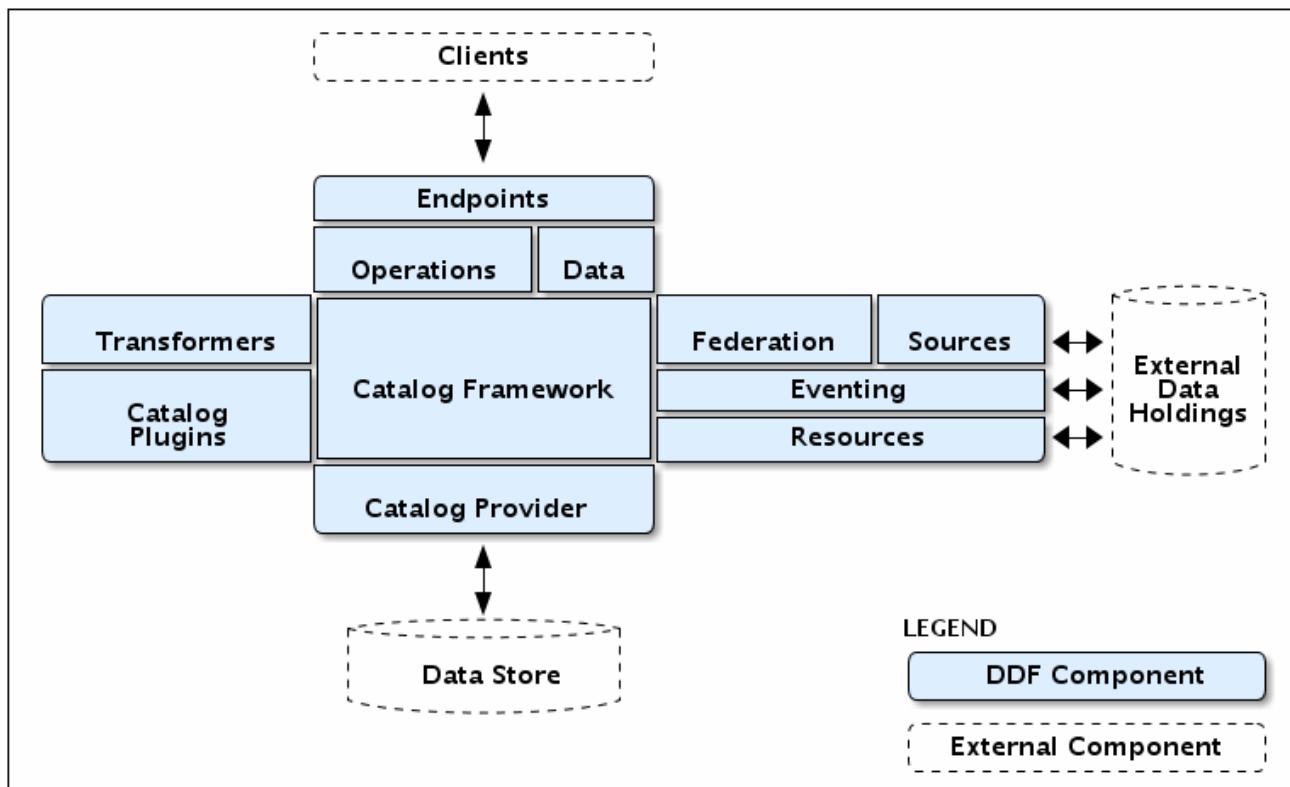
Managing DDF Catalog Application

On This Page

- [Catalog Architecture](#)

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.

Catalog Architecture



DDF Catalog Application Install and Uninstall

On This Page

- Prerequisites
- Install
- Verify
- Uninstall
 - Revert the Uninstall
- Upgrade

Prerequisites

Before the DDF Catalog application can be installed:

- the DDF Kernel must be running
- the DDF Platform application must be installed

Install

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.

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

Verify

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.

Uninstall

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

```
features:removeurl -u <DDF application's feature repository URL>

Example:   features:removeurl -u
mvn:ddf.catalog/catalog-app/2.3.0/xml/features
```

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

The repository URLs for installed applications can be obtained by entering:

```
features:listrepositories -u
```

Revert 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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf .catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

Upgrade

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.

```
features:install admin-app
```

4. Complete the steps in the Verify section above to determine if the upgrade was successful.

Federation UI

Overview

The federation user interface is a convenient way to manage federated data sources for the DDF.

Federation enables including remote sources, including other DDF installations in queries. For a full description of Federation, see Extending Federation.

- Overview
- Installing
- Configuring
- Using
 - Adding a Source
 - Editing a Source
 - Enabling/Disabling a Source
 - Removing a Source

Installing

The Federation UI is installed by default.

Configuring

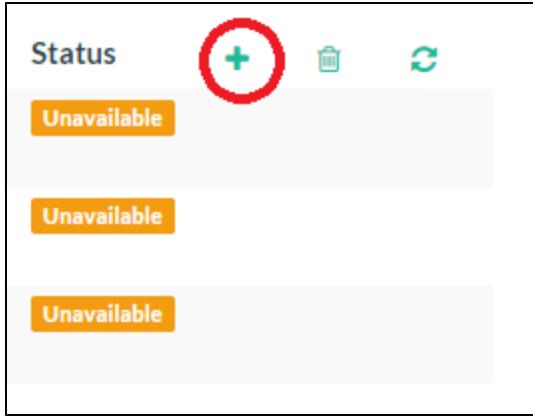
No configuration is required.

Using

1. Go to the admin ui at <http://localhost:8181/admin/index.html>
2. Open DDF-Catalog Application

Adding a Source

1. Press add button.



2. Name the source.

Add Source

Source Name ?
Source0001

Source Type
Catalog OpenSearch Federated Source

OpenSearch service URL ?
http://localhost:81/services/catalog/query

Username ?

Password ?

Always perform local query ?
On Off

Convert to BBox ?
On Off

Add Cancel

3. Choose source type. The type of source selected will determine the options to configure.

Add Source

Source Name ?
Source0001

Source Type
WFS v1.0.0 Federated Source

Catalog CDDA Opensearch Federated Source

Catalog OpenSearch Federated Source

WFS v1.0.0 Federated Source

WFS v1.0.0 Connected Source

WFS 2.0.0 Federated Source

WFS 2.0.0 Connected Source

CSW Federated Source

CSW Connected Source

Convert to BBox ?
On Off

Add Cancel

Editing a Source

1. Click the name of the source to edit.
2. Update relevant properties.

Edit Source0001

Source Name ?	Source0001
Source Type	Catalog OpenSearch Federated Source
OpenSearch service URL ?	http://localhost:8181/services/catalog/query
Username ?	<input type="text"/>
Password ?	<input type="password"/>
Always perform local query ?	<input checked="" type="checkbox"/> On <input type="checkbox"/> Off
Convert to BBox ?	<input checked="" type="checkbox"/> Yes <input type="checkbox"/> No
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

- Click save.

Enabling/Disabling a Source

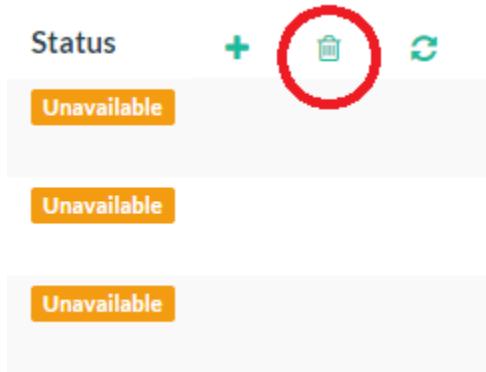
- Select the drop down menu for the source under the heading Type.

Name	Type
source0001	Catalog OpenSearch Federated Source
source0002	Disabled
source0003	Catalog OpenSearch Federated Source

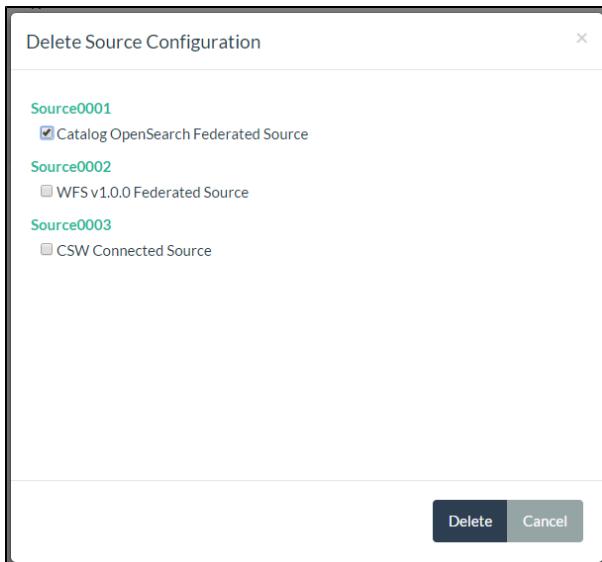
- Set to enabled/disabled.

Removing a Source

- Click the delete source icon.



- Check box next to the source to delete.



3. Click delete.

Integrating DDF Catalog Application

Overview

This guide supports integration of the Catalog Application.

Integrating Endpoints

Overview

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.

On This Page

- Overview
- Existing Endpoints
 - DDF Catalog RESTful CRUD Endpoint
 - OpenSearch Endpoint
- Developing a New Endpoint
 - Common Endpoint Business Logic
- Additional Information



Unknown macro: 'plantuml'

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 in the catalog provider as follows:

Operation	HTTP Request	Details	Example URL
create	HTTP POST	HTTP request body contains the input to be ingested. See InputTransformers for more information.	<code>http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog</code>
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.	<code>http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId></code> where <code><metacardId></code> 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.	<code>http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId></code> where <code><metacardId></code> 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 X-ML representation of the Metocard.	<code>http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/<metacardId></code> where <code><metacardId></code> 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.	<code>http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/sources/<sourceId>/<metacardId></code> where <code><sourceId></code> is the FEDERATED SOURCE ID and <code><metacardId></code> is the Metocard.ID of the Metocard to be retrieved
sources	HTTP GET	Retrieves information about federated sources including source id, availability, contentTypes, and version.	<code>http://<DISTRIBUTION_HOST>:<DISTRIBUTION_PORT>/services/catalog/sources/</code>

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/<meta  
cardId>?transform=<TRANSFORMER_ID>
```

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

Unable to render {children}. Page not found: Included Metocard Transformers.

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

InputTransformers

This REST Endpoint uses InputTransformers to create metacards from a create or a HTTP POST operation. The REST Endpoint dynamically finds InputTransformers that support the Content-Type stated in the HTTP header of a HTTP POST. InputTransformers register as Services with a list of mime-types. The REST Endpoint receives a list of InputTransformers that match the Content-Type and one-by-one calls 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 Content-Type header would need to include application/json in order to match the mime-type GeoJSON Input Transformer supports.

InputTransformers can be added to the system at any time.

Implementation Details

Imported Services

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

Exported Services

Registered Interface	Service Property	Value
ddf.action.ActionProvider	id	catalog.data.metocard.view
ddf.catalog.util.DdfConfigurationWatcher		

Known Issues

None

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. For more information on its parameters view the [OpenSearch Description Document](#) section below.

Installing and Uninstalling

The OpenSearch Endpoint can be installed and uninstalled using the normal processes described in the [Configuring DDF](#) section.

Configuring

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

Using the OpenSearch Endpoint

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

Using the endpoint

From Code:

The OpenSearch specification defines a file format to describe an OpenSearch endpoint. This file is XML-based and is used to 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 <ul style="list-style-type: none">• atom• html• kml see Included Query Response Transformers for more possible values	default: atom

Temporal Extension

OS Element	HTTP Parameter	Possible Values	Comments
start	dtstart	RFC-3399-defined value	yyyy-MM-dd'T'HH:mm:ss.SSSZZ
end	dtend	RFC-3399-defined value	yyyy-MM-dd'T'HH:mm:ss.SSSZZ

The start and end temporal criteria must be of the format specified above. Other formats are currently not supported. Example: 2011-01-01T12:00:00.111-04:00 .

The start and end temporal elements are based on modified timestamps for a 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 deprecated . Use geometry instead.
box	bbox	4 comma-separated EPSG:4326 decimal degrees	west, south, east, north
geometry	geometry	WKT Geometries: POINT, POLYGON, MULTIPOINT, MULTIPOLYGON	Examples: POINT(10 20) where 10 is the longitude and 20 is the latitude. POLYGON ((30 10, 10 20, 20 40, 40 40, 30 10)). 30 is longitude and 10 is latitude for the first point. Make sure to repeat the starting point as the last point to close the polygon.

Extensions

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

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 contextual query format is based on the "IC/DoD Keyword Query Language Specification, V2.0" DRAFT 9/4/2012.

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

Using parenthesis 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
<code>ddf.catalog.CatalogFramework</code>	required	false
<code>ddf.catalog.filter.FilterBuilder</code>	required	false

Exported Services

Registered Interface	Service Property	Value
<code>ddf.catalog.util.DdfConfigurationWatcher</code>		

Example Output

The default output for OpenSearch is Atom. Detailed documentation on Atom output, including query result mapping and example output, can be found on the [Atom Query Response Transformer](#) page.

Known Issues

None

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.*
```

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

4. Retrieve an instance of `CatalogFramework` from the OSGi registry. (Refer to the [Working with OSGi - Service Registry](#) section for examples.)
5. Deploy the packaged service to DDF. (Refer to the [Working with OSGi - Bundles](#) section.)

No implementation of an interface is required

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

Common Endpoint Business Logic

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

Additional Information

Refer to [Working with the Catalog Framework](#) for more details.

DDF Data Migration

On This Page

- [Overview](#)
- [Set Up](#)
 - [Move Metadata from One Catalog Provider to Another](#)
 - [Translate Metadata from One Format to Another](#)

Overview

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.

Set Up

Set up DDF as instructed in [Starting DDF](#) section.

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 `dump` command to export all metadata from the catalog provider into serialized data files . 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. Additional information for ingest and dump commands can be found here: [Catalog Commands](#).

Integrating Catalog Framework

On This Page

- Catalog Framework
- Example Catalog Frameworks
- Catalog Framework Sequence Diagrams
 - Ingest
 - Error Handling
 - Query
 - Product Retrieval
 - Product Caching
 - Design
 - Product Download Status

 Unknown macro: 'plantuml'

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 [Extending 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:

[Integrating 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

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 [Extending 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.

More details of this flow are shown in the diagram below:



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

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:

```
log:set ERROR ingestLogger
```

Query

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



Product Retrieval

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

Product Caching

The Catalog Framework optionally provides caching of products, so future requests to retrieve the same product will be serviced much quicker. If caching is enabled, each time a retrieve product request is received, the Catalog Framework will look in its cache (default location `<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 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.

Design

 Unknown macro: 'plantuml'

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.

DDF Catalog Schematron

On This Page

- Overview
- Understanding Schematron
 - Schematron Validation Plugin
 - Schematron Client "Ruleset" Bundle(s)
- Install and Uninstall
- Configure

Overview

The Schematron Validation Plugin (`plugin-schematron-validation bundle`) provides a pre-ingest interceptor that validates the incoming request against a Schematron ruleset (or rule sets). If the request has warnings or errors based on the Schematron validation, the request is marked as invalid and a SOAP fault is returned with details on the exact reason why the request was invalid. This bundle has the following characteristics:

- It provides the Schematron engine, meaning it provides the infrastructure to load, parse, and apply Schematron rule sets.
- It does not contain any Schematron ruleset(s) - those must be installed (as features) separately.

The Schematron validation bundle works with Schematron rule set bundles to obtain the rules for validation. The Schematron validation bundle and the Schematron rule set bundle are uninstalled by default. More information about Schematron in general can be found at <http://www.schematron.com>.

Understanding Schematron

Schematron is a language for making assertions about the presence or absence of patterns in XML documents. It is not a replacement for XML Schema (XSD) validation. Rather, it is used in conjunction with many grammar-based structure-validation languages, such as XSD.

Schematron is an ISO standard: ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation -- Schematron

Schematron assertions are based on two simple actions:

1. First, find context nodes in the document (typically an element) based on XPath criteria.
2. Then, check to see if some other XPath expressions are true, for each of the nodes returned in the first step.

Schematron assertions (or rules) are defined in a `.sch` file by convention, which is an XML file conforming to Schematron's rules for defining assertions. This file is referred to as a "Schematron ruleset." These rules are contained in one `.sch` file or a hierarchy of `.sch` files. However, there is ultimately one `.sch` file that includes or uses all of the other `.sch` files. This one `.sch` file is the "ruleset" used by the DDF Schematron Validation Service.

Schematron also includes SVRL (Schematron Validation Report Language) report generation, which is in XML format. This report includes the results of all of the Schematron rulesets' assertions, classifying them as warnings or errors (based on the ruleset).

DDF implements Schematron as a Pre-Ingest Plugin, running the Schematron ruleset(s) against each catalog entry in each create and update ingest request that DDF receives. The DDF Schematron Validation Pre-Ingest Plugin consists of two components: the Schematron "engine" and the client ruleset bundle(s). Each are described below.

Schematron Validation Plugin

The Schematron Validation Service is in a single OSGi bundle named `plugin-schematron-validation`. This bundle includes all of the code to implement:

- Loading and pre-compilation of the client ruleset bundle
- Executing the ruleset against ingest requests
- Generating the SVRL report. From this report, the Schematron Validation Service determines if errors and/or warnings were detected during validation. If errors or warnings exist, validation fails and the ingest request is rejected. A SOAP fault is then returned to the client, including details on why the request is invalid.

The client's ruleset bundle determines what rules generate warnings and what rules generate errors. The Schematron Validation Service provides a configuration option (accessible via the Web Console's Configuration page) to suppress warnings. When this option is set, if only warnings are detected during Schematron validation, then the request is considered valid. By default, this suppress warnings option is unset (hence warnings result in invalid requests by default).

Validation is executed per catalog entry in the ingest request. Note that if multiple catalog entries are in the request, Schematron validation stops once a catalog entry is determined to be invalid. For example, if ten catalog entries are in a single create ingest request and entry #4 is invalid, entries 5 through 10 will not even be validated. Schematron returns an invalid status after entry #4 is validated.

If only the Schematron Validation Service is installed, no Schematron validation occurs. This is because the Schematron Validation Service has no ruleset to validate the request against; it only provides the framework for Schematron rulesets to be applied to ingest requests. At least one client ruleset bundle must also be installed.

Schematron Client "Ruleset" Bundle(s)

A client must deploy at least one Schematron ruleset bundle before Schematron validation occurs.

The Schematron ruleset bundle consists of three required items:

- The `.sch` ruleset file defining the Schematron applied rules
- A bundle wiring specification file (e.g., Blueprint, Spring DM, Declarative Services, etc.) specifying the `.sch` file used and associating the ruleset to the Schematron Validation Service
- An OSGi metatype XML file that specifies the configurable options for the Schematron Validation Service (namely the suppress warnings option)

The diagram below illustrates how these Schematron components interact:

Install and Uninstall

The Schematron Validation Library can be installed and uninstalled using the normal processes described in the [Configuring DDF](#) section.

Configure

There are no configuration options for this application.

Developer Use of OGC Filter

Related Topic: [OGC Filter with DDF Frequently Asked Questions in the User's Guide.](#)

Frequently Asked Questions

- How do I create a filter?
- I see there is a Geotools Query, does DDF use that?
- How do I parse a filter to interpret it for a provider?
- I don't see fuzzy capabilities in the OGC Filter, how is that represented?
- What are property names and which ones does DDF use?
- Can I perform a nearest neighbor search in DDF?
- How do I represent a relative temporal search in the OGC Filter?
- How do I represent an entry criteria search in the OGC Filter?

How do I create a filter?

First, get a reference to a `FilterBuilder` from the OSGi registry (refer to [Working with OSGi](#) for more details on Blueprint injection).

```
<reference id="filterBuilder" interface="ddf.catalog.filter.FilterBuilder"
/>
```

Creating filters is simple with the Filter Builder API. Here is a simple example of "AND"ing a contextual search with a temporal search:

```
// filterBuilder injected via Blueprint.
ddf.catalog.filter.FilterBuilder filterBuilder;
String searchPhrase = "ied";

org.opengis.filter.Filter contextualFilter =
filterBuilder.attribute(Metacard.ANY_TEXT).is().like().text(searchPhrase);

Date endDate = Calendar.getInstance().getTime();
Date startDate = new Date(endDate.getTime() - 60000);

org.opengis.filter.Filter temporalFilter =
filterBuilder.attribute(Metacard.MODIFIED).is().during()
.dates(startDate, endDate);

org.opengis.filter.Filter finalFilter =
filterBuilder.allOf(contextualFilter, temporalFilter);
```

Compare this to using the Geotools `FilterFactoryImpl`. The Filter Builder API will create filters in the correct format for the widest support by Extending Sources.

```

org.opengis.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;

org.opengis.filter.expression.Expression propertyName =
filterFactory.property(Metacard.ANY_TEXT) ;
String searchPhrase = "ied" ;
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

org.opengis.filter.Filter contextualFilter =
filterFactory.like(propertyName,searchPhrase, wildcardChar,singleChar,
escapeChar, isCaseSensitive);

Date endDate = Calendar.getInstance().getTime();
Date startDate = new Date(endDate.getTime() - 60000) ;

org.opengis.temporal.Instant startInstant = new
org.geotools.temporal.object.DefaultInstant(new
DefaultPosition(startDate));
org.opengis.temporal.Instant endInstant = new
org.geotools.temporal.object.DefaultInstant(new DefaultPosition(endDate));
org.opengis.temporal.Period period = new
org.geotools.temporal.object.DefaultPeriod(startInstant, endInstant);

org.opengis.filter.Filter temporalFilter =
filterFactory.during(filterFactory.property(Metacard.MODIFIED),
filterFactory.literal(period)) ;

org.opengis.filter.Filter finalFilter = filterFactory.and(contextualFilter,
temporalFilter) ;

```

I see there is a Geotools Query, does DDF use that?

No, DDF has its own `Query` interface, which extends `org.opengis.filter.Filter`.

How do I parse a filter to interpret it for a provider?

The simplest approach is to use the `FilterAdapter` and `FilterDelegate` (refer to [Developing a Filter Delegate](#) for more details).

Alternatively, use the interface `org.opengis.filter.FilterVisitor` and "visit" each part of the filter tree. The `FilterVisitor` uses the [Visitor pattern](#) (<http://www.oodesign.com/visitor-pattern.html>). Geotools has a few implementations of the `FilterVisitor` interface. One of note is the `DefaultFilterVisitor` which visits every node in the filter tree. One can overwrite certain methods of this `FilterVisitor` to add one's own business logic for the various filters. For more information on parsing, refer to the [Parsing Filters](#) section.

In all cases, the provider is responsible for handling the filter, including when the filter is null. The `OpenSearchSource` in DDF checks for a null filter and silently handles it; i.e., no exception is thrown. It is the decision of the provider as to how the filter is handled.

I don't see fuzzy capabilities in the OGC Filter, how is that represented?

In order to represent fuzzy searching, a fuzzy function was created. The fuzzy function is intended to signify when a property should be searched in a fuzzy manner. When it is not present, then no fuzzy capability is intended. More information about the fuzzy function can be found in [Creating Filters](#).

What are property names and which ones does DDF use?

As described in the [Filter Encoding Specification](http://www.opengeospatial.org/standards/filter) (<http://www.opengeospatial.org/standards/filter>), a property is "a facet or attribute or an object referenced by name." In short, property names can be used to represent the value of a property for a particular instance of an object. An example is title or id. The queryable properties of DDF are labeled as constants of a `metocard` class.

Can I perform a nearest neighbor search in DDF?

OGC Filter does not explicitly have a filter for a nearest neighbor operation, but DDF does have an equivalent query. What is recommended is to use the Beyond Filter and provide it with a property and geometry criteria with the most important part being that 0 is provided as the distance. Refer to the example query implementation below:

```
// filterBuilder injected via Blueprint.  
ddf.catalog.filter.FilterBuilder filterBuilder;  
  
String wkt = "POINT (10 20)";  
  
org.opengis.filter.Filter spatialFilter =  
builder.attribute("location").nearestTo().wkt(wkt, 0.0);
```

How do I represent a relative temporal search in the OGC Filter?

Refer to the example temporal offset filter below:

```
// filterBuilder injected via Blueprint.  
ddf.catalog.filter.FilterBuilder filterBuilder;  
long offsetInMillisSeconds = ...;  
org.opengis.filter.Filter temporalFilter =  
filterBuilder.attribute(Metocard.EFFECTIVE).is().during().last(offsetInMillisSeconds);
```

How do I represent an entry criteria search in the OGC Filter?

The property represented by the constant Metocard.ID is used to represent the id of the record to be found, so one could do a search where (Metocard.ID = <given_id>). Refer to the example entry criteria search below:

```
// filterBuilder injected via Blueprint.  
ddf.catalog.filter.FilterBuilder filterBuilder;  
filterBuilder.attribute(Metocard.ID).is().text(id);
```

Extending DDF Catalog Application

Overview

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.

Whitelist

The following packages have been exported by the DDF Catalog application and are approved for use by third parties:

- ddf.catalog
- ddf.catalog.util
- ddf.catalog.event
- ddf.catalog
- ddf.catalog.validation
- ddf.catalog.source
- ddf.catalog.filter
- ddf.catalog.federation
- ddf.catalog.plugin
- ddf.catalog.operation
- ddf.catalog.transform
- ddf.catalog.data
- ddf.catalog.resource
- ddf.measure
- ddf.catalog.filter.delegate
- ddf.catalog.impl.filter
- ddf.services.schematron
- ddf.geo.formatter

Catalog Application Services

Overview

As an OSGi system, DDF does Intra-module conversations via services.
This page summarizes DDF internal services within the Catalog application.

On This Page

- Overview
- Catalog Framework
- Sources
 - CatalogProvider
 - ConnectedSource
 - FederatedSource
- Plugins
 - "Pre-" Plugins
 - "Post-" Plugins
- Transformers

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.

Sources

A source is a system consisting of a catalog containing Metacards.

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.

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.

FederatedSource

A Federated Source is a remote source that can be optionally included or excluded from queries.

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.
Pre-Subscription Plugin	Performs any changes before creating a subscription.
Pre-Delivery Plugin	Performs any changes before delivered a subscribed event.

"Post-" Plugins

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

Transformers

Transformers are used to alter the format of a resource or its metadata to or from the catalog's metocard format

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.

Catalog Development Fundamentals

Overview

This section introduces the fundamentals of working with the Catalog API the OGC Filter for Queries.

On This Page

- Overview
- Simple Catalog API Implementations
- Use of the Whiteboard Design Pattern
- Working with Queries
 - Query Options
 - Creating a query
 - Evaluating a query
- Working with Filters
 - Using Filters
 - Creating Filters
 - Parsing Filters
 - Filter Profile
- Commons-DDF Utilities
 - Noteworthy Classes
- Working with Settings
 - Property Values

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 `MetacardImpl` 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.

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.

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. See the Creating Filters section for more on Filters.

Query Options

Option	Description
StartIndex	1-based index that states which metocard the Source should return first out of the requested metacards.
PageSize	Represents the maximum amount of metacards the Source should return.
SortBy	Determines how the results are sorted and on which property.
RequestsTotalResultsCount	Determines whether the total number of results should be returned.
TimeoutMillis	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.

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
```

Using Filters

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.

The fluent API is best accessed using an IDE that supports code-completion. For additional details, refer to the Catalog API Javadoc.

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)

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 [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 (<http://www.opengeospatial.org/standards/symbol>) 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`.

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 `PropertyIsLike` OGC 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.

 Unknown macro: 'plantuml'

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.

Fuzzy Function Usage

```
String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard,
singleChar

boolean isCaseSensitive = false ;

Filter fuzzyFilter = filterFactory.like(
    new ddf.catalog.impl.filter.FuzzyFunction(
        Arrays.asList(Expression)
(filterFactory.property(Metocard.ANY_TEXT)),
        filterFactory.literal("")),
    searchPhrase,
    wildcardChar,
    singleChar,
    escapeChar,
    isCaseSensitive);

QueryImpl query = new QueryImpl(fuzzyFilter);
```

Parsing Filters

According to the OGC Filter Specification (04-095: <http://www.opengeospatial.org/standards/filter>), 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 Visitor pattern (<http://www.oodesign.com/visitor-pattern.html>). Essentially, `FilterVisitor` instances "visit" each part of the `Filter` tree allowing developers to implement logic to handle the filter's operations. Geotools 8 includes implementations of the `FilterVisitor` interface. The `DefaultFilterVisitor`, as an example, provides only business logic to visit every node in the `Filter` tree. The `DefaultFilterVisitor` methods are meant to be overwritten with the correct business logic. The simplest approach when using `FilterVisitor` instances is to build the appropriate query syntax for a target language as each part of the `Filter` is visited. For instance, when given an incoming `Filter` object to be evaluated against a RDBMS, a `CatalogProvider` instance could use a `FilterVisitor` to interpret each filter operation on the `Filter` object and translate those operations into SQL. The `FilterVisitor` may be needed to support `Filter` functionality not currently handled by the `FilterAdapter` and `FilterDelegate` reference implementation.

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.
```



Unknown macro: 'plantuml'

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').
```

***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;
    ...
}
```

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 PropertyIsGreaterThan 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	LessThan	LessThanOrEqual	Like	NotEqualTo	Null
BINARY_TYPE		✓						✓	✓
BOOLEAN_TYPE		✓						✓	✓
DATE_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
DOUBLE_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
FLOAT_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
GEO_TYPE									✓
INTEGER_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
LONG_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
OBJECT_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
SHORT_TYPE	✓	✓	✓	✓	✓	✓		✓	✓
STRING_TYPE	✓	✓	✓	✓	✓	✓	✓	✓	✓
XML_TYPE		✓					✓		✓

The following table describes each comparison operator.

Operator	Description
PropertyIsBetween	Lower <= Property <= Upper

PropertyIsEqualTo	Property == Literal
PropertyIsGreater Than	Property > Literal
PropertyIsGreater ThanOrEqual To	Property >= Literal
PropertyIsLess Than	Property < Literal
PropertyIsLess ThanOrEqual To	Property <= Literal
PropertyIsLike	Property LIKE Literal Equivalent to SQL "like"
PropertyIsNotEqual To	Property != Literal
PropertyIsNull	Property == null

Logical Operators

Logical operators apply Boolean logic to one or more child filters.

	And	Not	Or
Supported Filters	✓	✓	✓

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.

	After	AnyInteracts	Before	Begins	BegunBy	During	EndedBy	Meets	MetBy	OverlappedBy	TContains	TEquals	TOverlaps
DATE_TYPE	✓		✓			✓							

The following table describes each temporal operator. Literal values can be either date instants or date periods.

Operator	Description
After	Property > (Literal Literal.end)
Before	Property < (Literal Literal.start)
During	Literal.start < Property < Literal.end

Spatial Operators

Spatial operators compare a geometry associated with a property name to a given Literal geometry. The following table displays the supported spatial operators.

	BBox	Beyond	Contains	Crosses	Disjoint	Equals	DWithin	Intersects	Overlaps	Touches	Within
GEO_TYPE		✓	✓	✓	✓		✓	✓	✓	✓	✓

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

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.

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.

Working with Settings

DDF provides the ability to obtain DDF settings/properties. For a list of DDF settings, refer to the Catalog API and Global Settings in the Integrator's Guide. 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.

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
```

2. 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 );
    }
}
```

3. 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>
```

4. 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.*

5. Deploy the packaged service to DDF (refer to the [Working with OSGi - Bundles](#) section).

Extending Catalog Plugins

On This Page

- Overview
- Existing Plugins
 - Pre-Ingest Plugin
 - Metocard Groomer
 - Post-Ingest Plugin
 - Pre-Query Plugin
 - Post-Query Plugin
 - Metocard Resource Size Plugin
 - Other Types of Plugins
 - Pre-Get Resource Plugin
 - Post-Get Resource Plugin
 - Pre-Subscription Plugin
 - Pre-Delivery Plugin
- Developing a Catalog Plugin
 - Create New Plugins
 - Implement Plugin Interface

 Unknown macro: 'plantuml'

Overview

The [Integrating Catalog Framework](#) calls Catalog Plugins to process requests and responses as they enter and leave the Framework.

Existing Plugins



Pre-Ingest Plugin

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. That is, 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.

QueryPlugin Flow



```
graph LR; A[Unknown macro: 'plantuml'] --> B[QueryPlugin Flow]
```

Pre-Query Plugin

Using

Pre-query plugins are invoked before a query operation is sent to any of the Extending 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. That is, 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. 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 response is returned to the requesting endpoint.

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

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. 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 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 Su

scription, 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. That is, 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`.

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. Refer to [Catalog Framework](#) for more information on how these plugins fit in the ingest and query flows.

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 prior to 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 being 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
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

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 (described in the [OSGi Services](#) section).
 - a. Import DDF packages.
 - b. Register plugin class as service to OSGi registry.
4. Deploy to DDF. (Refer to the [Working with OSGi - Bundles](#) section.)

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;
```

An example of an implementation of a `PreQueryPlugin.process()` method's business logic is shown below and can be found in the DDF SDK in the `sample-plugins` project.

How to build PreQueryPlugin filter

```
FilterDelegate<Filter> delegate = new
CopyFilterDelegate(filterBuilder);
try {
    // Make a defensive copy of the original filter (just in case
anyone else expects
    // it to remain unmodified)
    Filter copiedFilter = filterAdapter.adapt(query, delegate);

    // Define the extra query clause(s) to add to the copied filter
    // This will create a filter with a search phrase of:
    //      ((("schematypeseach") and ("test" and
({/ddms:Resource/ddms:security/@ICISM:releasableTo}:"ISAF" or
{/ddms:Resource/ddms:security/@ICISM:releasableTo}:"CAN")))
    Filter contextualFilter =
filterBuilder.attribute(Metacard.ANY_TEXT).like().text("test");
    Filter releasableToFilter1 =
filterBuilder.attribute("/ddms:Resource/ddms:security/@ICISM:releasab
leTo").like().text("ISAF");
    Filter releasableToFilter2 =
filterBuilder.attribute("/ddms:Resource/ddms:security/@ICISM:releasab
leTo").like().text("CAN");
    Filter orFilter = filterBuilder.anyOf(releasableToFilter1,
releasableToFilter2);
    Filter extraFilter = filterBuilder.allOf(contextualFilter,
orFilter);

    // AND this PreQueryPlugin's extra query clause(s) to the copied
filter
    Filter modifiedFilter = filterBuilder.allOf(copiedFilter,
extraFilter);

    // Create a new QueryRequest using the modified filter and the
attributes from the original query
    QueryImpl newQuery = new QueryImpl(modifiedFilter,
query.getStartIndex(),
        query.getPageSize(), query.getSortBy(),
        query.requestsTotalResultsCount(), query.getTimeoutMillis());
    newQueryRequest = new QueryRequestImpl(newQuery,
input.isEnterprise(), input.getSourceIds(), input.getProperties());
}
catch (UnsupportedQueryException e)
{
    throw new PluginExecutionException(e);
}
```

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="[[SamplePreQueryPlugin]]"  
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.

Blueprint descriptor example

```
<service ref="[[SamplePostQueryPlugin]]"  
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="[[SamplePreDeliveryPlugin]]"  
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;
```

An example of an implementation of a `PreSubscriptionPlugin.process()` method's business logic is shown below and can be found in the DDF SDK in the `sample-plugins` project.

PreSubscriptionPlugin example

```
FilterDelegate<Filter> delegate = new
CopyFilterDelegate(filterBuilder);
try {
    // Make a defensive copy of the original filter (just in case
anyone else expects
    // it to remain unmodified)
    Filter copiedFilter = filterAdapter.adapt(input, delegate);

    // Define the extra query clause(s) to add to the copied filter
    Filter extraFilter =
filterBuilder.attribute("/ddms:Resource/ddms:security/@ICISM:releasab
leTo").like().text("CAN");

    // AND the extra query clause(s) to the copied filter
    Filter modifiedFilter = filterBuilder.allOf(copiedFilter,
extraFilter);

    // Create a new subscription with the modified filter
    newSubscription = new SubscriptionImpl(modifiedFilter,
input.getDeliveryMethod(),
input.getSourceIds(), input.isEnterprise());
}
catch (UnsupportedQueryException e)
{
    throw new PluginExecutionException(e);
}
```

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.event
```

4. Export the service to the OSGi registry.

Blueprint descriptor example

```
<service ref="[[SamplePreSubscriptionPlugin]]"
interface="ddf.catalog.plugin.PreSubscriptionPlugin" />
```

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" />
```

Extending Operations

Overview

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 [Extending 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. Refer to the [Developer's Guide](#) for details about using this extensibility.

On This Page

- Overview

 Unknown macro: 'plantuml'

Extending Data and Metadata Basics

Overview

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.

On This Page

- Overview
- Metocard
 - Metocard Type
 - Default Metocard Type and Attributes
 - Required Attributes
 - Extensible Metacards
 - Metocard Extensibility
- Metocard Type Registry
 - Attribute
 - Attribute Type
 - Attribute Format
- Result
- Creating Metacards
- Limitations
- Processing Metacards
- Basic Types

 Unknown macro: 'plantuml'

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.

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 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	metadata-content-type	STRING	Attribute name for accessing the metadata content type of a Metocard.
CONTENT_TYPE_VERSION	metadata-content-type-version	STRING	Attribute name for accessing the version of the metadata content type of a Metocard.
CREATED	created	DATE	Attribute name for accessing the date/time this Metocard was created.
EFFECTIVE	effective	DATE	Attribute name for accessing the date/time of the product represented by the Metocard.
EXPIRATION	expiration	DATE	Attribute name for accessing the date/time the Metocard is no longer valid and could be removed.
GEOGRAPHY	location	GEOMETRY	Attribute name for accessing the location for this Metocard.
ID	id	STRING	Attribute name for accessing the ID of the Metocard.
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 accessing the target namespace of the 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.

It is highly recommended when referencing a default attribute name to use the `ddf.catalog.data.Metocard` constants whenever possible.

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.

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 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.

Metocard Type Registry

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 Metocard.

Attribute

A single field of a metocard, an instance of an attribute type. Attributes are typically indexed for searching by a Source or Catalog Provider.

Attribute 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 byte[] and AttributeType.getBinding() should return Class<Array>of byte.
BOOLEAN	Attributes of this attribute format must have a value that is a Java boolean.
DATE	Attributes of this attribute format must have a value that is a Java date.
DOUBLE	Attributes of this attribute format must have a value that is a Java double.
FLOAT	Attributes of this attribute format must have a value that is a Java float.
GEOMETRY	Attributes of this attribute format must have a value that is a WKT-formatted Java string.
INTEGER	Attributes of this attribute format must have a value that is a Java integer.
LONG	Attributes of this attribute format must have a value that is a Java long.
OBJECT	Attributes of this attribute format must have a value that implements the serializable interface.

SHORT	Attributes of this attribute format must have a value that is a Java short.
STRING	Attributes of this attribute format must have a value that is a Java string and treated as plain text.
XML	Attributes of this attribute format must have a value that is a XML-formatted Java string.

Result

A single "hit" included in a query response.

A result object consists of the following:

- a metocard
- a relevance score if included
- distance in meters if included

Creating Metacards

The quickest way to create a Metocard is to extend or construct the MetacardImpl object. MetacardImpl is the most commonly used and extended Metocard implementation in the system because it provides a convenient way for developers to retrieve and set Attributes without having to create a new MetacardType. MetacardImpl uses BASIC_METACARD as its MetacardType (see below).

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 MetacardTypes in 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 MetacardType 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	MetacardType	representing all required Metocard Attributes
BINARY_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.BINARY.
BOOLEAN_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.BOOLEAN.
DATE_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.DATE .
DOUBLE_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.DOUBLE.
FLOAT_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.FLOAT.
GEO_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.GEOMETRY.
INTEGER_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.INTEGER.
LONG_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.LONG .
OBJECT_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.OBJECT.
SHORT_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.SHORT.

STRING_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.STRING.
XML_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.XML.

Extending Catalog Framework

Overview

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.

Table of Contents

- [Overview](#)
- [Included Catalog Frameworks](#)
 - [Catalog API](#)
 - [Standard Catalog Framework](#)
 - [Catalog Fanout Framework](#)
 - [Catalog Framework Camel Component](#)
- [Working with the Catalog Framework](#)
 - [Methods](#)
- [Developing Complementary Frameworks](#)
 - [Recommendations for Framework Development](#)
- [Developing Console Commands](#)
 - [Console Commands](#)
 - [Custom DDF Console Commands](#)

 Unknown macro: 'plantuml'

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 Catalog API. [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

Catalog Standard Framework

Property	Type	Description	Default Value	Required

fanoutEnabled	Boolean	When enabled the Framework acts as a proxy, federating requests to all available sources. All requests are executed as federated queries and resource retrievals, allowing the framework to be the sole component exposing the functionality of all of its Federated Sources.	false	yes
poolSize	Integer	The federation thread pool size (0 for unlimited)	0	yes
productCacheDirectory	String	Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Catalog Framework will attempt to create those directories. Out of the box (without configuration), the product cache directory is <INSTALL_DIR>/data/product-cache. If a relative path is provided it will be relative to the <INSTALL_DIR>. It is recommended to enter an absolute directory path such as /opt/product-cache in Linux or C:/product-cache in Windows."	(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
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

Managed Service PID	ddf.catalog.CatalogFrameworkImpl
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
ddf.catalog.federation.FederationStrategy	shortname	sorted
org.osgi.service.event.EventHandler	event.topics	ddf/catalog/event/CREATED, ddf/catalog/event/UPDATED, ddf/catalog/event/DELETED

ddf.catalog.CatalogFramework		
org.codice.ddf.configuration.ConfigurationWatcher		
ddf.catalog.event.EventProcessor		
ddf.catalog.plugin.PostIngestPlugin		

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.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

Known Issues

None

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.

 Unknown macro: 'plantuml'

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](#).

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.

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 Web Console.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Federation Thread Pool Size (0 for unlimited)	poolSize	Integer	The federation thread pool size. (0 for unlimited)	0	yes
Default Timeout (in milliseconds)	defaultTimeout	Integer	The maximum amount of time to wait for a response from the Extending Sources .	60000	yes
Product Cache Directory	productCacheDirectory	String	Directory where retrieved products will be cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Catalog Framework will attempt to create those directories. Out of the box (without configuration), the product cache directory is <INSTALL_DIR>/data/product-cache. If a relative path is provided, it will be relative to the <INSTALL_DIR> It is recommended to enter an absolute directory path, such as /opt/product-cache in Linux or C:/product-cache in Windows.	(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
Always Cache Product	cacheWhenCanceled	Boolean	Check to enable caching of retrieved products, even if client cancels the download.	false	no

Managed Service PID	<code>ddf.catalog.impl.service.fanout.FanoutCatalogFramework</code>
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>	<code>shortname</code>	sorted
<code>org.osgi.service.event.EventHandler</code>	<code>event.topics</code>	<code>ddf/catalog/event/CREATED</code> , <code>ddf/catalog/event/UPDATED</code> , <code>ddf/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
<code>ddf.cache.CacheManager</code>		false
<code>ddf.catalog.plugin.PostFederatedQueryPlugin</code>	optional	true
<code>ddf.catalog.plugin.PostIngestPlugin</code>	optional	true
<code>ddf.catalog.plugin.PostQueryPlugin</code>	optional	true
<code>ddf.catalog.plugin.PostResourcePlugin</code>	optional	true
<code>ddf.catalog.plugin.PreDeliveryPlugin</code>	optional	true
<code>ddf.catalog.plugin.PreFederatedQueryPlugin</code>	optional	true
<code>ddf.catalog.plugin.PreIngestPlugin</code>	optional	true
<code>ddf.catalog.plugin.PreQueryPlugin</code>	optional	true
<code>ddf.catalog.plugin.PreResourcePlugin</code>	optional	true
<code>ddf.catalog.plugin.PreSubscriptionPlugin</code>	optional	true
<code>ddf.catalog.resource.ResourceReader</code>	optional	true
<code>ddf.catalog.source.ConnectedSource</code> ,	optional	true
<code>ddf.catalog.source.FederatedSource</code>	optional	true
<code>org.osgi.service.event.EventAdmin</code>		false

Known Issues

None

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
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" />
        <setHeader headerName="operation">
            <constant>CREATE</constant>
        </setHeader>
        <to uri="catalog:framework" />
    </route>
```

Working with the Catalog Framework

Catalog Framework Reference

The Catalog Framework can be requested from the OSGi registry. See [OSGi Services](#) for more details on Blueprint injection.

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 `PreIngestPlugins` before execution and `PostIngestPlugins` after execution.

Query

Query methods search metadata from available [Extending 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 `PreQueryPlugins` before execution and `PostQueryPlugins` 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 `PreResourcePlugins` before execution and `PostResourcePlugins` 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;
```

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).
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

Developing Console Commands

Console Commands

DDF supports development of custom console commands. For more information, see the Karaf website on Extending the Console (<http://karaf.apache.org/manual/latest-2.2.x/developers-guide/extending-console.html>).

Custom DDF Console Commands

DDF includes custom commands for working with the Catalog, as described in the [Console Commands](#) section.

Extending Sources

Overview

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.

On This Page

- Overview
- Existing Source Types
 - Catalog Provider
 - Remote Sources
 - Connected Source
 - Federated Source
 - OpenSearch Source
- Developing a Source
 - Creating a New Source
 - Developing a Filter Delegate

 Unknown macro: 'plantuml'

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.

Remote Sources

Remote sources are read-only data sources that support query operations but cannot be used to create, update, or delete metacards.

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.

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.

 Unknown macro: 'plantuml'

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.

 Unknown macro: 'plantuml'

OpenSearch Source

The OpenSearch source provides a [Federated Source](#) that has the capability to do OpenSearch (<http://www.opensearch.org/Home>) queries for metadata from Content Discovery and Retrieval (CDR) Search V1.1 compliant sources. See <http://www.dni.gov/index.php/about/organization/chief-information-officer/cdr-search> for the CDR specifications. The OpenSearch source does not provide a [Connected Source](#) interface.

The OpenSearch source converts a query into OpenSearch format and then sends that request to the CDR-compliant search service. It then accepts the response, formatted in Atom, and translates the Atom entries to DDMS resources (using the [Atom Query Response Transformer](#)). Existing DDMS is used from the Atom entries when available. If it is not available, a new DDMS document is created from the Atom information. If no (or incomplete) security markings are present, this bundle adds security markings per the security settings that were configured in the Default Security Settings feature. The OpenSearch source then splits out the DDMS documents into results, which are sent back to the endpoint/client via the [Catalog Framework](#).

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
Source Name	shortname	String		DDF-OS
OpenSearch service URL	endpointUrl	String	The OpenSearch endpoint URL, e.g., DDF's OpenSearch endpoint (http://0.0.0.0:8181/services/catalog/query?q={searchTerms}...)	https://example.com
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.	

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.	
Always perform local query	localQueryOnly	Boolean	Always performs a local query by setting <code>src=local</code> OpenSearch parameter in endpoint URL. This must be set if federating to another DDF.	false
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

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 DDMS 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 turned on or off.**

1. Open the Web Console.
 - <http://localhost:8181/system/console>
 - 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."

If checked, the default security settings specified in this configuration are applied to all records returned from the CDR-compliant Search Service that do not contain DDMS metadata. (If a metadata record is returned without DDMS, it also is without security markings). If unchecked, the records without the DDMS metadata are removed from the result set.

5. If the applied defaults are selected, change the settings in the console to the default metadata security.
 - 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.

If this feature is turned off, the records from a CDR-compliant Search Service without DDMS metadata are dropped out of the result set. This means that a Max Results of 20 query parameters is specified and there are only 15 records containing DDMS metadata, the result set only contains 15 records. The total count will accurately specify the number of records that match the query criteria in the Source that is queried.

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.

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:

1. Create a new class that implements the specified Source interface.
2. Implement the required methods.
3. Create an OSGi descriptor file to communicate with the OSGi registry. (Refer to the [OSGi Services](#) section.)
 - a. Import DDF packages.
 - b. Register source class as service to the OSGi registry.
4. Deploy to DDF. (Refer to the [Working with OSGi - Bundles](#) section.)

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 (refer to the [Dummy Catalog Provider](#)).

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`.

```
public class TestFederatedSource implements  
ddf.catalog.source.FederatedSource
```

2. Implement the required methods of the `ddf.catalog.source.FederatedSource` interface.
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" />
```

The DDF Integrator's Guide provides details on the following Federated Sources that come with DDF out of the box (refer to [OpenSearch Source](#)).

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`.

```
public class TestConnectedSource implements  
ddf.catalog.source.ConnectedSource
```

2. Implement the required methods of the `ddf.catalog.source.ConnectedSource` interface.
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" />
```

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. We have seen 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 (<http://today.java.net/pub/a/today/2003/12/04/exceptions.html>)

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.

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 `UnsupportedQueryException`s.

Using the Filter Adapter

The `FilterAdapter` can be requested from the OSGi registry. (Refer to [Working with OSGi](#) for more details on Blueprint injection.)

```
<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
PropertyIsLess Than
PropertyIsLess ThanOrEqual To
PropertyIsLike
PropertyIsNotEqual To
PropertyIsNull

Spatial	Definition
Beyond	True if the geometry being tested is beyond the stated distance of the geometry provided.
Contains	True if the second geometry is wholly inside the first geometry.
Crosses	True if the intersection of the two geometries results in a value whose dimension is less than the geometries 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.
Disjoint	True if the two geometries do not touch or intersect.
DWithin	True if the geometry being tested is within the stated distance of the geometry provided.
Intersects	True if the two geometries intersect. This is a convenience method as you could always ask for Not Disjoint(A,B) to get the same result.
Overlaps	True if the intersection of the geometries results in a value of the same dimension as the geometries that is different from both of the geometries.
Touches	True if and only if the only common points of the two geometries are in the union of the boundaries of the geometries.
Within	True if the first geometry is wholly inside the second geometry.

Temporal
After (http://docs.geotools.org/latest/javadoc/org/opengis/filter/temporal/After.html)
Before (http://docs.geotools.org/latest/javadoc/org/opengis/filter/temporal/Before.html)
During (http://docs.geotools.org/latest/javadoc/org/opengis/filter/temporal/During.html)

Extending Catalog Transformers

Overview

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, Metacard transformers, and Query Response transformers. Additionally, XSLT transformers are provided to aid in developing custom, lightweight Metacard 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.

On This Page

- [Overview](#)
- [Communication Diagram](#)
- [Working with Transformers](#)
 - [Catalog Framework](#)
 - [Dependency Injection](#)
 - [OSGi Service Registry](#)
- [Included Input Transformers](#)
 - [Tika Input Transformer](#)
 - [GeoJSON Input Transformer](#)
- [Developing an Input Transformer](#)
- [Included Metacard Transformers](#)
 - [HTML Metacard Transformer](#)
 - [XML Metacard Transformer](#)
 - [GeoJSON Metacard Transformer](#)
 - [Thumbnail Metacard Transformer](#)
 - [Metadata Metacard Transformer](#)
 - [Resource Metacard Transformer](#)
- [Developing a Metacard Transformer](#)
 - [Create a New Metacard Transformer](#)
 - [Variable Descriptions](#)
- [Included Query Response Transformers](#)
 - [Atom Query Response Transformer](#)
 - [XML Query Response Transformer](#)
 - [SearchUI](#)
- [Developing a Query Response Transformer](#)
 - [Create a New Query Response Transformer](#)
 - [Variable Descriptions](#)
- [XSLT Transformer](#)
 - [XSLT Transformer Framework](#)
 - [Examples](#)
- [Developing an XSLT Transformer](#)
 - [Examples](#)
 - [Implement an XSLT Transformer](#)
 - [Bundle POM Configuration](#)

Unknown macro: 'plantuml'

The current transformers do not support Non-Western Characters (e.g., Hebrew). If the data being transformed contains these characters, they may not be displayed properly. For example, the characters after transformation are not displayed properly (e.g., the word will show up as squares). In other words, transformers only work for UTF-8. It is recommended not to use international character sets.

Communication Diagram

Unknown macro: 'plantuml'

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.

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

```
1 // inject CatalogFramework instance or retrieve an instance
2 private CatalogFramework catalogFramework;
3
4 public RSSEndpoint(CatalogFramework catalogFramework)
5 {
6     this.catalogFramework = catalogFramework ;
7     // implementation
8 }
9
10 // Other implementation details ...
11
12 private void convert(QueryResponse queryResponse ) {
13     // ...
14     String transformerId = "rss";
15
16     BinaryContent content = catalogFramework.transform(queryResponse,
transformerId, null);
17
18     // ...
19
20 }
```

Line #	Action
4	CatalogFramework is injected, possibly by dependency injection framework.
16	queryResponse is transformed into the RSS format, which is stored in the BinaryContent instance

Dependency Injection

Using Blueprint or another injection framework, transformers can be injected from the OSGi Service Registry. See [OSGi Services](#) for more details on how to use injected instances.

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);
```

See [Working with OSGi - Service Registry](#) for more details.

OSGi Service Registry

In the vast majority of cases, working with the OSGi Service Reference directly should be avoided. Instead, dependencies should be injected via a dependency injection framework like Blueprint.

Transformers are registered with the OSGi Service Registry. Using a `BundleContext` and a filter, references to a registered service can be retrieved.

OSGi Service Registry Reference Example

```
ServiceReference[] refs =

bundleContext.getServiceReferences(ddf.catalog.transform.InputTransformer.
class.getName(),"(shortname=" + transformerId + ")");
InputTransformer inputTransformer = (InputTransformer)
context.getService(refs[0]);
Metocard entry = inputTransformer.transform(messageInputStream);
```

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 [Extending 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 following 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 Metacard. 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 DDF Content Repository and 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

Install the catalog-transformer-tika feature using the [Web Console](http://localhost:8181/system/console) (<http://localhost:8181/system/console>) or System 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 DDF Content Repository and/or the Metadata Catalog.

Service Properties

Key	Value
mime-type	application/pdf application/vnd.openxmlformats-officedocument.wordprocessingml.document application/vnd.openxmlformats-officedocument.spreadsheetml.sheet application/vnd.openxmlformats-officedocument.presentationml.presentation application/vnd.openxmlformats-officedocument.presentationml.presentation application/vnd.ms-powerpoint.presentation.macroenabled.12 application/vnd.ms-powerpoint.slideshow.macroenabled.12 application/vnd.openxmlformats-officedocument.presentationml.slideshow application/vnd.ms-powerpoint.template.macroenabled.12 application/vnd.oasis.opendocument.text
shortname	
id	
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](#).

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 Web Console (<http://localhost:8181/system/console>) or System 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 (<http://geojson.org/geojson-spec.html#geojson-objects>) 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.

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 (<http://geojson.org/geojson-spec.html>) 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. It uses DDF's extensible metocard support to do this. 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",  
        "metocard-type": "ddf.metocard.custom.type"  
    },  
    "type": "Feature",  
    "geometry": {  
        "type": "Point",  
        "coordinates": [  
            30.0,  
            10.0  
        ]  
    }  
}
```

When the GeoJSON Input Transformer gets GeoJSON with the MetocardType specified, it will perform a lookup in the MetocardTypeRegistry to obtain the specified MetocardType in order to understand how to parse the GeoJSON. If no MetocardType is specified, the GeoJSON Input Transformer will assume the default MetocardType. If an unregistered MetocardType is specified, an exception will be returned to the client indicating that the MetocardType was not found.

Packaging Details

Feature Information

N/A

Included Bundles

N/A

Services

Exported Services

ddf.catalog.transform.InputTransformer	
mime-type	application/json
id	geojson

Implementation Details

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.

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>  
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Variable Descriptions

Blueprint Service Properties

Key	Description of Value	Example
shortname	(Required) An abbreviation for the return-type of the BinaryContent being sent to the user.	<i>atom</i>
title	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	<i>Atom Entry Transformer Service</i>
description	(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

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>
```

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. Note that the specified XSL stylesheet must be on the classpath of the bundle that this blueprint file is packaged in.
5	Defines the XSLT to be used to transform the vehicle input into GeoJSON format using the Apache Camel provided XSLT component.
6	Defines the route node that accepts GeoJSON formatted input and transforms it into a Mmtacard, using the DDF custom catalog component that is an InputTransformer registered with an id of <code>application/json; id=geojson</code> .

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`

Included Metocard Transformers

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?transformer=html
```

Example Output



Implementation Details

Registered Interface	Service Property	Value
<code>ddf.catalog.transform.MetocardTransformer</code>	<code>title</code>	View as html...
	<code>description</code>	Transforms query results into html
	<code>shortname (for backwards compatibility)</code>	html

Known Issues

None

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 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 XML shortname.

```
http://localhost:8181/services/catalog/ac0c6917d5ee45bfb3c2bf8cd2ebaa67?tr  
ansform=xml
```

Example Output

The schema file for the XML metocard format, `metocard.xsd`, is attached to this page.

```
<ns3:metocard ns1:id="ac0c6917d5ee45bfb3c2bf8cd2ebaa67"  
xmlns:ns1="http://www.opengis.net/gml" xmlns:ns3="urn:catalog:metocard">  
    <ns3:type>ddf.metocard</ns3:type>  
    <ns3:source>ddf</ns3:source>  
    <ns3:dateTime name="modified">  
        <ns3:value>2013-01-29T17:09:19.980-07:00</ns3:value>  
    </ns3:dateTime>  
    <ns3:stringxml name="metadata">  
        <ns3:value>  
            <ddms:Resource  
                xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/2.0/"  
                xmlns:ICISM="urn:us:gov:ic:ism:v2"  
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
                <ddms:identifier ddms:qualifier="http://example.test#URI"  
                    ddms:value="http://example.test.html"/>  
                <ddms:title ICISM:classification="U"  
                    ICISM:ownerProducer="USA">Example Title</ddms:title>  
                <ddms:description ICISM:classification="U"  
                    ICISM:ownerProducer="USA">Example description.</ddms:description>  
                <ddms:dates ddms:posted="2013-01-29"/>  
                <ddms:rights ddms:copyright="true"  
                    ddms:intellectualProperty="true" ddms:privacyAct="false"/>  
                <ddms:creator ICISM:classification="U"  
                    ICISM:ownerProducer="USA">  
                    <ddms:Person>  
                        <ddms:name>John Doe</ddms:name>  
                        <ddms:surname>Doe</ddms:surname>  
                    </ddms:Person>  
                </ddms:creator>
```

```
<ddms:subjectCoverage>
  <ddms:Subject>
    <ddms:category ddms:code="nitf" ddms:label="nitf"
ddms:qualifier="SubjectCoverageQualifier"/>
    <ddms:keyword ddms:value="schematypesearch"/>
  </ddms:Subject>
</ddms:subjectCoverage>
<ddms:temporalCoverage>
  <ddms:TimePeriod>
    <ddms:start>2013-01-29</ddms:start>
    <ddms:end>2013-01-29</ddms:end>
  </ddms:TimePeriod>
</ddms:temporalCoverage>
<ddms:security ICISM:classification="U"
ICISM:ownerProducer="USA"/>
</ddms:Resource>
</ns3:value>
</ns3:stringxml>
<ns3:string name="resource-size">
  <ns3:value>N/A</ns3:value>
</ns3:string>
<ns3:geometry name="location">
  <ns3:value>
    <ns1:Point>
      <ns1:pos>2.0 1.0</ns1:pos>
    </ns1:Point>
  </ns3:value>
</ns3:geometry>
<ns3:dateTime name="created">
  <ns3:value>2013-01-29T17:09:19.980-07:00</ns3:value>
</ns3:dateTime>
<ns3:string name="resource-uri">
  <ns3:value>http://example.com</ns3:value>
</ns3:string>
<ns3:string name="metadata-content-type-version">
  <ns3:value>v2.0</ns3:value>
</ns3:string>
<ns3:string name="title">
  <ns3:value>Example Title</ns3:value>
</ns3:string>
<ns3:string name="metadata-content-type">
  <ns3:value>Resource</ns3:value>
</ns3:string>
<ns3:dateTime name="effective">
```

```

<ns3:value>2013-01-29T17:09:19.980-07:00</ns3:value>
</ns3:dateTime>
</ns3:metocard>

```

Implementation Details

Metocard to XML Mappings

Metocard Variables	XML Element
id	metocard/@gml:id
metocardType	metocard/type
sourceld	metocard/source
all other attributes	metocard/<AttributeType>[name='<AttributeName>']/value For instance, the value for the metocard attribute named "title" would be found at metocard/string[@name='title']/value

AttributeTypes

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

Known Issues

None

GeoJSON Metocard Transformer

GeoJSON Metocard Transformer translates a Metocard into GeoJSON.

Installing and Uninstalling

Install the catalog-transformer-json feature using the Web Console (<http://localhost:8181/system/console>) or System Console.

Configuring

None

Using

The GeoJSON Metocard Transformer can be used programmatically by requesting a MetocardTransformer 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://\u2fexample.com",  
    "created": "2012-08-31T23:55:19.518+0000",  
    "metadata-content-type-version": "myVersion",  
    "metadata-content-type": "myType",  
    "metadata": "<xml>text</xml>",  
    "modified": "2012-08-31T23:55:19.518+0000",  
    "metacard-type": "ddf.metacard"  
  },  
  "type": "Feature",  
  "geometry": {  
    "type": "LineString",  
    "coordinates": [  
      [  
        30.0,  
        10.0  
      ],  
      [  
        10.0,  
        30.0  
      ],  
      [  
        40.0,  
        40.0  
      ]  
    ]  
  }  
}
```

Implementation Details

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

Known Issues

None

Thumbnail Metacard Transformer

The Thumbnail Metacard Transformer retrieves the thumbnail bytes of a Metacard by returning the `Metacard.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="metacardTransformer"
interface="ddf.catalog.transform.MetacardTransformer"
filter="(id=thumbnail)"/>
```

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

Implementation Details

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

Known Issues

None

Metadata Metacard Transformer

The Metadata Metacard Transformer returns the `Metacard.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 using the normal processes described in the [Configuring DDF](#) section of this documentation.

Configuring

None

Using

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

Example REST GET method with the Metadata MetocardTransformer

```
http://localhost:8181/services/catalog/0123456789abcdef0123456789abcdef?transformer=metadata
```

Implementation Details

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

Known Issues

None.

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 by installing the feature associated with the transformer "catalog-transformer-resource". To uninstall the transformer, you must uninstall 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	Get Resource ...

Known Issues

None

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>  
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Variable Descriptions

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.

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 (<http://tools.ietf.org/html/rfc4287>) feed. The Atom transformer maps a `QueryResponse` object as described in the Query Result Mapping.

Installing and Uninstalling

Catalog Transformers application will install this feature when deployed. This transformer's feature, `catalog-transformer-atom`, can be uninstalled or installed using the normal processes described in the [Configuring DDF](#) section.

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](#) (<https://tools.codice.org/#>). 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. (See [Implementation Details](#) for details about acquiring the service.)

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/"
    xmlns: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/ee7a161e01754b9db1
  872bfe39dlea09" rel="alternate" title="View Complete Metacard" />
<category term="Resource" />
<georss:where xmlns:gml="http://www.opengis.net/gml">
  <gml:Point>
    <gml:pos>32.8751900768792 13.1874561309814</gml:pos>
  </gml:Point>
</georss:where>
<content type="application/xml">
  <ns3:metacard xmlns:ns3="urn:catalog:metacard"
    xmlns:ns2="http://www.w3.org/1999/xlink"
    xmlns:ns1="http://www.opengis.net/gml"
    xmlns:ns4="http://www.w3.org/2001/SMIL20/"
    xmlns:ns5="http://www.w3.org/2001/SMIL20/Language"
    ns1:id="4535c53fc8bc4404a1d32a5ce7a29585">
    <ns3:type>ddf.metacard</ns3:type>
    <ns3:source>ddf.distribution</ns3:source>
    <ns3:geometry name="location">
      <ns3:value>
        <ns1:Point>
          <ns1:pos>32.8751900768792 13.1874561309814</ns1:pos>
        </ns1:Point>
      </ns3:value>
    </ns3:geometry>
    <ns3:dateTime name="created">
      <ns3:value>2013-01-31T16:22:31.648-07:00</ns3:value>
    </ns3:dateTime>
    <ns3:dateTime name="modified">
      <ns3:value>2013-01-31T16:22:31.648-07:00</ns3:value>
    </ns3:dateTime>
    <ns3:stringxml name="metadata">
      <ns3:value>
        <ns6:xml xmlns:ns6="urn:sample:namespace"
          xmlns="urn:sample:namespace">Example description.</ns6:xml>
      </ns3:value>
    </ns3:stringxml>
    <ns3:string name="metadata-content-type-version">
      <ns3:value>myVersion</ns3:value>
    </ns3:string>
    <ns3:string name="metadata-content-type">
      <ns3:value>myType</ns3:value>
    </ns3:string>
    <ns3:string name="title">
      <ns3:value>Example title</ns3:value>
    </ns3:string>
  </ns3:metacard>
```

```

</content>
</entry>
</feed>

```

Query Result Mapping

XPath to Atom XML	Value
/feed/title	"Query Response"
/feed/updated	ISO 8601 date/Time of when the feed was generated
/feed/id	Generated UUID URN (http://en.wikipedia.org/wiki/Universally_Unique_Identifier)
/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:sourcelId	Source Id from which the Result came. <i>Metocard.getSourceId()</i>
/feed/entry/relevance:score	Result's relevance score if applicable. <i>Result.getRelevanceScore()</i>
/feed/entry/id	urn:catalog:id:<Metocard.ID>
/feed/entry/title	Metocard.TITLE
/feed/entry/updated	ISO 8601 date/Time of Metocard.MODIFIED
/feed/entry/published	ISO 8601 date/Time of Metocard.CREATED
/feed/entry/link[@rel='related']	URL to retrieve underlying resource (if applicable and link is available)
/feed/entry/link[@rel='alternate']	Link to alternate view of the Metocard (if a link is available)
/feed/entry/category	Metocard.CONTENT_TYPE
/feed/entry//georss:where	GeoRSS GML of every Metocard attribute with format <i>AttributeFormat.GEOMETRY</i>
/feed/entry/content	Metocard XML generated by <code>ddf.catalog.transform.MetocardTransformer</code> with shortname= If no transformer found, /feed/entry/content/@type will be <code>text</code> and Metocard.ID is displayed
Sample Content with no Metocard Transformation	
<pre><content type="text">4e1f38d1913b4e93ac622e6c1b258f89</content></pre>	

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 Web Console (<http://localhost:8181/system/console>) or System Console.

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:metocard"
  xmlns:ns4="http://www.w3.org/2001/SMIL20/"
  xmlns:ns5="http://www.w3.org/2001/SMIL20/Language">
  <ns3:metocard ns1:id="000ba4dd7d974e258845a84966d766eb">
    <ns3:type>ddf.metocard</ns3:type>
    <ns3:source>southwestCatalog1</ns3:source>
    <ns3:dateTime name="created">
      <ns3:value>2013-04-10T15:30:05.702-07:00</ns3:value>
    </ns3:dateTime>
    <ns3:string name="title">
      <ns3:value>Input 1</ns3:value>
    </ns3:string>
  </ns3:metocard>
  <ns3:metocard ns1:id="00c0eb4ba9b74f8b988ef7060e18a6a7">
    <ns3:type>ddf.metocard</ns3:type>
    <ns3:source>southwestCatalog1</ns3:source>
    <ns3:dateTime name="created">
      <ns3:value>2013-04-10T15:30:05.702-07:00</ns3:value>
    </ns3:dateTime>
    <ns3:string name="title">
      <ns3:value>Input 2</ns3:value>
    </ns3:string>
  </ns3:metocard>
</ns3:metacards>
```

Implementation Details

Registered Interface	Service Property	Value
ddf.catalog.transform.QueryResponseTransformer	shortname	xml
	description	Transforms query results into xml
	title	View as XML...

See [XML Metocard Transformer Implementation Details](#) as to how metocard Java object information is mapped into 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 using the normal processes described in the [Configuring DDF](#) section.

Configuring

In the Admin Console the SearchUI can be configured under the Catalog HTML Query Response Transformer.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Header	header	String	Specifies the header text to be rendered on the SearchUI		yes
Footer	footer	String	Specifies the footer text to be rendered on the SearchUI		yes
Template	template	String	Specifies the path to the Template	/templates/searchpage.ftl	yes
Text Color	color	String	Specifies the Text Color of the Header and Footer	yellow	yes
Background Color	background	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, clicking on the following link <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	Metocard.TITLE	The title maybe hyperlinked to view the full Metocard
Source	Metocard.getSourceld()	Source where the Metocard was discovered
Location	Metocard.LOCATION	Geographical location of the Metocard
Time	Metocard.CREATED or Metocard.EFFECTIVE	Time received/created
Thumbnail	Metocard.THUMBNAIL	No column shown if no results have thumbnail
Resource	Metocard.RESOURCE_URI	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 Metocard.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*.

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.

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>  
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Variable Descriptions

Blueprint Service properties

Key	Description of Value	Example
-----	----------------------	---------

shortname	An abbreviation for the return-type of the BinaryContent being sent to the user.	<i>atom</i>
title	A user-readable title that describes (in greater detail than the shortname) the service.	<i>Atom Entry Transformer Service</i>
description	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>

XSLT Transformer

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.

Developing an XSLT Transformer

The XSLT Transformer Framework allows developers to create light-weight Query Response Transformers (<https://login.macefusion.com/secure/login?service=https%3A%2F%2Fwiki.macefusion.com%2Fpages%2Fviewpage.action%3Ftitle%3DQuery%2BResponse%2BTransformer%26spaceKey%3DDDF>) and Metocard Transformers (<https://login.macefusion.com/secure/login?service=https%3A%2F%2Fwiki.macefusion.com%2Fpages%2Fviewpage.action%3Ftitle%3DMetocard%2BTransformer%26spaceKey%3DDDF>) 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>${project.artifactId}</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 Metocard 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>
  <metocard>
    <id>[[Metocard 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>
      [[Metocard XML]]
    </document>
  </metocard>
  ...
</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
- Metocard Transformers
 - `id` - metocard ID
 - `siteName` - source ID
 - `services` - list of displayable titles and URLs of available metocard 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>
          </instructions>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Line #	Comment
8	Use the current release version.
21	Set the MIME type to the RSS MIME type.

3. 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.

4. 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:for-each>  
</channel>
```

```

    </rss>
  </xsl:template>
</xsl:stylesheet>

```

Line #	Comment
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.

Extending Federation

Overview

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 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](#).

On This Page

- [Overview](#)
- [Communication Diagram](#)
- [Federation Strategy](#)
 - [Usage](#)
 - [Sorted Federation Strategy](#)
 - [Configuration](#)

 Unknown macro: 'plantuml'

Communication Diagram

 Unknown macro: 'plantuml'

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.

Sorted Federation Strategy

The Sorted 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:

- metocard'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.

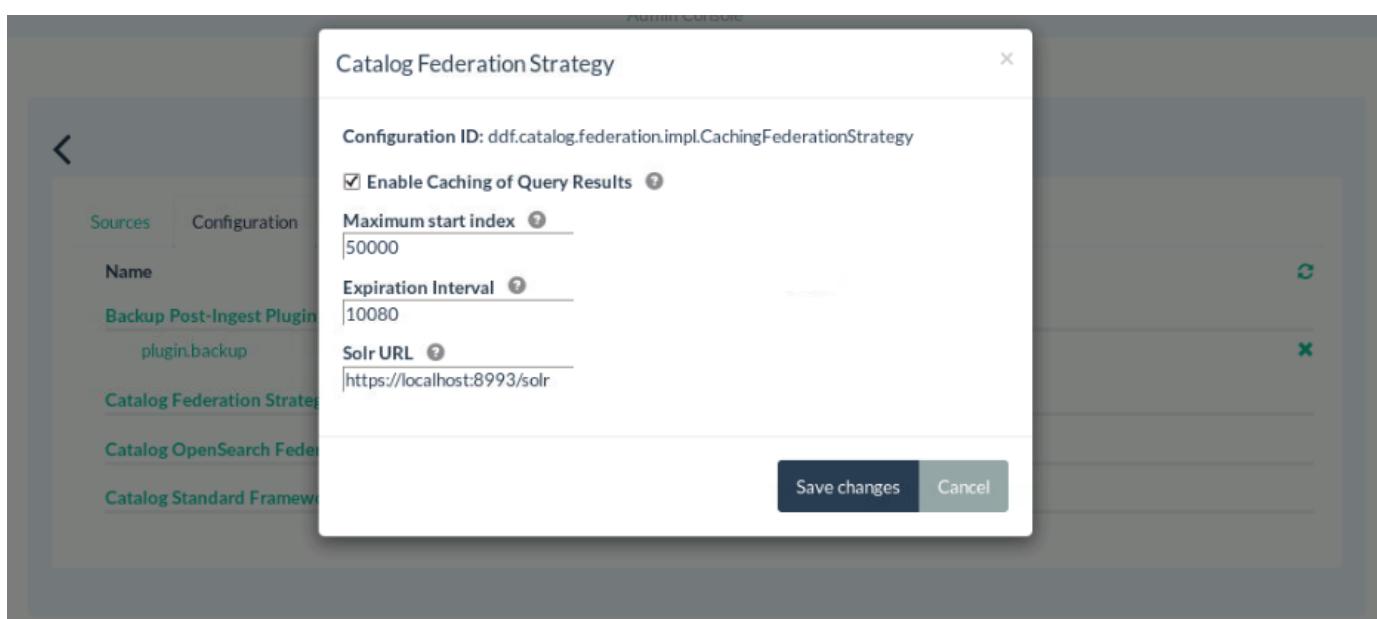
The default sorting value/order automatically used is relevance descending.

The `SortedFederationStrategy` 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 `SortedFederationStrategy` expects that result to come at the end of the result list.

Configuration

The Sorted Federation Strategy configuration can be found in the web console under **Configuration Catalog Federation Strategy**.

Property	Type	Description	Default Value	Required
Enable Caching of Query Results	Check Box	Check Box is true by default enabling the caching of query results. Disabling caching of query results will force discovery from the source each time a query is run. This may slow down query response.	Checked	Optional
Maximum Start Index	Integer	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 will cycle through all records in memory. Things to consider when setting this value are: <ul style="list-style-type: none">• How much memory is allocated to the DDF Server• How many sites are being federated with.	50000	yes
Expiration Interval	Integer	Solr Cache expiration interval in minutes. Default value is equivalent to 7 days.	10080	yes
Solr URL	url	https://localhost:8993/solr		



Managed Service PID ddf.catalog.SortedFederationStrategy

Extending Eventing

Overview

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 hostility 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

On This Page

- Overview
- Subscription
 - Subscription Lifecycle
 - Creation
 - Evaluation
 - Update Evaluation
 - Durability
 - Eventing Sequence Diagrams
 - Creating a Subscription
 - Delivery Method
 - Event Processor
 - Event Processing and Notification
 - Standard Event Processor
 - Installing and Uninstalling
 - Known Issues
 - Fanout Event Processor
 - Installing and Uninstalling
 - Known Issues
 - Working with Subscriptions
 - Creating a Subscription
 - Using DDF Implementation
 - Creating a Custom Implementation
 - Registering a Subscription
 - Creating a Delivery Method
 - Deleting a Subscription

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.

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.

Eventing Sequence Diagrams

This section describes the basic subscription creation flow, how an event is processed by the Catalog, and subsequently the broadcast to subscribers. These following flow charts are for illustrative purposes only and do not necessarily represent every step in each procedure.

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.

 Unknown macro: 'plantuml'

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

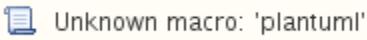
The Event Processor provides an engine that creates, updates, and deletes [subscriptions](#) for event notification. These subscriptions optionally specify a filter criteria so that only events of interest to the subscriber are posted for notification.

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 [Catalog Framework](#) 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.

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

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(Metocard.ANY_TEXT).like().text("*");

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

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

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

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

```

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. See the [Creating Filters](#) section of the Developer's Guide for more information on OGC filters.

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. More information on how to create a `DeliveryMethod` is in the [Creating A Delivery Method](#) section of this page.

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;  
    }  
}
```

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);
```

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(Metacard newMetacard) {  
        // Perform custom code on create  
    }  
  
    @Override  
    public void updatedHit(Metacard newMetacard, Metacard oldMetacard) {  
        // Perform custom code on update (where both new and old metacards  
        matched filter)  
    }  
  
    @Override  
    public void updatedMiss(Metacard newMetacard, Metacard oldMetacard) {  
        // Perform custom code on update (where one of the two metacards did not  
        match the filter)  
    }  
  
    @Override  
    public void deleted(Metacard oldMetacard) {  
        // Perform custom code on delete  
    }  
}
```

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);
```

Extending Resource Components

Overview

Resource components are used when working with resources, i.e., the data that is represented by the cataloged 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.

On This Page

- Overview
- Resource Readers
 - URL Resource Reader
 - Installing and Uninstalling
 - Configuring
 - Using
 - Implementation Details
 - Known Issues
 - Developing a Resource Reader
 - Create a New ResourceReader
 - Implementing the ResourceReader Interface
 - `retrieveResource`
 - Implement `retrieveResource()`
 - `getSupportedSchemes`
 - Export to OSGi Service Registry
 - Resource Writers
 - Examples
 - Developing a Resource Writer
 - Create a New ResourceWriter
 - Developing a Registry Client
 - Example
 - Working with Resources
 - Metacards and Resources
 - Retrieve Resource
 - Options
 - Store Resource
 - BinaryContent
 - Additional Information

 Unknown macro: 'plantuml'

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.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`.

Installing and Uninstalling

`URLResourceReader` is installed by default with the DDF Catalog.

Configuring

This `URLResourceReader` has no configurable properties. It can only be installed or uninstalled.

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	<ul style="list-style-type: none">• <code>http</code>• <code>https</code>• <code>file</code>
-------------------	--

If a file-based URL is passed to the `URLResourceReader`, that file path needs to be accessible by the DDF instance.

Known Issues

None

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 DDFrun-time. (Refer to the [Working with OSGi - Bundles](#) section.)

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.

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 here: <http://www.w3.org/Addressing/URL/uri-spec.html>.

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 CatalogF framework). 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`.

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`.

For example:

```
if ( !uri.getScheme().equals("http") )  
{  
    throw new ResourceNotSupportedException("Unsupported scheme  
received, was expecting http")  
}
```

3. Implement the business logic.
4. 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();
```

The `Resource` needs to be accessible from the DDF installation. 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).

5. Return `Resource` in `ResourceResponse`.

For example:

```
        return ResourceResponseImpl( new ResourceImpl( new  
        BufferedInputStream( is ), new MimeType( mimeType ), url.getFile() )  
    );
```

6. 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 for others they 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.

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 by looking at 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" />
```

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.

Developing a Resource Writer

Before implementing a Resource Writer, refer to the Content Framework for alternatives.

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;
    }
}
```

2. Register the implementation as a Service in the OSGi Service Registry.

Blueprint Service Registration Example

```
...
<service ref="[[ResourceWriterReference]]"
interface="ddf.catalog.resource.ResourceWriter" />
...
```

3. Deploy the OSGi bundled packaged service to the DDF run-time (Refer to the [Working with OSGi - Bundles](#) section.)

ResourceWriter Javadoc

Refer to the DDF Catalog API Javadoc for more information about the methods required for implementing the interface.

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.

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.

See the [Developing a Resource Reader](#) section or the [Developing a Source](#) section in the Developer's Guide for more information and examples.

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 ResourceRe

ader 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.

Store Resource

```
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
```

See the [Developing a Resource Writer](#) section in the Developer's Guide for more information and examples.

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)
- URI Javadoc (<http://docs.oracle.com/javase/6/docs/api/java/net/URI.html>)

Developing Catalog Components

Overview

This section describes how to create Catalog components. Use in conjunction with the Javadoc to begin extending the DDF Catalog.

Table of Contents

Javadocs

On This Page

- Released APIs
- Beta APIs
- Private APIs

Released APIs

The following APIs and specifications will be released for developers to implement or use to their direction.

Catalog Core API

[zip](#)
[html](#)

Beta APIs

The following APIs are currently being improved and are considered to be in BETA state. They are available for review. Beta APIs may change in the future with no guarantee of backwards compatibility until the APIs have been released.

Content Core API

Action Core API

Mime Core API

Security Core API

Security Encryption API
[zip](#)

[zip](#)

[zip](#)

[zip](#)
[html](#)

[html](#)

[html](#)

[html](#)

[html](#)

Private APIs

The Metrics Framework and Metrics Endpoint are for internal use at this time. The endpoint is likely to change in future versions; therefore, any custom applications that are built to make use of the endpoint should be created with caution.

Securing DDF Catalog Application

Overview

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 covers implementations and protocols for enhancing security.

DDF Catalog Application Release Notes

Versions
<ul style="list-style-type: none">Release Version: catalog-2.6.1Release Version: catalog-2.5.0<ul style="list-style-type: none">Release Version: catalog-2.5.1Release Version: ddf-catalog-2.4.1

Release Version: catalog-2.6.1

On This Page
<ul style="list-style-type: none">New CapabilitiesResolved IssuesKnown IssuesApp PrerequisitesThird Party Licenses

New Capabilities

- Sources UI
- Catalog Backup Plugin

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
<p>⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</p>										

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
<p>⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</p>										

App Prerequisites

Before the DDF Catalog Application 2.6.1 can be installed:

- the DDF Kernel 2.6.1 must be running
- the DDF Platform App 2.6.1 must be running

Third Party Licenses

[Third Party License File Notice](#)

Release Version: catalog-2.5.0

On This Page

- New Capabilities
- Resolved Issues
- Known Issues
- App Prerequisites
- App Dependencies
- Third Party Licenses

New Capabilities

- Resource Caching
- Reliable Resource Retrieval
 - Notifications
 - Activities
 - Ability to Cancel
- Resource Retrieval is associated with a user

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

App Prerequisites

Before the DDF Catalog Application 2.5.0 can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.5.0 must be running

App Dependencies

```
ch.qos.logback:logback-classic:jar:0.9.24
com.codahale.metrics:metrics-core:jar:3.0.1
com.drewnoakes:metadata-extractor:jar:2.6.2
com.googlecode.json-simple:json-simple:jar:1.1.1
com.googlecode.pojosr:de.kalpatec.pojojr.framework:jar:0.1.8
com.sun.jersey:jersey-server:jar:1.5
com.vividsolutions:jts:jar:1.12
commons-codec:commons-codec:jar:1.7
commons-collections:commons-collections:jar:3.2.1
commons-io:commons-io:jar:2.1
commons-lang:commons-lang:jar:2.6
ddf.action.core:action-core-api:jar:2.3.0
ddf.action.core:action-core-impl:jar:2.3.0
ddf.metrics.collector:metrics-collector:jar:2.3.0
ddf.mime.core:mime-core-api:jar:2.3.0
ddf.platform:platform-configuration:jar:2.3.0
ddf.security.core:security-core-api:jar:2.3.0
javax.servlet:servlet-api:jar:2.5
javax.ws.rs:javax.ws.rs-api:jar:2.0-m10
joda-time:joda-time:jar:1.6.2
log4j:log4j:jar:1.2.17
net.minidev:json-smart:jar:1.1.1
net.sf.saxon:saxon-dom:jar:9.1.0.8
net.sf.saxon:saxon:jar:9.1.0.8
org.apache.abdera:abdera-extensions-geo:jar:1.1.3
org.apache.abdera:abdera-extensions-opensearch:jar:1.1.3
org.apache.camel:camel-core-osgi:jar:2.12.1
org.apache.camel:camel-core:jar:2.12.1
org.apache.commons:commons-compress:jar:1.4.1
org.apache.cxf:cxf-rt-frontend-jaxrs:jar:2.7.8
org.apache.cxf:cxf-rt-ws-security:jar:2.7.8
org.apache.karaf.shell:org.apache.karaf.shell.console:jar:2.3.3
org.apache.lucene:lucene-core:jar:3.0.2
org.apache.pdfbox:pdfbox:jar:1.7.1
org.apache.poi:poi-ooxml:jar:3.9
org.apache.poi:poi-scratchpad:jar:3.9
org.apache.poi:poi:jar:3.9
org.apache.shiro:shiro-core:jar:1.2.1
org.apache.tika:tika-bundle:jar:1.2
org.apache.tika:tika-core:jar:1.2
org.apache.ws.commons.axiom:axiom-api:jar:1.2.14
org.geotools.xsd:gt-xsd-gml3:jar:8.4
org.geotools:gt-jts-wrapper:jar:8.4
org.geotools:gt-main:jar:8.4
org.geotools:gt-opengis:jar:8.4
org.jscience:jscience:jar:4.3.1
org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.6.0
org.jvnet.ogc:gml-v_3_1_1-schema:jar:1.0.3
```

```
org.jvnet.ogc:ogc-tools-gml-jts:jar:1.0.3
org.ops4j.pax.swissbox:pax-swissbox-extender:jar:1.3.1
org.osgi:org.osgi.compendium:jar:4.3.1
org.osgi:org.osgi.core:jar:4.3.1
org.parboiled:parboiled-java:jar:1.1.3
org.rrd4j:rrd4j:jar:2.0.7
org.slf4j:slf4j-ext:jar:1.7.1
org.springframework.osgi:spring-osgi-core:jar:1.1.0
```

```
xalan:serializer:jar:2.7.1  
xalan:xalan:jar:2.7.1  
xerces:xercesImpl:jar:2.9.1
```

Third Party Licenses

[Third Party License File Notice](#)

Release Version: catalog-2.5.1

Contents

- Overview
 - Bug Fixes
- Known Issues
- App Prerequisites
- App Dependencies
- Third Party Licenses

Overview

This release contains bug fixes that were discovered in [ddf-catalog 2.5.0](#)

Bug Fixes

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

App Prerequisites

Before the DDF Catalog Application 2.5.1 can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.5.1 must be running

App Dependencies

```
ch.qos.logback:logback-classic:jar:0.9.24
com.codahale.metrics:metrics-core:jar:3.0.1
com.drewnoakes:metadata-extractor:jar:2.6.2
com.googlecode.json-simple:json-simple:jar:1.1.1
com.googlecode.pojosr:de.kalpatec.pojojr.framework:jar:0.1.8
com.sun.jersey:jersey-server:jar:1.5
com.vividsolutions:jts:jar:1.12
commons-codec:commons-codec:jar:1.7
commons-collections:commons-collections:jar:3.2.1
commons-io:commons-io:jar:2.1
commons-lang:commons-lang:jar:2.6
ddf.action.core:action-core-api:jar:2.3.0
ddf.action.core:action-core-impl:jar:2.3.0
ddf.metrics.collector:metrics-collector:jar:2.3.0
ddf.mime.core:mime-core-api:jar:2.3.0
ddf.platform:platform-configuration:jar:2.3.0
ddf.security.core:security-core-api:jar:2.3.0
javax.servlet:servlet-api:jar:2.5
javax.ws.rs:javax.ws.rs-api:jar:2.0-m10
joda-time:joda-time:jar:1.6.2
log4j:log4j:jar:1.2.17
net.minidev:json-smart:jar:1.1.1
net.sf.saxon:saxon-dom:jar:9.1.0.8
net.sf.saxon:saxon:jar:9.1.0.8
org.apache.abdera:abdera-extensions-geo:jar:1.1.3
org.apache.abdera:abdera-extensions-opensearch:jar:1.1.3
org.apache.camel:camel-core-osgi:jar:2.12.1
org.apache.camel:camel-core:jar:2.12.1
org.apache.commons:commons-compress:jar:1.4.1
org.apache.cxf:cxf-rt-frontend-jaxrs:jar:2.7.8
org.apache.cxf:cxf-rt-ws-security:jar:2.7.8
org.apache.karaf.shell:org.apache.karaf.shell.console:jar:2.3.3
org.apache.lucene:lucene-core:jar:3.0.2
org.apache.pdfbox:pdfbox:jar:1.7.1
org.apache.poi:poi-ooxml:jar:3.9
org.apache.poi:poi-scratchpad:jar:3.9
org.apache.poi:poi:jar:3.9
org.apache.shiro:shiro-core:jar:1.2.1
org.apache.tika:tika-bundle:jar:1.2
org.apache.tika:tika-core:jar:1.2
org.apache.ws.commons.axiom:axiom-api:jar:1.2.14
org.geotools.xsd:gt-xsd-gml3:jar:8.4
org.geotools:gt-jts-wrapper:jar:8.4
org.geotools:gt-main:jar:8.4
org.geotools:gt-opengis:jar:8.4
org.jscience:jscience:jar:4.3.1
org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.6.0
org.jvnet.ogc:gml-v_3_1_1-schema:jar:1.0.3
org.jvnet.ogc:ogc-tools-gml-jts:jar:1.0.3
org.ops4j.pax.swissbox:pax-swissbox-extender:jar:1.3.1
org.osgi:org.osgi.compendium:jar:4.3.1
org.osgi:org.osgi.core:jar:4.3.1
```

```
org.parboiled:parboiled-java:jar:1.1.3
org.rrd4j:rrd4j:jar:2.0.7
org.slf4j:slf4j-ext:jar:1.7.1
org.springframework.osgi:spring-osgi-core:jar:1.1.0
```

```
xalan:serializer:jar:2.7.1  
xalan:xalan:jar:2.7.1  
xerces:xercesImpl:jar:2.9.1
```

Third Party Licenses

[Third Party License File Notice](#)

Release Version: ddf-catalog-2.4.1

Contents

- [App Dependencies](#)

App Dependencies

```
abdera-client-1.1.3.jar
abdera-core-1.1.3.jar
abdera-extensions-geo-1.1.3.jar
abdera-extensions-opensearch-1.1.3.jar
abdera-i18n-1.1.3.jar
abdera-server-1.1.3.jar
axiom-api-1.2.10.jar
commons-codec-1.4.jar
commons-collections-3.2.1.jar
commons-httpclient-3.1.0_1.jar
geotools-suite-8.4_1.jar
gt-opengis-8.4_1.jar
guava-16.0.1.jar
jaxb2-basics-runtime-0.6.0.jar
jscience-4.3.1.jar
json-simple-1.1.1.jar
json-smart-1.1.1.jar
jts-1.12_1.jar
lucene-core-3.0.2_1.jar
metrics-collector-2.4.0.jar
metrics-core-3.0.1.jar
mime-core-api-2.4.0.jar
notifications-1.0.0.jar
ogc-tools-gml-jts-1.0.3.jar
org.apache.servicemix.bundles.abdera-parser-1.1.3_1.jar
org.apache.servicemix.bundles.axiom-impl-1.2.12-2.jar
org.apache.servicemix.bundles.dom4j-1.6.1_5.jar
org.apache.servicemix.bundles.jdom-1.1.2_1.jar
picocontainer-1.2_1.jar
rrd4j-2.0.7.jar
tika-bundle-1.2.jar
tika-core-1.2.jar
vecmath-1.3.2_1.jar
```

DDF Content Application

Overview

The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data.

DDF Content Application Table of Contents

- [Managing DDF Content Application — DDF Content Application _admin_guide_intro](#)
 - [Installing DDF Content Application](#)
- [Integrating DDF Content Application — DDF Content Application DDF Integrator's Guide](#)
 - [DDF Content Framework](#) — The DDF Content Framework is a framework for storing, reading, processing, and transforming content information. Content information is defined as files that the client wants parsed and created with a metocard that is subsequently used to create a catalog entry in the Metadata Catalog and stored in the DDF Content Repository.
 - [Content Framework Architecture](#)
 - [Content Component Types](#)
 - [Content Framework Execution Flow](#)
 - [DDF Content Core](#)
 - [DDF Content REST CRUD Endpoint](#) — allows clients to perform CRUD operations on DDF Content using REST, a simple architectural style, over HTTP
- [Extending DDF Content Application — _Content_App_Description](#)
- [Securing DDF Content Application](#)
- [DDF Content Application Release Notes](#) — This list compiles release notes for previous versions of the Content Application.
 - [Release Version: content-2.4.0](#)
 - [Release Version: content-2.4.2](#)

Managing DDF Content Application

Overview

The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data. This guide documents the installation, maintenance, and support of this application.

Installing DDF Content Application

Overview

This page describes:

- which applications must be installed prior to installing this application,
- how to install the DDF Content Application,
- how to verify if the DDF Content Application was successfully installed,
- how to uninstall the DDF Content Application,
- how to upgrade the DDF Content Application.

On This Page

- [Overview](#)
- [Installing](#)
- [Verifying](#)
- [Uninstalling](#)
 - [Reverting the Uninstall](#)
- [Upgrading](#)

Prerequisites

Before the DDF Content application can be installed, the following prerequisites must be met:

- the [DDF Kernel](#) must be running,
- the [DDF Platform Application](#) must be installed, and
- the [DDF Catalog Application](#) must be installed.

The Content application will continue to function properly as a content store without the Catalog application. However, the Content application will not support creating metacards for ingested content. In addition, without the Catalog application, the Content application will be displayed as FAILED by the [Platform Status Service](#) and the [Application Commands](#).

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.

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

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.

Uninstalling

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

```
features:removeurl -u <DDF application's feature repository URL>
```

Example: `features:removeurl -u mvn:ddf.catalog/catalog-app/2.3.0/xml/features`

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

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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf.catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

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.

```
features:install admin-app
```

4. Complete the steps in the Verify section above to determine if the upgrade was successful.

Integrating DDF Content Application

Overview

The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data. This guide provides instructions for configuring, maintaining and operation components of the Distributed Data Framework.

Table of Contents

- DDF Content Framework
- DDF Content Core
- DDF Content REST CRUD Endpoint

DDF Content Framework

Overview

The DDF Content Framework is a framework for storing, reading, processing, and transforming content information. Content information is defined as files that the client wants parsed and created with a metocard that is subsequently used to create a catalog entry in the Metadata Catalog and stored in the DDF Content Repository.

The files passed into the DDF Content Framework can be of any type, e.g., NITF, PDF, Microsoft Word, etc., as long as their mime type can be resolved, an [Input Transformer](#) exists to parse their content into a metocard, and the generated metocard satisfies the constraints of the catalog provider into which the generated metocard will be inserted. For example, if the [Tika Input Transformer](#) is installed, Microsoft Office documents and PDF files can be transformed into metacards. If the [Solr catalog provider](#) is being used, the generated metocard can be successfully inserted.

Clients typically perform create, read, update, and delete (CRUD) operations against the content repository. At the core of the Content functionality is the Content Framework, which routes all requests and responses through the system, invoking additional processing per the system configuration.

The DDF Content Framework has several components and an API that connects them together. The Content API consists of the Java interfaces that define DDF functionality. These interfaces provide the ability for components to interact without a dependency on a particular underlying implementation; therefore, 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 Content API interfaces, and reused by other DDF installations.

The DDF Content API will evolve with DDF itself, but great care is taken to retain backwards compatibility with developed components. Compatibility is reflected in version numbers. For more information, see the [Software Versioning](#) section in the [Appendix](#).

Content Framework Architecture

On This Page

- Architecture
- Design
 - DDF Content API Packaging
 - DDF Content API Reference Implementation

Architecture

 Unknown macro: 'plantuml'

Design

The DDF Content Framework design consists of several major components, namely endpoints, input transformers, the core Content Framework, storage providers, and Content plugins.

The dndpoints provide external clients access to the Content Framework. Input transformers convert incoming content into a metocard. The core Content Framework routes requests and responses through the system. The storage providers provide storage of the content to specific types of storage, e.g., file system, relational database, XML database. The Content plugins provide pluggable functionality that can be executed after the content has been stored/update/deleted but before the response has been returned to the client.

The UML class diagrams below illustrates how these components are related to each other.

 Unknown macro: 'plantuml'

DDF Content API Packaging

The Content API consists of the Java interfaces that define the methods that this API supports. The following illustrations show these interfaces and how they are packaged.

 Unknown macro: 'plantuml'

DDF Content API Reference Implementation

The Content API Reference Implementation consists of the Java classes that implement the methods defined in the Content API. The following illustrations show these classes and how they are packaged.

 Unknown macro: 'plantuml'

Content Component Types

- **Content Data Components** — domain objects representing the content to be stored
- **Content Endpoints** — components that accept external requests and interface with the internal components, storing content in the content repository and/or creating catalog entries
- **Content Framework** — the core of the Content application, routes requests and responses between the Content components
- **Content Operations** — represents all transactions that occur in the Content Framework, specifically requests and responses
- **Content Plugins** — process Content operations after content storage, generally to parse content's metadata and create a metocard from it to insert into the catalog provider
- **Storage Providers** — proxies to storage mechanisms used to store the content

Content Data Components

On This Page

- Content Item

 Unknown macro: 'plantuml'

Content Item

Content Item is the domain object, which is populated by the [Content Endpoint](#) from the client request, that represents the information about the content to be stored in the [Storage Provider](#). A Content Item encapsulates the content's globally unique ID, mime type, and input stream (i.e., the actual content).

Content Endpoints

On This Page

- Overview
- Examples

 Unknown macro: 'plantuml'

Overview

Content endpoints act as a proxy between the client and the Content Framework. Endpoints expose the client to the Content Framework.

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

- SOAP Web services

- RESTful services
- JMS
- RMI
- JSON
- OpenSearch

Content endpoints provide the capability to create, read, update, and delete content in the content repository, as well as create, update, and delete metacards corresponding to the content in the Metadata Catalog.

Endpoints are the only client-accessible components in DDF.

Examples

The following endpoints are provided with the Content Framework out of the box:

Content REST CRUD Endpoint

Content Framework

On This Page

- Overview
- Examples

 Unknown macro: 'plantuml'

Overview

The Content Framework wires all Content components together via OSGi and the Content API. It handles all Content operations requested by endpoints, invoking [Content Plugins](#) as needed, and for most [Operations](#), sending the request to a [Storage Provider](#) for execution.

Examples

The DDF Content comes with the following Content Frameworks out of the box:

Standard Content Framework

Content Operations

On This Page

- Overview

 Unknown macro: 'plantuml'

Overview

The DDF Content provides the capability to read, create, update, and delete content from the DDF Content Repository.

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 [Content Framework's](#) operation to communicate with the [Storage Providers](#) and [Content Plugins](#). The response is the output from the execution of the operation that is returned to the client and contains all of the data returned by the [Storage Providers](#) and [Content Plugins](#). For each operation, there is an associated request/response pair, e.g., the CreateRequest and CreateResponse pair for the [Content Framework's](#) create 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 Content API, helping to maintain backwards compatibility. Refer to the [Developer's Guide](#) for details on using this extensibility.

Content Plugins

On This Page

- Overview

 Unknown macro: 'plantuml'

Overview

The Content Framework calls Content plugins to process requests after they have been processed by the Storage Provider. If the request does not specify content storage (only processing), the Content Plugins are called immediately by the Content Framework.

Types of Content Plugins available out of the box:

[Content Cataloger Plugin](#)

Storage Providers

On This Page

- Overview
- Examples

 Unknown macro: 'plantuml'

Overview

Storage providers act as a proxy between the Content Framework and the mechanism storing the content, e.g., file system, relational database. Storage providers expose the storage mechanism to the Content Framework.

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

Examples

The following storage providers are provided with the Content Framework out of the box:

[File System Storage Provider](#)

Content Framework Execution Flow

On This Page

- Create Content and Catalog Entry from a GeoJson File

Create Content and Catalog Entry from a GeoJson File

The UML sequence diagram below illustrates how the DDF Content components interact when storing a GeoJson file and creating a catalog entry for it by parsing the GeoJson file and creating a Metocard.

 Unknown macro: 'plantuml'

DDF Content Core

Overview

The `content-core` bundle is a collection of default catalog components that can be used for most situations.

- Overview
- Standard Content Framework
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues
- Content Cataloger Plugin
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues
- Directory Monitor
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues
- File System Storage Provider
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues

Standard Content Framework

The Standard Content Framework provides the reference implementation of a [Content Framework](#) that implements all requirements of the Content API. `ContentFrameworkImpl` is the implementation of the Standard Content Framework.

Using

The Standard Content Framework is the core class of DDF Content. It provides the methods for read, create, update, and delete (CRUD) operations on the [Storage Provider](#).

Use this framework if:

- access to a storage provider to create, update, and delete content items in the DDF Content Repository is required or
- the ability to parse content, create a metocard, and then create, update, and delete catalog entries in the Metadata Catalog based on the parsed content are required.

Installing and Uninstalling

The Standard Content Framework is bundled in the `content-core` feature and is part of the `content-core-app`. It can be installed and uninstalled using the normal processes described in the [Configuration](#) section.

Configuring

There are no configuration properties for this component. This component can only be installed and uninstalled.

Known Issues

None

Content Cataloger Plugin

The Content Cataloger Plugin provides the implementation to parse content, create a Metocard, and create, update, and delete catalog entries in the Metadata Catalog.

The Content Cataloger Plugin is an implementation of the `ContentPlugin` interface. When installed, it is invoked by the [Content Framework](#) after a content item has been processed by the [Storage Provider](#) but before the response is returned to the [Content Endpoint](#).

The Content Cataloger Plugin searches the OSGi service registry for all services registered as `inputTransformers` that can process the content item's mime type. If such a service is found, the service is invoked (for `create` and `update` operations; `delete` operations are handled internally by the Content Cataloger Plugin). The `inputTransformer` service accepts the content item's `InputStream` and parses it, creating a Metocard that is returned to the Content Cataloger Plugin. This Metocard is then used in the `create` and `update` operations invoked on the [Integrating Catalog Framework](#) to interface with the Metadata Catalog.

Details on how to develop an Input Transformer with either Java or Apache Camel can be found in the [Developing an Input Transformer](#) section of [Extending Catalog Transformers](#).

Using

Use the Content Cataloger Plugin if create/update/delete of catalog entries in the Metadata Catalog based on the content item are desired. These CUD operations on the Metadata Catalog are made possible by parsing the content item to create a metocard and then using this metocard in the CUD operations on the [Catalog Framework](#). The Content Cataloger Plugin is the only component in the DDF Content Framework that has the ability to interface with the [Catalog Framework](#) (and hence the Metadata Catalog).

Installing and Uninstalling

The Content Cataloger Plugin is bundled as the `content-core-catalogerplugin` feature and can be installed and uninstalled using the normal processes described in the [Configuration](#) section of the Administrator's Guide.

Configuring

There are no configurable properties for this component. This component can only be installed and uninstalled.

Known Issues

Content Cataloger Plugin is only partially transactional. On create operations where the content is being stored in the content repository and the content is being parsed to generate a metocard for insertion into the Metadata Catalog, the content storage will be undone (i.e., the recently inserted content removed from the content repository if the Metadata Catalog insertion encountered problems.) Update and delete operations have no transactional capabilities. Once the content is updated or deleted this cannot be undone. Therefore, the content repository and Metadata Catalog could get out of sync.

Directory Monitor

The Content Directory Monitor allows files placed in a monitored directory to be ingested into the DDF Content Repository and/or the Metadata Catalog (MDC). A monitored directory is a directory configured to be polled by DDF periodically (typically every one second) for any new files added to the directory that should be ingested into the Content 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 Content Framework and processed based on whether the monitored directory's processing directive was:
 - configured to just store the file in the DDF Content Repository,
 - configured to just process the file's metadata and ingest it into the MDC, or
 - configured to both store the file in the Content Repository and ingest it into the MDC.
3. If the response from the Content 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 Content 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 Content 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 Content 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 Content Framework.
 - The Content 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 Content 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 Content 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 Content Framework.
 - The Content 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 Content 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 Content Framework.
 - The Content 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 Content 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 Content Framework.
 - The Content Framework generates a GUID for the create request for this ingested file.
 - The Content 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 Content 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.

Installing and Uninstalling

The Content Directory Monitor is packaged as the `content-core-directorymonitor` feature and is part of the `content-core-app`. It is installed by default.

It can be installed and uninstalled using the normal processes described in the Configuration section.

Note that the `content-core-catalogerplugin` feature must be installed for the metacards to be created and inserted into the MDC. This feature provides the linkage between the [Content Framework](#) and the [Integrating Catalog Framework](#). If the client attempts a STORE_AND_PROCESS or a PROCESS only without this feature installed a failure will be returned.

Configuring

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

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

Configuring Content Directory Monitors

Managed Service Factory PID:

```
ddf.content.core.directorymonitor.ContentDirectoryMonitor
```

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Directory Path	monitoredDirectoryPath	String	Specifies the directory to be monitored. Can be a fully-qualified directory or a relative path (which is relative to the DDF installation directory).	N/A	Yes

Processing Directive	directive	String	One of three possible values from a drop down box: <ul style="list-style-type: none"> • Store only - indicates to only store content in Content Repository • Process only - indicates to only create metocard and insert into MDC • Store and Process - do both 	Store and Process	Yes
Copy Files to Backup Directory	copyIngestedFiles	Boolean	Checking this option indicates that a backup of the file placed in the monitored directory should be made upon successful processing of the file. The file is moved into the <code>.ingested</code> sub-directory of the monitored directory.	False	No

Known Issues

None

File System Storage Provider

The File System Storage Provider is used to create/update/delete content items as files in the DDF Content Repository. The File System Storage Provider is an implementation of the Storage Provider interface. When installed, it is invoked by the [Content Framework](#) to create, update, or delete a file in the DDF Content Repository.

- For create operations, the File System Storage Provider (using the [MimeTypeMapper](#)) examines the mime type of the content item and determines the extension to use for the file to be stored. The File System Storage Provider also auto-generates a Globally Unique ID (GUID) for the content item. This GUID is used as the sub-directory for the content item's location in the Content Repository. This is to insure the files in the Content Repository are more evenly distributed rather than all being stored in one monolithic directory. The content is stored using the file name specified in the create request.

As an example, if the content item's mime type was `image/nitf`, then:

- the file extension would be `.nitf`,
- a GUID would be auto-generated (an example GUID would be `54947df8-0e9e-4471-a2f9-9af509fb5889`),
- the file name is specified in the create request (example: `myfile.nitf`), and
- the location in the Content Repository would be determined based on the GUID and the file name specified in the request (example: `54947df80e9e4471a2f99af509fb5889/myfile.nitf`).
- For read operations, the File System Storage Provider reads the content file with the GUID specified in the ReadRequest.
- For update operations, the File System Storage Provider updates the content file with the content item's new InputStream contents. The GUID of the content file to be updated is included in the UpdateRequest.
- For delete operations, the File System Storage Provider deletes the content file with the GUID specified in the DeleteRequest.

A sub-directory is created for each entry in the content store, so there will be limitations based on the file system that is used, i.e., the maximum amount of sub-directories supported for a file system.

Using

Use the File System Storage Provider if creating, reading, updating, and/or deleting contents in a file system is desired..

Installing and Uninstalling

The File System Storage Provider is packaged as the `content-core-filesystemstorageprovider` feature and can be installed and uninstalled using the normal processes described in the [Configuration](#) section. This feature is installed by default.

Configuring

The location used for content storage can be configured in the webconsole under **Configuration Content File System Storage Provider**.

Known Issues

None.

DDF Content REST CRUD Endpoint

On This Page

- Overview
- Usage
 - Interact with REST Endpoint
 - Create Request Multipart/Form-Data Parameters
 - Update and Delete Request HTTP Header Parameters
 - Sample Requests and Responses
 - cURL Commands
- Install and Uninstall
- Configuration
- Known Issues

Overview

The Content REST endpoint provides a CDR REST Retrieve v2.0-compliant DDF endpoint that allows clients to perform CRUD operations on the Content Repository using REST, a simple architectural style that performs communication using HTTP.

The URL exposing the REST functionality will be located at `http://<DDF_HOST>:<DDF_PORT>/services/content`, where `DDF_HOST` is the IP address of where DDF is installed and `DDF_PORT` is the port number on which DDF is listening.

The Content REST CRUD endpoint provides the capability to read, create, update, and delete content in the Content Repository, as well as create, update, and delete metacards in the catalog provider, i.e., the Metadata Catalog (MDC). Furthermore, this endpoint allows the client to perform the create/update/delete operations on just the Content Repository, just the MDC, or both in one operation.

The Content Framework is currently transactional for create operations only. Therefore, the client sends a create request to create content in the DDF Content Repository, processes the content to create a metocard, and ingests it into the MDC (i.e., `directive=STORE_AND_PROCESS`). If a problem is encountered during the catalog ingest, the content is removed from the DDF Content Repository, analogous to a rollback. This is so that the DDF Content Repository and the MDC are kept in sync.

The Content Framework does not support rollback capability for update or delete operations that affect both the DDF Content Repository and the MDC.

Usage

The Content REST CRUD endpoint provides the capability to read, create, update, and delete content in the DDF Content Repository as well as create, update, and delete metacards in the catalog provider as follows. Sample requests and responses

are provided in a separate table.

Operation	HTTP Request	Details	Example URL
Create Content and Catalog Entry	HTTP POST	<p>The multipart/form-data REST request that contains the binary data to be stored in the DDF Content Repository and to be parsed to create a metocard for ingest into the MDC. This binary data can be included in the request's body or as a file attachment.</p> <p>An HTTP 201 CREATED status code is returned to the client with:</p> <ul style="list-style-type: none">• Content-ID HTTP header set to GUID assigned to content item by Content Framework• Catalog-ID HTTP header set to the catalog ID assigned to the new catalog entry created based on the metocard generated from the parsed content• Content-URI HTTP header set to the resource URI for the content stored in the DDF Content Repository• Location URI HTTP header with URI containing the content ID	<code>http://<DDF_HOST>:<DDF_PORT>/services/content</code> Where the <code>directive</code> form parameter is set to <code>STORE_AND_PROCESS</code> , and the <code>file</code> form parameter that specifies the binary data with an optional <code>filename</code> parameter that should be stored as in the DDF Content Repository.

Create Content Only	HTTP POST	<p>The multipart/form-data REST request that contains the binary data to be stored in the DDF Content Repository. This binary data can be included in the request's body or as a file attachment.</p> <p>An HTTP 201 CREATED status code is returned to the client with:</p> <ul style="list-style-type: none"> Content-ID HTTP header set to GUID assigned to content item by Content Framework Location URI HTTP header with URI containing the content ID 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content</code></p> <p>Where the <code>directive</code> form parameter is set to <code>STORE</code>, and the <code>file</code> form parameter that specifies the binary data with an optional <code>filename</code> parameter that should be stored as in the DDF Content Repository.</p>
Create Catalog Entry Only	HTTP POST	<p>The multipart/form-data REST request that contains the binary data to be parsed to create a metocard for ingest into the MDC. This binary data can be included in the request's body or as a file attachment.</p> <p>An HTTP 200 OK status code is returned to the client with:</p> <ul style="list-style-type: none"> Catalog-ID HTTP header set to the catalog ID assigned to the new catalog entry created based on the metocard generated from the parsed content 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content</code></p> <p>Where the <code>directive</code> form parameter is set to <code>PROCESS</code>, the <code>contentUri</code> form parameter is set to the URI of content being processed, and the <code>file</code> form parameter specifying the binary data.</p>
Update Content and Catalog Entry	HTTP PUT	<p>The ID of the content item in the DDF Content Repository to be updated is appended to the end of the URL.</p> <p>The body of the REST request contains the binary data to update the DDF Content Repository.</p> <p>An HTTP 200 OK status code is returned to the client with:</p> <ul style="list-style-type: none"> Content-ID HTTP header set to GUID updated by the Content Framework Catalog-ID HTTP header set to the catalog ID that was updated in the MDC 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content/ABC123</code></p> <p>Where ABC123 is the ID of the content item to be updated, and the <code>directive</code> HTTP header parameter is set to <code>STORE_AND_PROCESS</code>.</p>
Update Content Only	HTTP PUT	<p>The ID of the content item in the DDF Content Repository to be updated is appended to the end of the URL.</p> <p>The body of the REST request contains the data to update the DDF Content Repository.</p> <p>An HTTP 200 OK status code is returned to the client with:</p> <ul style="list-style-type: none"> Content-ID HTTP header set to GUID updated by the Content Framework 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content/ABC123</code></p> <p>Where ABC123 is the ID of the content item to be updated, and the <code>directive</code> HTTP header parameter is set to <code>STORE</code>.</p>
Update Catalog Entry Only and Content ID is provided	HTTP PUT	<p>The ID of the content item in the DDF Content Repository to be updated is appended to the end of the URL.</p> <p>The body of the REST request contains the data to update the catalog entry in the MDC.</p> <p>An HTTP 200 OK status code is returned to the client with:</p> <ul style="list-style-type: none"> Catalog-ID HTTP header set to the catalog ID that was updated in the MDC Content-ID HTTP header set to GUID updated by the Content Framework 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content/ABC123</code></p> <p>Where ABC123 is the ID of the content item to be updated, and the <code>directive</code> HTTP header parameter is set to <code>STORE_AND_PROCESS</code>.</p>
Update Catalog Entry Only and Content URI is provided	HTTP PUT	<p>The URI of the content item in the MDC to be updated is specified in the <code>contentUri</code> HTTP header parameter.</p> <p>The body of the REST request contains the data to update the catalog entry in the MDC.</p> <p>An HTTP 200 OK status code is returned to the client with:</p> <ul style="list-style-type: none"> Catalog-ID HTTP header set to the catalog ID that was updated in the MDC 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content</code></p> <p>The <code>directive</code> is set to <code>PROCESS</code> in the Content REST Endpoint; it does not need to be explicitly set in the <code>directive</code> HTTP header parameter.</p>

Delete Content and Catalog Entry	HTTP DELETE	<p>The ID of the content item in the DDF Content Repository to be deleted is appended to the end of the URL.</p> <p>HTTP status code of 204 NO CONTENT is returned upon successful deletion.</p> <ul style="list-style-type: none"> Content-ID HTTP header set to GUID deleted by the Content Framework Catalog-ID HTTP header set to the catalog ID that was deleted from the MDC 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content/ABC123</code></p> <p>Where ABC123 is the ID of the content item to be deleted, and the <code>directive</code> HTTP header parameter is set to <code>STORE_AND_PROCESS</code>.</p>
Delete Content Only	HTTP DELETE	<p>The ID of the content item in the DDF Content Repository to be deleted is appended to the end of the URL.</p> <p>HTTP status code of 204 NO CONTENT is returned upon successful deletion.</p>	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content/ABC123</code></p> <p>Where ABC123 is the ID of the content item to be deleted, and the <code>directive</code> HTTP header parameter is set to <code>STORE</code>.</p>
Delete Catalog Entry Only	HTTP DELETE	<p>The URI of the content item in the MDC to be deleted is specified in the <code>contentUri</code> HTTP header parameter.</p> <p>HTTP status code of 204 NO CONTENT is returned to the client upon successful deletion with:</p> <ul style="list-style-type: none"> Catalog-ID HTTP header set to the catalog ID that was deleted from the MDC 	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content</code></p> <p>The <code>contentUri</code> HTTP header parameter is set to the URI of the catalog entry in the MDC to be deleted.</p>
Read	HTTP GET	<p>The ID of the content item in the DDF Content Repository to be retrieved is appended to the end of the URL.</p> <p>An HTTP 200 OK status code is returned upon successful read, and the contents of the retrieved content item are contained in the HTTP body.</p>	<p><code>http://<DDF_HOST>:<DDF_PORT>/services/content/ABC123</code></p> <p>Where ABC123 is the ID of the content item to be retrieved</p>

For all Content REST CRUD commands, only one content item ID is supported in the URL; i.e., bulk operations are not supported.

Interact with REST 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.).

Create Request Multipart/Form-Data Parameters

The `create` (HTTP POST) request is a multipart/form-data request, allowing the binary data (i.e., the content) to be either included in the request's body or attached as a file. This binary data is defined in a `Content-Disposition` part of the request where the `name` parameter is set to `file`, and the optional `filename` parameter indicates the name of the file that the content should be stored as.

Optional form parameters for the `create` request are the `directive` and `contentUri`. The `directive` form parameter's value can be either `STORE`, `PROCESS`, or `STORE_AND_PROCESS`, indicating if the content should be only stored in the Content Repository, only processed to generate a metocard and then ingested into the MDC, or both. The `directive` form parameter will default to `STORE_AND_PROCESS` if it is not specified.

The `contentUri` form parameter allows the client to specify the URI of a product stored remotely/externally (relative to DDF). This `contentUri` is used to indicate that the client will manage the content storage but wants the Content Framework to parse the content and create/update/delete a catalog entry in the MDC using this content URI as the entry's product URI. This parameter is used when the `directive` is set to `PROCESS`.

Update and Delete Request HTTP Header Parameters

Two optional HTTP header parameters are available on the update and delete RESTful URLs.

The `directive` header parameter allows the client to optionally direct the Content Framework to:

- only store the content in the DDF Content Repository (`directive=STORE`)
- store the content in the repository and parse the content to create a metocard (`directive=STORE_AND_PROCESS`); this metocard is then created/updated/deleted in the Metadata Catalog (by invoking the [Integrating Catalog Framework](#) operations)

`STORE_AND_PROCESS` is the default value for the `directive` header parameter. The `directive` header parameter is only used on the `PUT` and `DELETE` RESTful URLs that have a `contentId` in the URL.

The `contentUri` header parameter allows the client to specify the URI of a product stored remotely/externally (relative to DDF). The `contentUri` header parameter is only used with the `PUT` and `DELETE` RESTful URLs, where the `contentId` is not appended to the URL.

Sample Requests and Responses

The table below displays sample REST requests and their responses for each of the operations supported by the Content REST endpoint.

For the examples below, DDF was running on host DDF_HOST on port DDF_PORT. Also, for all examples below the binary data, i.e., the "content", is not included in the request's body.

Operation	Request	Response
Create Content and Catalog Entry	<pre>POST http://DDF_HOST:DDF_PORT/services/content/ HTTP/1.1 Content-Type: multipart/form-data; boundary=ARCFormBoundaryuxprlpjxmakbj4i --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="directive" STORE_AND_PROCESS --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="file"; filename="C:\DDF\geojson_valid.json" Content-Type: application/json;id=geojson <content included in payload but omitted here for brevity> --ARCFormBoundaryuxprlpjxmakbj4i--</pre>	<pre>HTTP/1.1 201 Created Catalog-ID: e82a31253e634a409c83d7164638fc Content-ID: ef0ef614bbdb4ede99e2371ebd2280 Content-Length: 0 Content-URI: content:ef0ef614bbdb4ede99e23 Date: Wed, 13 Feb 2013 21:56:15 GMT Location: http://127.0.0.1:8181/services/c 71ebd2280ee Server: Jetty(7.5.4.v20111024)</pre>
Create Content Only	<pre>POST http://DDF_HOST:DDF_PORT/services/content/ HTTP/1.1 Content-Type: multipart/form-data; boundary=ARCFormBoundaryuxprlpjxmakbj4i --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="directive" STORE --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="file"; filename="C:\DDF\geojson_valid.json" Content-Type: application/json;id=geojson <content included in payload but omitted here for brevity> --ARCFormBoundaryuxprlpjxmakbj4i--</pre>	<pre>HTTP/1.1 201 Created Content-ID: 7d671cd8e9aa4637960b37c7b3870a Content-Length: 0 Content-URI: content:7d671cd8e9aa4637960b3 Date: Wed, 13 Feb 2013 21:56:16 GMT Location: http://127.0.0.1:8181/services/c 7c7b3870aed Server: Jetty(7.5.4.v20111024)</pre>

Create Catalog Entry Only	<pre> POST http://DDF_HOST:DDF_PORT/services/content/ HTTP/1.1 Content-Type: multipart/form-data; boundary=ARCFormBoundaryuxprlpjxmakbj4i --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="directive" PROCESS --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="contentUri" http://localhost:8080/some/path/file.json --ARCFormBoundaryuxprlpjxmakbj4i Content-Disposition: form-data; name="file"; filename="C:\DDF\geojson_valid.json" Content-Type: application/json;id=geojson <content included in payload but omitted here for brevity> --ARCFormBoundaryuxprlpjxmakbj4i-- </pre>	HTTP/1.1 200 OK Catalog-ID: 94d8fae228a84e29a7396196542e26 Content-Length: 0 Date: Wed, 13 Feb 2013 21:56:16 GMT Server: Jetty(7.5.4.v20111024)
Update Content and Catalog Entry	<pre> PUT http://DDF_HOST:DDF_PORT/services/content/bf9763c2e74d46f68a9ed591c4b74591 HTTP/1.1 Accept-Encoding: gzip,deflate directive: STORE_AND_PROCESS Content-Type: application/json;id=geojson User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181 Content-Length: 9608 <content included in payload but omitted here for brevity> </pre>	HTTP/1.1 200 OK Catalog-ID: d9ccbc9d139a4abbb0b1cdded1de09 Content-ID: bf9763c2e74d46f68a9ed591c4b745 Content-Length: 0 Date: Wed, 13 Feb 2013 21:56:25 GMT Server: Jetty(7.5.4.v20111024)
Update Content Only	<pre> PUT http://DDF_HOST:DDF_PORT/services/content/bf9763c2e74d46f68a9ed591c4b74591 HTTP/1.1 Accept-Encoding: gzip,deflate directive: STORE Content-Type: application/json;id=geojson User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181 Content-Length: 9608 <content included in payload but omitted here for brevity> </pre>	HTTP/1.1 200 OK Content-ID: 7a702cd5c95347d2aa79ccc25b39e4 Content-Length: 0 Date: Wed, 13 Feb 2013 21:56:25 GMT Server: Jetty(7.5.4.v20111024)

Update Catalog Entry Only and Content ID is provided (STORE_AND_PROCESS)	<pre>PUT http://DDF_HOST:DDF_PORT/services/content/bf9763c2e74d46f68a9ed591c4b74591 HTTP/1.1 Accept-Encoding: gzip,deflate directive: STORE_AND_PROCESS Content-Type: application/json;id=geojson User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181 Content-Length: 9608 <content included in payload but omitted here for brevity></pre>	HTTP/1.1 200 OK Catalog-ID: 54a42215bf514322ba60bee97dab68 Content-ID: bf9763c2e74d46f68a9ed591c4b745 Content-Length: 0 Date: Wed, 11 Sep 2013 15:22:59 GMT Server: Jetty(7.6.8.v20121106)
Update Catalog Entry Only and Content URI is provided (PROCESS only)	<pre>PUT http://DDF_HOST:DDF_PORT/services/content/ HTTP/1.1 Accept-Encoding: gzip,deflate contentUri: http://DDF_HOST:DDF_PORT/some/path4/file.json Content-Type: application/json;id=geojson <content included in payload but omitted here for brevity></pre>	HTTP/1.1 200 OK Catalog-ID: b7a95aab99cd4318b8021eeef2715e Content-Length: 0 Date: Wed, 11 Sep 2013 15:23:01 GMT Server: Jetty(7.6.8.v20121106)
Delete Content and Catalog Entry	<pre>DELETE http://DDF_HOST:DDF_PORT/services/content/911e27aba723448ea420142b0e793d38 HTTP/1.1 Accept-Encoding: gzip,deflate directive: STORE_AND_PROCESS User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181</pre>	HTTP/1.1 204 No Content Catalog-ID: 5236910acbd14d97a786f1fa95d43c Content-ID: 911e27aba723448ea420142b0e793d38 Content-Length: 0 Date: Wed, 13 Feb 2013 21:56:31 GMT Server: Jetty(7.5.4.v20111024)
Delete Content Only	<pre>DELETE http://DDF_HOST:DDF_PORT/services/content/eb91c8ee225d4cddb4d9fbe2d9bf5dc HTTP/1.1 Accept-Encoding: gzip,deflate directive: STORE User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181</pre>	HTTP/1.1 204 No Content Content-ID: eb91c8ee225d4cddb4d9fbe2d9bf5dc Content-Length: 0 Date: Wed, 13 Feb 2013 21:56:31 GMT Server: Jetty(7.5.4.v20111024)
Delete Catalog Entry Only	<pre>DELETE http://DDF_HOST:DDF_PORT/services/content/ HTTP/1.1 Accept-Encoding: gzip,deflate contentUri:http://DDF_HOST:DDF_PORT/some/path5/file.json User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181</pre>	HTTP/1.1 204 No Content Catalog-ID: c9a2b1c395f74300b33529483f0951 Content-Length: 0 Date: Wed, 13 Feb 2013 21:56:31 GMT Server: Jetty(7.5.4.v20111024)
Read	<pre>GET http://DDF_HOST:DDF_PORT/services/content/d34fd2b31f314aa6ade162015ba3016f HTTP/1.1 Accept-Encoding: gzip,deflate User-Agent: Jakarta Commons-HttpClient/3.1 Host: 127.0.0.1:8181</pre>	HTTP/1.1 200 OK Content-Length: 9579 Content-Type: application/json;id=geojson Date: Wed, 13 Feb 2013 21:56:24 GMT Server: Jetty(7.5.4.v20111024) ... (remaining data of content item retrieved omitted for brevity)

cURL Commands

The table below illustrates sample cURL commands corresponding to a few of the above REST requests. Pay special attention to the flags, as they vary between operations.

For these examples, DDF was running on host DDF_HOST on port DDF_PORT. We ingested/updated a file named geojson_valid.json whose MIME type was application/json;id=geojson, and whose content ID ended up being CONTENT_ID.

To perform each operation without using the catalog, replace STORE_AND_PROCESS with STORE . To manipulate the catalog entry only, replace STORE_AND_PROCESS with PROCESS.

Operation	Command
Create Content and Catalog Entry	<code>curl -i -X POST -F "directive=STORE_AND_PROCESS" -F "filename=geojson_valid.json" -F "file=@geojson_valid.json;type=application/json;id=geojson" http://DDF_HOST:DDF_PORT/services/content/</code>
Update Content and Catalog Entry	<code>curl -i -X PUT -H "directive: STORE_AND_PROCESS" -H "Content-Type: application/json;id=geojson" --data-binary "@geojson_valid.json" http://DDF_HOST:DDF_PORT/services/content/CONTENT_ID</code>
Delete Content and Catalog Entry	<code>curl -i -X DELETE -H "directive: STORE_AND_PROCESS" http://DDF_HOST:DDF_PORT/services/content/CONTENT_ID</code>
Read	<code>curl -i -X GET http://DDF_HOST: DDF_PORT/services/content/CONTENT_ID</code>

Install and Uninstall

The Content REST CRUD endpoint, packaged as the content-rest-endpoint feature, can be installed and uninstalled using the normal processes described in the [Configuring DDF](#) section.

Configuration

The Content REST CRUD endpoint has no configurable properties. It can only be installed or uninstalled.

Known Issues

None

Extending DDF Content Application

Overview

The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data.

This guide supports developers creating extensions of the existing framework. There are currently no DDF Content development details beyond those covered in the [DDF Developer's Guide](#).

Whitelist

The following packages have been exported by the DDF Content Application and are approved for use by third parties:

- ddf.content
- ddf.content.data
- ddf.content.operation
- ddf.content.plugin
- ddf.content.storage
- ddf.content.util
- ddf.content.core.directorymonitor

Table of Contents

Securing DDF Content Application

Overview

The DDF Content application provides a framework for storing, reading, processing, transforming and cataloging data.

This guide covers implementations and protocols for enhancing security.

Table of Contents

DDF Content Application Release Notes

Overview

This list compiles release notes for previous versions of the Content Application.

Versions

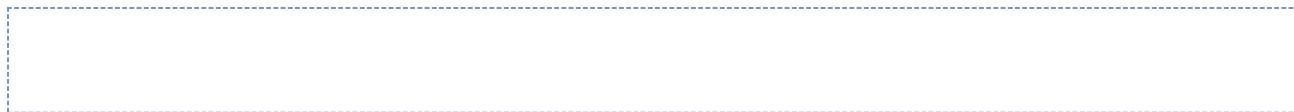
- Release Version: content-2.4.2
- Release Version: content-2.4.0

Release Version: content-2.4.0

Contents

- App Dependencies

App Dependencies



Release Version: content-2.4.2

Contents

- Resolved Issues
- Known Issues

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

DDF Platform Application

Overview

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. These building blocks include subsets of:

- Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
- Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
- Camel (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

Documentation

DDF Platform Application Table of Contents

- [Managing DDF Platform Application](#) — provides the fundamental building blocks that the DDF distribution needs in order to run, including subsets of Karaf, CXF, Cellar, and Camel
 - [Installing DDF Platform Application](#)
- [Integrating DDF Platform Application](#) — DDF Platform Application
 - [Platform Global Settings](#) — the system-wide configuration settings used throughout DDF
 - [DDF Mime Framework](#)
 - [Metrics Collection](#) — collects metric data periodically for pre-configured metrics and dynamically configured source metrics
 - [Metrics Reporting Application](#) — allows clients to retrieve metrics historical data graphed over a time range
 - [Security Core API](#)
 - [Compression Services](#)
- [Extending DDF Platform Application](#) — DDF Platform Application
 - [Developing Action Components \(Action Framework\)](#) — describes how and why to create Action Components
- [Securing DDF Platform Application](#) — DDF Platform Application
- [DDF Platform Application Release Notes](#) — This page lists Release notes for various versions of ApplicationName.
 - Release Version: platform-2.3.1
 - Release Version: ddf-platform-2.4.0
 - Release Version: platform-2.5.0
 - Release Version: platform-2.5.1
 - Release Version: platform-2.6.1

Managing DDF Platform Application

On This Page

- [Overview](#)
- [Usage](#)
- [Install and Uninstall](#)
- [Configuration](#)
 - [Configurable Properties](#)

Overview

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. These building blocks include subsets of:

- Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
- Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
- Camel (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

Usage

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 that is included in the Platform application should be used by those that need or like the convenience of a "platform independent" method of running certain commands, such as backing up data or logging settings. More information can be found on the [Command Scheduler](#) page.

Install and Uninstall

Refer to the [DDF Platform Application Install/Uninstall](#) page for how this application can be installed and uninstalled.

Configuration

This component can be configured using the normal processes described in the [Configuring DDF](#) section. The configurable properties are accessed from the **Schedule Command** Configuration in the Admin Console.

Configurable Properties

Property	Type	Description	Default Value	Required
command	String	Shell command to be used within the container. For example, <code>log:set DEBUG</code> .		yes
intervalInSeconds	Integer	The interval of time in seconds between each execution. This must be a positive integer. For example, 3600 is 1 hour.		yes

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 anytime with future releases of the Platform application. The exact versions of the third party applications that are used can be found in the [Release Notes](#) for the Platform application.

Installing DDF Platform Application

Overview

On This Page

- [Overview](#)
- [Prerequisites](#)
- [Installing](#)
- [Verifying](#)
- [Uninstalling](#)
 - [Reverting the Uninstall](#)
- [Upgrading](#)

Prerequisites

Before the DDF Platform application can be installed:

- the [DDF Kernel](#) must be running.

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.

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

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.

Uninstalling

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

```
features:removeurl -u <DDF application's feature repository URL>

Example:   features:removeurl -u
          mvn:ddf.platform/platform-app/2.4.0/xml/features
```

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

The repository URLs for installed applications can be obtained by entering:

```
features:listrepositories -u
```

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., platform-app.

Reverting DDF application's uninstall

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf .platform/platform-app/2.4.0/xml/features
ddf@local>features:install platform-app
```

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 platform-app-X.Y.Z..kar file to the <INSTALL_DIRECTORY>/deploy directory.
3. Start the application.

```
features:install platform-app
```

4. Complete the steps in the Verifying section above to determine if the upgrade was successful.

Integrating DDF Platform Application

Overview

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. These building blocks include subsets of:

- Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
- Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
- Camel (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

This guide supports integration of this application with external frameworks.

Table of Contents

- Platform Global Settings
- DDF Mime Framework
- Metrics Collection
- Metrics Reporting Application
- Security Core API
- Compression Services

Platform Global Settings

On This Page

- Overview
- Configuration
 - Configurable Properties

Overview

The Platform Global Settings are the system-wide configuration settings used throughout DDF to specify the information about the machine

hosting DDF.

Configuration

Configuration can be performed using the processes described in the [Configuring DDF](#) section. The configurable properties for the platform-wide configuration are accessed from **Configuration Platform Global Configuration** in the Web Console.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Protocol	protocol	String	Default protocol that should be used to connect to this machine.	http	yes
Host	host	String	The host name or IP address of the machine that DDF is running on. Do not enter localhost.		yes
Port	port	String	The port that DDF is running on.		yes
Site Name	id	String	The site name for this DDF instance.	ddf.distribution	yes
Version	version	String	The version of DDF that is running. This value should not be changed from the factory default.	DDF 2.3.0	yes
Organization	organization	String	The organization responsible for this installation of DDF	Codice Foundation	yes

DDF Mime Framework

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 Content Framework to read a file from the content file repository.

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

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

Included Mime Type Mappers

DDF Mime Type Mapper

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 part of the `mime-core-app` application. This feature can be installed and uninstalled using the normal processes described in the [Configuration](#) section.

The `mime-core` feature is installed by default.

Configuring

There is no configuration for this feature.

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 MimeMapper 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 [tika-mimetypes.xml](#) file that Apache Tika uses to define all of its default mime types that it supports is attached to this page.)

The TikaMimeTypeResolver is assigned a default priority of -1 to insure that it is always invoked last by the MimeMapper. 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 can be installed and uninstalled using the normal processes described in the [Configuring DDF](#) section.

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>		tika-mimetypes.xml

Custom Mime Type Resolver

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

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

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

New custom mime type resolver mappings can be added using the Web 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 that are not supported by DDF out of the box need to be added. 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. This feature can be installed and uninstalled using the normal processes described in the [Configuration](#) section.

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

Configuring

This component can be configured using the normal processes described in the [Configuring DDF](#) section.

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

Managed Service Factory PID

DDF_Custom_Mime_Type_Resolver

Configurable Properties

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

Implementation Details

Imported Services

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

Exported Services

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

Metrics Collection

On This Page

- Overview
- Source Metrics
- Usage
- Install and Uninstall
- Configuration
- Known Issues

Overview

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.

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

permanently lost.

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

The metrics currently being collected by DDF are:

Metric	JMX MBean Name	MBean Attribute Name	Description
Catalog Exceptions	ddf.metrics.catalog:name=Exceptions	Count	A count of the total number of exceptions, of all types, thrown across all catalog queries executed.
Catalog Exceptions Federation	ddf.metrics.catalog:name=Exceptions.Federation	Count	A count of the total number of Federation exceptions thrown across all catalog queries executed.
Catalog Exceptions Source Unavailable	ddf.metrics.catalog:name=Exceptions.SourceUnavailable	Count	A count of the total number of SourceUnavailable exceptions thrown across all catalog queries executed. These exceptions occur when the source being queried is currently not available.
Catalog Exceptions Unsupported Query	ddf.metrics.catalog:name=Exceptions.UnsupportedQuery	Count	A count of the total number of UnsupportedQuery exceptions thrown across all catalog queries executed. These exceptions occur when the query being executed is not supported or is invalid.
Catalog Ingest Created	ddf.metrics.catalog:name=Ingest.Created	Count	A count of the number of catalog entries created in the Metadata Catalog.
Catalog Ingest Deleted	ddf.metrics.catalog:name=Ingest.Deleted	Count	A count of the number of catalog entries updated in the Metadata Catalog.
Catalog Ingest Updated	ddf.metrics.catalog:name=Ingest.Updated	Count	A count of the number of catalog entries deleted from the Metadata Catalog.
Catalog Queries	ddf.metrics.catalog:name=Queries	Count	A count of the number of queries attempted.
Catalog Queries Comparison	ddf.metrics.catalog:name=Queries.Comparison	Count	A count of the number of queries attempted that included a string comparison criteria as part of the search criteria, e.g., PropertyIsLike, PropertyIsEqualTo, etc.
Catalog Queries Federated	ddf.metrics.catalog:name=Queries.Federated	Count	A count of the number of federated queries attempted.
Catalog Queries Fuzzy	ddf.metrics.catalog:name=Queries.Fuzzy	Count	A count of the number of queries attempted that included a string comparison criteria with fuzzy searching enabled as part of the search criteria.
Catalog Queries Spatial	ddf.metrics.catalog:name=Queries.Spatial	Count	A count of the number of queries attempted that included a spatial criteria as part of the search criteria.
Catalog Queries Temporal	ddf.metrics.catalog:name=Queries.Temporal	Count	A count of the number of queries attempted that included a temporal criteria as part of the search criteria.
Catalog Queries Total Results	ddf.metrics.catalog:name=Queries.TotalResults	Mean	An average of the total number of results returned from executed queries. This total results data is averaged over the metric's sample rate.
Catalog Queries Xpath	ddf.metrics.catalog:name=Queries.Xpath	Count	A count of the number of queries attempted that included a Xpath criteria as part of the search criteria.
Catalog Resource Retrieval	ddf.metrics.catalog:name=Resource	Count	A count of the number of products retrieved.
Services Latency	ddf.metrics.services:name=Latency	Mean	The response time (in milliseconds) from receipt of the request at the endpoint until the response is about to be sent to the client from the endpoint. This response time data is averaged over the metric's sample rate.

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 Web 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 <sourceld> Exceptions	<code>ddf.metrics.catalog.source:name=<sourceld>.Exceptions</code>	Count	A count of the total number of exceptions, of all types, thrown from catalog queries executed on this source.
Source <sourceld> Queries	<code>ddf.metrics.catalog.source:name=<sourceld>.Queries</code>	Count	A count of the number of queries attempted on this source.
Source <sourceld> Queries Total Results	<code>ddf.metrics.catalog.source:name=<sourceld>.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.

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.

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.

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.

Known Issues

None

Metrics Reporting Application

On This Page

- Overview
- Usage
 - Metric Data Supported Formats
 - Metrics Aggregate Reports
 - Add Custom Metrics to the Metrics Tab
- Install and Uninstall
- Configuration
- Known Issues

Overview

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.

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 all measured from the time that the hyperlink is selected. They are 15 minutes, 1 hour, 1 day, 1 week, 1 month, 3 months, 6 months, and 1 year.

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.

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

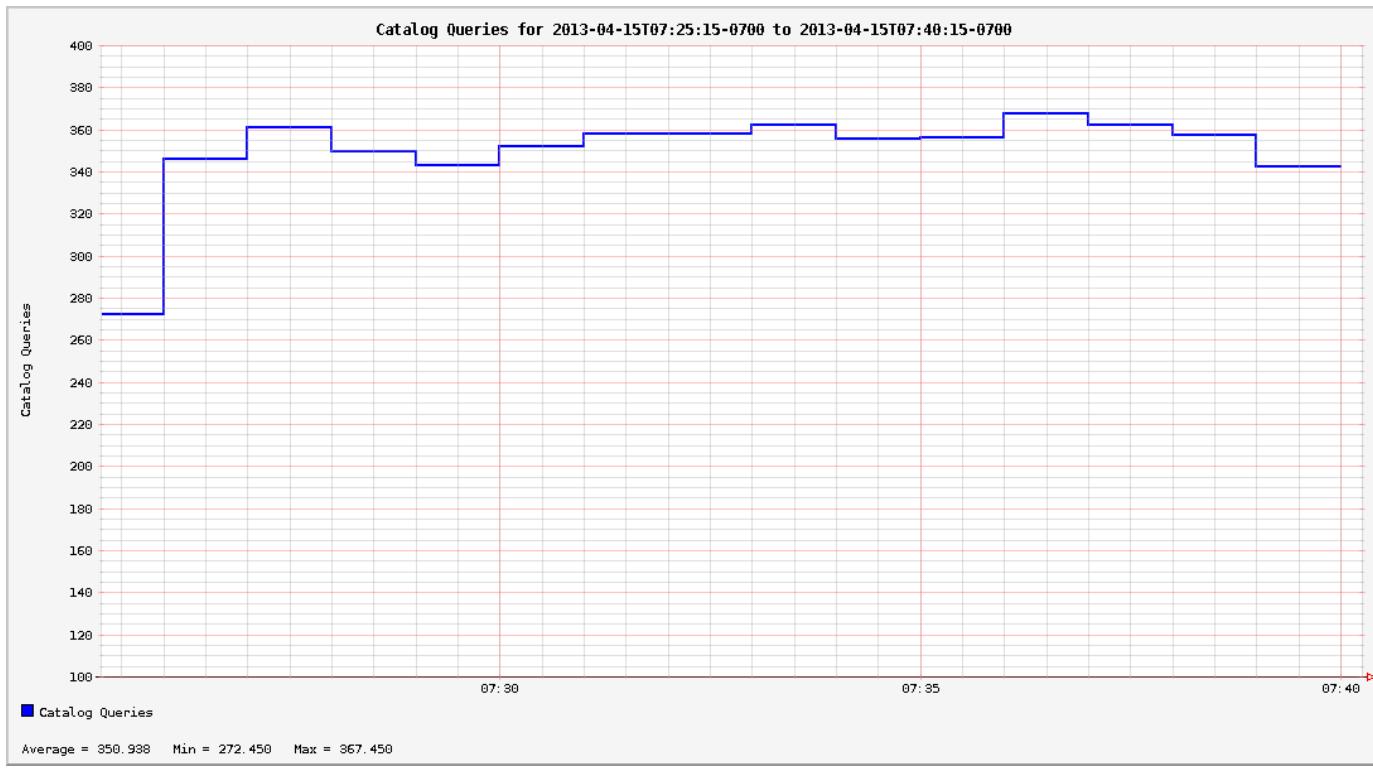
There is a hyperlink per format in which the metric's historical data can be displayed. For example, the PNG hyperlink for 15m for the Catalog Queries metric maps to http://<DDF_HOST>:<DDF_PORT>/services/internal/metrics/catalogQueries.png?dateOffset=900, where the dateOffset=900 indicates the previous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

Metrics

	Admin	Bundles	Configuration	Configuration Status	Events	Features	Gogo	Licenses	Log Service	Metrics	Services	System Information						
Metric	15m			1h			1d			1w			1M	3M	6M	1y		
Catalog Exceptions	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Exceptions Federation	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Exceptions Source Unavailable	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Exceptions Unsupported Query	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Ingest Created	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Ingest Deleted	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Ingest Updated	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Comparison	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Federated	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Fuzzy	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Spatial	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Temporal	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Total Results	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Queries Xpath	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Catalog Resource Retrieval	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Services Latency	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS	PNG	CSV	XLS
Weekly Reports																		
9 September 2013 - 15 September 2013	XLS			PPT														
2 September 2013 - 8 September 2013	XLS			PPT														
26 August 2013 - 1 September 2013	XLS			PPT														
19 August 2013 - 25 August 2013	XLS			PPT														
Monthly Reports																		
1 August 2013 - 31 August 2013	XLS			PPT														
1 July 2013 - 31 July 2013	XLS			PPT														
1 June 2013 - 30 June 2013	XLS			PPT														
1 May 2013 - 31 May 2013	XLS			PPT														
1 April 2013 - 30 April 2013	XLS			PPT														
1 March 2013 - 31 March 2013	XLS			PPT														
1 February 2013 - 28 February 2013	XLS			PPT														
1 January 2013 - 31 January 2013	XLS			PPT														
1 December 2012 - 31 December 2012	XLS			PPT														
1 November 2012 - 30 November 2012	XLS			PPT														
1 October 2012 - 31 October 2012	XLS			PPT														
1 September 2012 - 30 September 2012	XLS			PPT														
Yearly Reports																		
2012	XLS			PPT														

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.

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?startDat=2013-03-31T06:00:00-07:00&endDate=2013-04-01T11:00:00-07:00
```

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

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

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

Parameter	Description	Example
startDate	Specifies the start of the time range of the search on the metric's data (RFC-3339 - Date and Time format, i.e. YYYY-MM-DDTHH:mm:ssZ). Date/time must be earlier than the endDate. <i>This parameter cannot be used with the dateOffset parameter.</i>	startDate=2013-03-31T06:00:00-07:00
endDate	Specifies the endof the time range of the search on the metric's data (RFC-3339 - Date and Time format, i.e. YYYY-MM-DDTHH:mm:ssZ). Date/time must be later than the startDate. <i>This parameter cannot be used with the dateOffset parameter.</i>	endDate=2013-04-01T11:00:00-07:00
dateOffset	Specifies an offset, backwards from the current time, to search on the modified time field for entries. Defined in seconds and must be a positive Integer. <i>This parameter cannot be used with the startDate or endDate parameters.</i>	dateOffset=1800
yAxisLabel	(optional) the label to apply to the graph's y-axis. Will default to the metric's name, e.g., Catalog Queries. <i>This parameter is only applicable for the metric's graph display format.</i>	Catalog Query Count
title	(optional) the title to be applied to the graph. Will default to the metric's name plus the time range used for the graph. <i>This parameter is only applicable for the metric's graph display format.</i>	Catalog Query Count for the last 15 minutes

Metric Data Supported Formats

The metric's historical data can be displayed in several formats, including the PNG format previously mentioned, 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	<p>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.</p>	<p>Via hyperlink on the Metrics tab or directly via custom URL.</p>	<p>Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds): <code>/services/internal/metrics/catalogQueries.png?dateOffset=28800&yAxisLabel=my%20label&title=my%20graph%20title</code></p> <p>Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013: <code>/services/internal/metrics/catalogQueries.png?</code> <code>startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00&</code> <code>yAxisLabel=my%20label&title=my%20graph%20title</code></p> <p><i>Note that the <code>yAxisLabel</code> and <code>title</code> parameters are optional.</i></p>
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>	<p>Via hyperlink on the Metrics tab or directly via custom URL.</p>	<p>Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds): <code>/services/internal/metrics/catalogQueries.csv?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: <code>/services/internal/metrics/catalogQueries.csv?</code> <code>startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00</code></p>
XLS	<p>Displays the metric's data as an Excel (XLS) file, which can be auto-displayed in Excel based on browser settings.</p> <p>The generated XLS file will consist of:</p> <ul style="list-style-type: none"> • 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>	<p>Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds): <code>/services/internal/metrics/catalogQueries.xls?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: <code>/services/internal/metrics/catalogQueries.xls?</code> <code>startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00</code></p>
PPT	<p>Displays the metric's data as a PowerPoint (PPT) file, which can be auto-displayed in PowerPoint based on browser settings.</p> <p>The generated PPT file will consist of a single slide containing:</p> <ul style="list-style-type: none"> • 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 	<p>via custom URL only</p>	<p>Accessing Catalog Queries metric data for last 8 hours (8 * 60 * 60 = 28800 seconds): <code>/services/internal/metrics/catalogQueries.ppt?dateOffset=28800</code></p> <p>Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013: <code>/services/internal/metrics/catalogQueries.ppt?</code> <code>startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00</code></p>

XML	<p>Displays the metric's data as an XML-formatted file.</p>	<p>via custom URL only</p>	<p>Accessing Catalog Queries metric data for last 8 hours ($8 * 60 * 60 = 28800$ seconds): <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: <code>/services/internal/metrics/catalogQueries.xml? startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00</code></p> <p>Sample XML-formatted output would look like:</p> <pre data-bbox="698 424 1454 1115"> <catalogQueries> <title>Catalog Queries for Apr 15 2013 08:45:53 to Apr 15 2013 09:00:53</title> <data> <sample> <timestamp>Apr 15 2013 08:45:00</timestamp> <value>361</value> </sample> <sample> <timestamp>Apr 15 2013 09:00:00</timestamp> <value>353</value> </sample> <totalCount>5721</totalCount> </data> </catalogQueries></pre>
-----	---	------------------------------------	--

<p>JSON</p> <p>Displays the metric's data as an JSON-formatted file.</p>	<p>via custom URL only</p> <p>Accessing Catalog Queries metric data for last 8 hours ($8 * 60 * 60 = 28800$ seconds): <code>/services/internal/metrics/catalogQueries.json?dateOffset=28800</code></p> <p>Accessing Catalog Queries metric data between 6:00 am on March 10, 2013, and 10:00 am on April 2, 2013: <code>/services/internal/metrics/catalogQueries.json?startDate=2013-03-10T06:00:00-07:00&endDate=2013-04-02T10:00:00-07:00</code></p> <p>Sample JSON-formatted output would look like:</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>
---	---

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

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`.

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.

Install and Uninstall

The Metrics Reporting application can be installed and uninstalled using the normal processes described in the [Configuring DDF](#) section.

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.

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.

Security Core API

On This Page

- Overview
- Configuration
- Install and Uninstall
- Implementation Details
 - Imported Services
 - Exported Services

Overview

The Security Core API contains all of the DDF Security Framework APIs that are used to perform security operations within DDF. More information on the APIs can be found on the [Managing Web Service Security](#) page.

Configuration

None

Install and Uninstall

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

Implementation Details

Imported Services

None

Exported Services

None

Compression Services

On This Page

- Overview
- Configuration
- Install and Uninstall
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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

Configuration

None

Install and Uninstall

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

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

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

Compression Type	Description
exi	Adds Efficient XML Interchange (EXI) support to outgoing responses. EXI is an W3C standard for XML encoding that shrinks xml to a smaller size than normal GZip compression. More information is available at http://www.w3.org/XML/EXI/
gzip	Adds GZip compression to in and outgoing messages through CXF components. Code comes with CXF.

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.

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

Extending DDF Platform Application

Overview

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. These building blocks include subsets of:

- Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
- Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
- Camel (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

This guide supports developers creating extensions of the existing framework.

Whitelist

The following packages have been exported by the DDF Platform application and are approved for use by third parties:

- ddf.action
- ddf.action.impl
- ddf.mime
- ddf.security
- ddf.security.assertion
- ddf.security.common.audit
- ddf.security.permission
- ddf.security.service
- [ddf.security.ws.policy](#)
- [ddf.security.ws.proxy](#)
- ddf.security.encryption
- org.codice.ddf.configuration
- org.codice.ddf.platform.status

Table of Contents

- [Developing Action Components \(Action Framework\)](#)

Developing Action Components (Action Framework)

On This Page

- Overview
- Usage
 - Naming Convention
 - Taxonomy
 - Action ID Service Descriptions

Overview

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

Usage

To provide a service, such as a link to a record, the `ActionProvider` interface should be implemented. An `ActionProvider` essentially provides an `Action` 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 `null`. 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.

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.

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
<code>catalog.data.metocard.view</code>	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 ...
<code>catalog.data.metocard.thumbnail</code>	Provides a valid URL to the bytes of a thumbnail (Metocard.THUMBNAIL) with MIME type <code>image/jpeg</code> .	Get Thumbnail
<code>catalog.data.metocard.html</code>	Provides a valid URL that, when invoked, provides an HTML representation of the metocard.	Export as ...
<code>catalog.data.metocard.resource</code>	Provides a valid URL that, when invoked, provides the underlying resource of the metocard.	Get Resource
<code>catalog.data.metocard.metadata</code>	Provides a valid URL to the XML metadata in the metocard (Metocard.METADATA).	Get Metadata

Securing DDF Platform Application

Overview

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. These building blocks include subsets of:

- Karaf (<http://karaf.apache.org/>), CXF (<http://cxf.apache.org/>),
- Cellar (<http://karaf.apache.org/index/subprojects/cellar.html>), and
- Camel (<http://camel.apache.org/>).

Included as part of the Platform application is also a Command Scheduler. The Command Scheduler allows users to schedule [Command Line Shell Commands](#) to run at certain specified intervals.

This guide covers implementations and protocols for enhancing security.

Table of Contents

DDF Platform Application Release Notes

Overview

This page lists Release notes for various versions of DDF.

Versions

- Release Version: platform-2.6.1
- Release Version: platform-2.5.1
- Release Version: platform-2.5.0
- Release Version: ddf-platform-2.4.0
- Release Version: platform-2.3.1

Release Version: platform-2.3.1

Contents

- New Capabilities
- Resolved Issues
- Known Issues
- Included Features
- App Prerequisites
- App Dependencies
- Third Party Licenses

New Capabilities

DDF Platform Application Initial Release

- Added support for accessing metrics tab in admin console via SSL

Resolved Issues

Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due
------	-----	---------	----------	----------	----------	--------	------------	---------	---------	-----

 JIRA project doesn't exist or you don't have permission to view it.

[View these issues in JIRA](#)

Known Issues

Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due
------	-----	---------	----------	----------	----------	--------	------------	---------	---------	-----



JIRA project doesn't exist or you don't have permission to view it.

[View these issues in JIRA](#)

Included Features

Feature Name	Bundles Installed	Description	Installed by Default
action-core-api	action-core-api	DDF Action API	✓
action-core-impl	action-core-impl	DDF Action Core	✓
camel			✓
camel-blueprint			✓
camel-core			✓
camel-freemarker			
camel-http			
camel-jetty			
camel-jms			
camel-spring			✓
camel-test			
cellar		Karaf clustering	
cellar-bundle		Bundle cluster support	
cellar-config		ConfigAdmin cluster support	
cellar-core		Karaf clustering core	
cellar-dosgi		DOSGi support	
cellar-event		OSGi events broadcasting in clusters	
cellar-features		Karaf features cluster support	
cellar-hazelcast		Cellar implementation based on Hazelcast	
cellar-management		Cellar management	
cellar-obr		OBR cluster support	
cellar-shell		Cellar shell commands	
cellar-webconsole		Cellar plugin for Karaf WebConsole	
cxf			✓
cxf-abdera			
cxf-bindings-coloc			✓
cxf-bindings-corba			✓
cxf-bindings-object			✓
cxf-bindings-soap			
cxf-core			✓

cxf-databinding-aegis			
cxf-databinding-jaxb			
cxf-databinding-jibx			
cxf-databinding-xmlbeans			
cxf-features-clustering			
cxf-frontend-javascript			
cxf-http			
cxf-http-async			
cxf-http-jetty			
cxf-javascript			
cxf-jaxb			
cxf-jaxrs			
cxf-jaxws			
cxf-rs-security-cors			
cxf-rs-security-sso-saml			
cxf-rs-security-xml			
cxf-rt-security			
cxf-specs			
cxf-sts			
cxf-tools			
cxf-transports-jms			
cxf-transports-local			
cxf-ws-addr			
cxf-ws-mex			
cxf-ws-policy			
cxf-ws-rm			
cxf-ws-security			
cxf-wsn-api			
cxf-xjc-runtime			
hazelcast		In memory data grid	
metrics-reporting	metrics-reporting, metrics-webconsole-plugin	Metrics reporting for DDF.	
metrics-services	metrics-interceptor	cxf interceptors to capture metrics.	
mime-core	mime-core-impl, mime-core-configurableresolver	DDF MIME Core	
mime-core-api	mime-core-api	DDF MIME API	
mime-tika-resolver	mime-tika-resolver	DDF Tika MIME Resolver.	
platform-app		DDF platform boot features	
platform-configuration	platform-configuration	Schedules tasks	
platform-scheduler	platform-scheduler	Schedules tasks	

platform-settings-config		Sets the DDF System Settings with default values	
platform-status	platform-status	Schedules tasks	
security-core-api	security-core-api	DDF Security Core	
security-encryption	security-encryption-api, security-encryption-impl, security-encryption-commands	DDF Security Encryption	
wss4j			
xml-specs-api			

App Prerequisites

Before the DDF Platform Application can be installed:

- the DDF Kernel must be running

App Dependencies

```
com.codahale.metrics:metrics-core:jar:3.0.1
com.googlecode.json-simple:json-simple:jar:1.1.1
commons-io:commons-io:jar:2.1
commons-lang:commons-lang:jar:2.6
ddf.security.core:security-core-api:jar:2.3.0
ddf.security.encryption:security-encryption-api:jar:2.3.0
ddf.security.encryption:security-encryption-commands:jar:2.3.0
ddf.security.encryption:security-encryption-impl:jar:2.3.0
javax.servlet:servlet-api:jar:2.5
javax.ws.rs:javax.ws.rs-api:jar:2.0-m10
joda-time:joda-time:jar:1.6.2
log4j:log4j:jar:1.2.17
org.apache.cxf:cxf-api:jar:2.7.8
org.apache.cxf:cxf-rt-frontend-jaxrs:jar:2.7.8
org.apache.cxf:cxf-rt-transports-http:jar:2.7.8
org.apache.cxf:cxf-rt-ws-policy:jar:2.7.8
org.apache.felix:org.apache.felix.gogo.command:jar:0.10.0
org.apache.felix:org.apache.felix.webconsole:jar:4.0.0
org.apache.karaf.bundle:org.apache.karaf.bundle.core:jar:3.0.0
org.apache.karaf.features:org.apache.karaf.features.core:jar:2.3.3
org.apache.karaf.shell:org.apache.karaf.shell.console:jar:2.3.3
org.apache.servicemix.bundles:org.apache.servicemix.bundles.poi:jar:3.9_2
org.apache.shiro:shiro-core:jar:1.2.1
org.apache.tika:tika-core:jar:1.2
org.jasypt:jasypt:jar:1.9.0
org.osgi:org.osgi.compendium:jar:4.3.1
org.osgi:org.osgi.core:jar:4.3.1
org.quartz-scheduler:quartz:jar:2.1.7
org.rrd4j:rrd4j:jar:2.0.7
org.slf4j:slf4j-api:jar:1.7.1
org.slf4j:slf4j-ext:jar:1.7.1
```

Third Party Licenses

[Third Party License File Notice](#)

Release Version: ddf-platform-2.4.0

Contents

- [App Dependencies](#)

App Dependencies

```
abdera-core-1.1.3.jar
```

abdera-extensions-main-1.1.3.jar
abdera-i18n-1.1.3.jar
abdera-parser-1.1.3.jar
apache-mime4j-core-0.7.2.jar
c3p0-0.9.1.1.jar
cal10n-api-0.7.4.jar
camel-blueprint-2.12.1.jar
camel-core-2.12.1.jar
camel-freemarker-2.12.1.jar
camel-http-2.12.1.jar
camel-jetty-2.12.1.jar
camel-jms-2.12.1.jar
camel-karaf-commands-2.12.1.jar
camel-spring-2.12.1.jar
camel-test-2.12.1.jar
commons-codec-1.8.jar
commons-collections-3.2.1.jar
commons-io-2.1.jar
commons-lang-2.6.jar
commons-pool-1.6.jar
cxf-api-2.7.8.jar
cxf-bundle-compatible-2.7.8.jar
cxf-karaf-commands-2.7.8.jar
cxf-rt-bindings-coloc-2.7.8.jar
cxf-rt-bindings-corba-2.7.8.jar
cxf-rt-bindings-object-2.7.8.jar
cxf-rt-bindings-soap-2.7.8.jar
cxf-rt-bindings-xml-2.7.8.jar
cxf-rt-core-2.7.8.jar
cxf-rt-databinding-aegis-2.7.8.jar
cxf-rt-databinding-jaxb-2.7.8.jar
cxf-rt-databinding-jibx-2.7.8.jar
cxf-rt-databinding-xmlbeans-2.7.8.jar
cxf-rt-features-clustering-2.7.8.jar
cxf-rt-frontend-jaxrs-2.7.8.jar
cxf-rt-frontend-jaxws-2.7.8.jar
cxf-rt-frontend-js-2.7.8.jar
cxf-rt-frontend-simple-2.7.8.jar
cxf-rt-javascript-2.7.8.jar
cxf-rt-management-2.7.8.jar
cxf-rt-rs-extension-providers-2.7.8.jar
cxf-rt-rs-extension-search-2.7.8.jar
cxf-rt-rs-security-cors-2.7.8.jar
cxf-rt-rs-security-sso-saml-2.7.8.jar
cxf-rt-rs-security-xml-2.7.8.jar
cxf-rt-security-2.7.8.jar
cxf-rt-transports-http-2.7.8.jar
cxf-rt-transports-http-hc-2.7.8.jar
cxf-rt-transports-http-jetty-2.7.8.jar
cxf-rt-transports-jms-2.7.8.jar
cxf-rt-transports-local-2.7.8.jar
cxf-rt-ws-addr-2.7.8.jar
cxf-rt-ws-mex-2.7.8.jar

```
cxftools-common-2.7.8.jar  
cxftools-javaws-2.7.8.jar  
cxftools-misctools-2.7.8.jar  
cxftools-validator-2.7.8.jar  
cxftools-wadlto-jaxrs-2.7.8.jar  
cxftools-wsdlto-core-2.7.8.jar  
cxftools-wsdlto-databinding-jaxb-2.7.8.jar  
cxftools-wsdlto-frontend-javascript-2.7.8.jar  
cxftools-wsdlto-frontend-jaxws-2.7.8.jar  
cxf-xjc-runtime-2.6.2.jar  
geronimo-annotation_1.0_spec-1.1.1.jar  
geronimo-jms_1.1_spec-1.1.1.jar  
geronimo-jta_1.1_spec-1.1.1.jar  
geronimo-osgi-registry-1.1.jar  
geronimo-servlet_2.5_spec-1.2.jar  
hazelcast-1.9.4.jar  
hazelcast-2.5.jar  
hazelcast-3.1.3.jar  
httpasyncclient-osgi-4.0-beta3.jar  
httpclient-osgi-4.2.5.jar  
httpcore-osgi-4.2.4.jar  
jettison-1.3.4.jar  
jibx-bind-1.2.5.jar  
jibx-run-1.2.5.jar  
jibx-schema-1.2.5.jar  
jibx-tools-1.2.5.jar  
joda-time-1.6.2.jar  
json-simple-1.1.1.jar  
mail-1.4.4.jar  
metrics-collector-2.4.0.jar  
metrics-core-3.0.1.jar  
neethi-3.0.2.jar  
ops4j-base-lang-1.2.2.jar  
org.apache.karaf.cellar.bundle-2.3.0.jar  
org.apache.karaf.cellar.config-2.3.0.jar  
org.apache.karaf.cellar.core-2.3.0.jar  
org.apache.karaf.cellar.dosgi-2.3.0.jar  
org.apache.karaf.cellar.event-2.3.0.jar  
org.apache.karaf.cellar.features-2.3.0.jar  
org.apache.karaf.cellar.hazelcast-2.3.0.jar  
org.apache.karaf.cellar.management-2.3.0.jar  
org.apache.karaf.cellar.obr-2.3.0.jar  
org.apache.karaf.cellar.shell-2.3.0.jar  
org.apache.karaf.cellar.utils-2.3.0.jar  
org.apache.karaf.cellar.webconsole-2.3.0.jar  
org.apache.servicemix.bundles.bcel-5.2_4.jar  
org.apache.servicemix.bundles.commons-httpclient-3.1_7.jar  
org.apache.servicemix.bundles.dom4j-1.6.1_5.jar
```

```
org.apache.servicemix.bundles.ehcache-2.5.1_1.jar
org.apache.servicemix.bundles.fastinfoset-1.2.13_1.jar
org.apache.servicemix.bundles.freemarker-2.3.20_1.jar
org.apache.servicemix.bundles.jaxb-impl-2.2.1.1_2.jar
org.apache.servicemix.bundles.jaxb-xjc-2.2.1.1_2.jar
org.apache.servicemix.bundles.jaxen-1.1.3_1.jar
org.apache.servicemix.bundles.jdom-1.1_4.jar
org.apache.servicemix.bundles.junit-4.11_1.jar
org.apache.servicemix.bundles.opensaml-2.5.3_3.jar
org.apache.servicemix.bundles.poi-3.9_2.jar
org.apache.servicemix.bundles.rhino-1.7R2_3.jar
org.apache.servicemix.bundles.saaj-impl-1.3.21_1.jar
org.apache.servicemix.bundles.saxon-9.1.0.8_1.jar
org.apache.servicemix.bundles.wsdl4j-1.6.3_1.jar
org.apache.servicemix.bundles.xalan-2.7.1_7.jar
org.apache.servicemix.bundles.xerces-2.11.0_1.jar
org.apache.servicemix.bundles.xmlbeans-2.6.0_2.jar
org.apache.servicemix.bundles.xmlresolver-1.2_5.jar
org.apache.servicemix.bundles.xpp3-1.1.4c_6.jar
org.apache.servicemix.specs.activation-api-1.1-2.2.0.jar
org.apache.servicemix.specs.jaxb-api-2.2-2.2.0.jar
org.apache.servicemix.specs.jaxws-api-2.2-2.2.0.jar
org.apache.servicemix.specs.jsr339-api-m10-2.2.0.jar
org.apache.servicemix.specs.saaj-api-1.3-2.2.0.jar
org.apache.servicemix.specs.stax-api-1.0-2.2.0.jar
pax-swissbox-extender-1.3.1.jar
pax-swissbox-lifecycle-1.3.1.jar
pax-swissbox-optional-jcl-1.3.1.jar
quartz-2.1.7.jar
rrd4j-2.0.7.jar
shiro-core-1.2.1.jar
slf4j-ext-1.7.1.jar
stax2-api-3.1.1.jar
tika-core-1.2.jar
velocity-1.7.jar
woodstox-core-asl-4.2.0.jar
wss4j-1.6.13.jar
xmlschema-core-2.0.3.jar
xmlsec-1.5.6.jar
```

Release Version: platform-2.5.0

Contents

- New Capabilities
- Resolved Issues
- Known Issues
- App Dependencies
- Third Party Licenses

New Capabilities

- Added a persistent store
- Solr Server (migrated from the Solr application)

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

App Prerequisites

Before the DDF Platform Application can be installed:

- the DDF Kernel must be running

App Dependencies

```
com.codahale.metrics:metrics-core:jar:3.0.1
com.googlecode.json-simple:json-simple:jar:1.1.1
commons-io:commons-io:jar:2.1
commons-lang:commons-lang:jar:2.6
ddf.security.core:security-core-api:jar:2.3.0
ddf.security.encryption:security-encryption-api:jar:2.3.0
ddf.security.encryption:security-encryption-commands:jar:2.3.0
ddf.security.encryption:security-encryption-impl:jar:2.3.0
javax.servlet:servlet-api:jar:2.5
javax.ws.rs:javax.ws.rs-api:jar:2.0-m10
joda-time:joda-time:jar:1.6.2
log4j:log4j:jar:1.2.17
org.apache.cxf:cxf-api:jar:2.7.8
org.apache.cxf:cxf-rt-frontend-jaxrs:jar:2.7.8
org.apache.cxf:cxf-rt-transports-http:jar:2.7.8
org.apache.cxf:cxf-rt-ws-policy:jar:2.7.8
org.apache.felix:org.apache.felix.gogo.command:jar:0.10.0
org.apache.felix:org.apache.felix.webconsole:jar:4.0.0
org.apache.karaf.bundle:org.apache.karaf.bundle.core:jar:3.0.0
org.apache.karaf.features:org.apache.karaf.features.core:jar:2.3.3
org.apache.karaf.shell:org.apache.karaf.shell.console:jar:2.3.3
org.apache.servicemix.bundles:org.apache.servicemix.bundles.poi:jar:3.9_2
org.apache.shiro:shiro-core:jar:1.2.1
org.apache.tika:tika-core:jar:1.2
org.jasypt:jasypt:jar:1.9.0
org.osgi:org.osgi.compendium:jar:4.3.1
org.osgi:org.osgi.core:jar:4.3.1
org.quartz-scheduler:quartz:jar:2.1.7
org.rrd4j:rrd4j:jar:2.0.7
org.slf4j:slf4j-api:jar:1.7.1
org.slf4j:slf4j-ext:jar:1.7.1
```

Third Party Licenses

[Third Party License File Notice](#)

Release Version: platform-2.5.1

Contents

- [Overview](#)
- [Resolved Issues](#)
- [Known Issues](#)
- [App Dependencies](#)
- [Third Party Licenses](#)

Overview

- This release is a bug fix release of platform 2.5.0

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

App Prerequisites

Before the DDF Platform Application can be installed:

- the DDF Kernel must be running

App Dependencies

```
com.codahale.metrics:metrics-core:jar:3.0.1
com.googlecode.json-simple:json-simple:jar:1.1.1
commons-io:commons-io:jar:2.1
commons-lang:commons-lang:jar:2.6
ddf.security.core:security-core-api:jar:2.3.0
ddf.security.encryption:security-encryption-api:jar:2.3.0
ddf.security.encryption:security-encryption-commands:jar:2.3.0
ddf.security.encryption:security-encryption-impl:jar:2.3.0
javax.servlet:servlet-api:jar:2.5
javax.ws.rs:javax.ws.rs-api:jar:2.0-m10
joda-time:joda-time:jar:1.6.2
log4j:log4j:jar:1.2.17
org.apache.cxf:cxf-api:jar:2.7.8
org.apache.cxf:cxf-rt-frontend-jaxrs:jar:2.7.8
org.apache.cxf:cxf-rt-transports-http:jar:2.7.8
org.apache.cxf:cxf-rt-ws-policy:jar:2.7.8
org.apache.felix:org.apache.felix.gogo.command:jar:0.10.0
org.apache.felix:org.apache.felix.webconsole:jar:4.0.0
org.apache.karaf.bundle:org.apache.karaf.bundle.core:jar:3.0.0
org.apache.karaf.features:org.apache.karaf.features.core:jar:2.3.3
org.apache.karaf.shell:org.apache.karaf.shell.console:jar:2.3.3
org.apache.servicemix.bundles:org.apache.servicemix.bundles.poi:jar:3.9_2
org.apache.shiro:shiro-core:jar:1.2.1
org.apache.tika:tika-core:jar:1.2
org.jasypt:jasypt:jar:1.9.0
org.osgi:org.osgi.compendium:jar:4.3.1
org.osgi:org.osgi.core:jar:4.3.1
org.quartz-scheduler:quartz:jar:2.1.7
org.rrd4j:rrd4j:jar:2.0.7
org.slf4j:slf4j-api:jar:1.7.1
org.slf4j:slf4j-ext:jar:1.7.1
```

Third Party Licenses

[Third Party License File Notice](#)

Release Version: platform-2.6.1

Contents

- [Overview](#)
- [Resolved Issues](#)
- [Known Issues](#)

Overview

- This release is a bug fix release of platform 2.6.0
- Added EXI Compression

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
<p> Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</p>										

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
<p> Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</p>										

App Prerequisites

Before the DDF Platform Application can be installed:

- the DDF Kernel must be running

DDF Registry Application

Overview

The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them.

DDF Registry Application Table of Contents

- [Managing DDF Registry Application](#) — DDF Registry Application
- [Integrating DDF Registry Application](#) — DDF Registry Application
- [Extending DDF Registry Application](#) — DDF Registry Application
- [Securing DDF Registry Application](#) — DDF Registry Application
- [DDF Registry Application Release Notes](#) — DDF Registry Application

The DDF Registry code is currently a very functional, but initial prototype that was created to show a basic registry framework working. As such, it does have limitations and should be tested for a desired use case before using.

Managing DDF Registry Application

Overview

The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them.

Table of Contents

Integrating DDF Registry Application

Overview

The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them.

Table of Contents

Extending DDF Registry Application

Overview

The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them.

Table of Contents

Securing DDF Registry Application

Overview

The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them.

Table of Contents

DDF Registry Application Release Notes

Overview

The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them.

Table of Contents

There are currently no released versions of the DDF Registry application.

DDF Security Application

Overview

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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

DDF Security Application Table of Contents

- [Managing DDF Security Application](#) — provides authentication, authorization, and auditing services for the ApplicationName
 - [DDF Security Application Install and Uninstall](#)
- [Integrating DDF Security Application](#) — DDF Security Application
 - [Security CAS](#)
 - [Security CAS Client](#)
 - [Security CAS Implementation](#)
 - [Security CAS Server](#)
 - [Security CAS Token Validator](#)
 - [Security CAS CXF Servlet Filter](#)
 - [Security Core](#)
 - [Security Core Commons](#)
 - [Security Core Implementation](#)
 - [Security Encryption](#)
 - [Security LDAP](#)
 - [Security PDP](#)
 - [Security PDP AuthZ Realm](#)
 - [Security PDP XACML Realm](#)
 - [Anonymous Interceptor](#)
- [Extending DDF Security Application](#) — DDF Security Application
 - [Developing Token Validators](#)
- [Securing DDF Security Application](#) — DDF Security Application
- [DDF Security Application Release Notes](#) — DDF Security Application
 - [Release Version: security-2.5.1](#)

Managing DDF Security Application

On This Page

- [Overview](#)
- [Install and Uninstall](#)
- [Configuration](#)
- [Whitelist](#)
- [Applications](#)

Overview

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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

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

The following pages give a description of the various services, their containing bundles, configuration settings for the bundles, and which services are exported and imported.

Install and Uninstall

Refer to the [DDF Security Application Install/Uninstall](#) page for how the Security services can be installed and uninstalled.

Configuration

This component can be configured using the normal processes described in the [Configuration](#) section. Within the pages for each of the applications, are specific instructions on the configurations for the bundles and any additional information that may help decide how the configuration should be set for use cases.

Whitelist

The following packages have been exported by the DDF Security application and are approved for use by third parties:

- ddf.security.expansion
- ddf.security.sts.client.configuration
- ddf.security.common.callback
- ddf.security.common.util

Applications

- Security Core
- Security CAS
- Security Encryption
- Security PEP
- Security PDP
- Security STS

DDF Security Application Install and Uninstall

On This Page

- Overview
- Prerequisites
- Install
- Verify
- Uninstall
 - Revert the Uninstall
- Upgrade

Overview

Prerequisites

Before the DDF Security application can be installed:

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

Install

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.

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

Verify

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.

Uninstall

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

```
features:removeurl -u <DDF application's feature repository URL>
```

Example: `features:removeurl -u mvn:ddf.security/security-services-app/2.4.0/xml/features`

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

The repository URLs for installed applications can be obtained by entering:

```
features:listrepositories -u
```

Revert 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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf.catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

Upgrade

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.

```
features:install admin-app
```

4. Complete the steps in the Verify section above to determine if the upgrade was successful.

Integrating DDF Security Application

Overview

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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide supports integration of this application with external frameworks.

Table of Contents

- Security CAS
- Security Core
- Security PDP
- Anonymous Interceptor

Security CAS

On This Page

- Overview
- Components

Overview

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.

Components

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

Security CAS Client

On This page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

Installation

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

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
<code>javax.servlet.Filter</code>	<code>ddf.security.cas.client.ProxyFilter</code>	CAS Filters

Security CAS Implementation

On This Page

- Overview
- Configuration
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

None.

Implementation Details

Imported Services

None

Exported Services

None

Security CAS Server

On This Page

- Overview
- Configuration
- Implementation Details

Overview

The Security CAS Server project creates a web application (.war) file that is configured to be deployed to a tomcat application server. Information on installing and configuring it within tomcat is available on the [CAS SSO Configuration page](#).

Configuration

N/A - Not a bundle

Implementation Details

N/A - Not a bundle

Security CAS Token Validator

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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

Configuration

Installation

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

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.TokenValidator</code>	<code>ddf.security.cas.WebSSOTokenValidator</code>	CAS Server URL and Encryption Service reference

Security CAS CXF Servlet Filter

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

Installation

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

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: /services/catalog/query.

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

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
javax.servlet.Filter	required	false

Exported Services

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

Security Core

On This Page

- Overview
- Components

Overview

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.

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

Security Core Commons

On This Page

- Overview
- Configuration
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

None

Implementation Details

Imported Services

None

Exported Services

None

Security Core Implementation

On This Page

- Overview
- Configuration
- Install and Uninstall
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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

Configuration

None

Install and Uninstall

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

Implementation Details

Imported Services

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

Exported Services

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

Security Encryption

On This Page

- Overview
- Components

Overview

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

Components

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

Security Encryption API

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

Installation

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.

Settings

None

Implementation Details

Imported Services

None

Exported Services

None

Security Encryption Commands

On This Page
<ul style="list-style-type: none">• Overview• Configuration<ul style="list-style-type: none">• Installation• Settings• Implementation Details<ul style="list-style-type: none">• Imported Services• Exported Services

Overview

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.

Configuration

Installation

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, administrators will not be able to encrypt and decrypt data from the console.

Settings

None

Implementation Details

Imported Services

None

Exported Services

None

Security Encryption Implementation

On This Page
<ul style="list-style-type: none">• Overview• Configuration<ul style="list-style-type: none">• Installation• Settings• Implementation Details<ul style="list-style-type: none">• Imported Services• Exported Services

Overview

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.

Configuration

Installation

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.

Settings

None

Implementation Details

Imported Services

None

Exported Services

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

Security LDAP

On This Page

- Overview
- Components

Overview

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.

Components

Bundle Name	Feature Located In	Description/Link to Bundle Page
ldap-embedded	ldap	Embedded LDAP Configuration

Configuring a Standalone LDAP Server

On This Page

- Overview
- Run a Standalone Embedded LDAP Instance
- Configuration
- Trust Certificates
- Connect to a Standalone LDAP Server

Overview

In some production environments it is suggested that the LDAP server be run separate from the DDF installation. Due to the minimal number of dependencies that the embedded LDAP application requires, this app can be run using a minimal install of DDF that uses much less memory and CPU than a standard installation.

Run a Standalone Embedded LDAP Instance

1. Obtain and unzip the DDF kernel (ddf-distribution-kernel-<VERSION>.zip).
2. Start the distribution (<https://login.macefusion.com/secure/login?service=https%3A%2F%2Fwiki.macefusion.com%2Fpages%2Fviewpage.action%3Ftitle%3DStarting%2Band%2BStopping%26spaceKey%3DDDF>).
3. When the kernel is loaded up with the DDF logo at the command prompt, execute:

```
la
```

which is short for "list all."

Since the kernel does not include all apps, if you were to do a "list" instead of "la," no results would be returned at this point.

4. Verify that all bundles are Active.
5. Deploy the Embedded LDAP app by copying the `ldap-embedded-app-<VERSION>.kar` file into the `<DISTRI-BUTION_HOME>/deploy` directory. You can verify that the LDAP server is installed by checking the DDF log or by performing an `la` and verifying that the OpenDJ bundle is in the Active state. Additionally, it should be responding to LDAP requests on the default ports, 1389 and 1636.
6. To perform any of the configurations identified below, the web console will need to be installed by executing:

```
features:install webconsole
```

Configuration

The configuration options are located on the standard DDF configuration web 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.

Trust Certificates

For LDAPS 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. Navigate to the `/etc/keystores` folder in the kernel distribution folder.
2. Find the `serverKeystore.jks` file and replace it with a keystore file valid for your operating environment.
3. If the DDF kernel is running, restart it so the changes will take place

Connect to a Standalone LDAP Server

DDF instances can connect to an external LDAP server by installing and configuring the `security-sts-server` feature detailed [here](#).

Embedded LDAP Configuration

On This Page
<ul style="list-style-type: none"> • Overview • Default Settings <ul style="list-style-type: none"> • Ports • Users <ul style="list-style-type: none"> • LDAP Users • LDAP Admin • Schemas • Configuration <ul style="list-style-type: none"> • Start and Stop • Settings • Limitations • External Links

Overview

The Embedded LDAP application contains an LDAP server (OpenDJ version 2.4.6) that has a default set of schemas and users loaded to help facilitate authentication and authorization testing.

Default Settings

Ports

Protocol	Default Port
LDAP	1389

LDAP	1389
LDAPS	1636
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 (http://opendj.forgerock.org/doc/admin-guide/index/chap-schema.html)
00-core.ldif	This file contains a core set of attribute type and objectclass definitions from several standard LDAP documents, including draft-ietf-boreham-numsubordinates, draft-findlay-ldap-groupofentries, draft-furuseh-ldap-untypedobject, draft-good-ldap-changelog, draft-ietf-ldup-subentry, draft-wahl-ldap-adminaddr, RFC 1274, RFC 2079, RFC 2256, RFC 2798, RFC 3045, RFC 3296, RFC 3671, RFC 3672, RFC 4512, RFC 4519, RFC 4523, RFC 4524, RFC 4530, RFC 5020, and X.501.
01-pwpolicy.ldif	This file contains schema definitions from draft-behera-ldap-password-policy, which defines a mechanism for storing password policy information in an LDAP directory server.
02-config.ldif	This file contains the attribute type and objectclass definitions for use with the directory server configuration.
03-changelog.ldif	This file contains schema definitions from draft-good-ldap-changelog, which defines a mechanism for storing information about changes to directory server data.
03-rfc2713.ldif	This file contains schema definitions from RFC 2713, which defines a mechanism for storing serialized Java objects in the directory server.
03-rfc2714.ldif	This file contains schema definitions from RFC 2714, which defines a mechanism for storing CORBA objects in the directory server.
03-rfc2739.ldif	This file contains schema definitions from RFC 2739, which defines a mechanism for storing calendar and vCard objects in the directory server. Note that the definition in RFC 2739 contains a number of errors, and this schema file has been altered from the standard definition in order to fix a number of those problems.
03-rfc2926.ldif	This file contains schema definitions from RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.
03-rfc3112.ldif	This file contains schema definitions from RFC 3112, which defines the authentication password schema.

03-rfc3712.ldif	This file contains schema definitions from RFC 3712, which defines a mechanism for storing printer information in the directory server.
03-uddiv3.ldif	This file contains schema definitions from RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.
04-rfc2307bis.ldif	This file contains schema definitions from the draft-howard-rfc2307bis specification, used to store naming service information in the directory server.
05-rfc4876.ldif	This file contains schema definitions from RFC 4876, which defines a schema for storing Directory User Agent (DUA) profiles and preferences in the directory server.
05-samba.ldif	This file contains schema definitions required when storing Samba user accounts in the directory server.
05-solaris.ldif	This file contains schema definitions required for Solaris and OpenSolaris LDAP naming services.
06-compat.ldif	This file contains the attribute type and objectclass definitions for use with the directory server configuration.

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 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 web 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.

Limitations

Current limitations for the embedded LDAP instances include:

1. 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.
2. Cannot be run standalone from DDF. In order to run embedded-ldap, the DDF must be started.

External Links

Location to the default base LDIF file in the DDF source code: <https://github.com/codice/ddf/blob/master/ldap/embedded/ldap-embedded-opendj/src/main/resources/default-users.ldif>

OpenDJ documentation: <http://opendj.forgerock.org/docs.html>

LDAP Administration

On This Page

- Overview
- Download the Admin Tools
- Use the Admin Tools
- Example Commands for Disabling/Enabling a User's Account
 - Verify the Account is Disabled
 - Enable the Account
 - Verify the Account is Enabled

Overview

OpenDJ provides a number of tools for LDAP administration. Refer to the OpenDJ Admin Guide (<http://opendj.forgerock.org/opendj-server/doc/admin-guide/>).

Download the Admin Tools

OpenDJ (Version 2.4.6) and the included tool suite can be downloaded at <http://www.forgerock.org/opendj-archive.html>.

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:

```

Security PEP

On This Page

- Overview
- Components

Overview

The DDF Security 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.

Components

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

Security PEP Interceptor

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

The Security PEP Interceptor bundle contains the `ddf.security.pep.interceptor.PEPAuthorizingInterceptor` class. This class uses CXF to intercept incoming SOAP messages and enforces service authorization policies by sending the service request to the security framework.

Configuration

Installation

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

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

Settings

None

Implementation Details

Imported Services

None

Exported Services

None

Security PEP Redaction

On This Page

- Overview
- Configuration
 - Installation
 - Configuration
- Implementation Details
 - Imported Services
 - Exported Services

Overview

The Security PEP Redaction bundle contains a redaction plugin that is added as a post query plugin in the DDF query lifecycle. This plugin looks at the security attributes on the metocard and compares them to the security attributes on the user who made the query request. If they do not match, the plug in will, depending on the configuration, filter the metocard out of the results or redact certain parts of the metocard that the user does not have permission to see.

Configuration

Installation

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

Configuration

None

Implementation Details

Imported Services

None

Exported Services

Registered Interface	Implementation Class	Properties Set
ddf.catalog.plugin.PostQueryPlugin	ddf.security.pep.redaction.plugin.RedactionPlugin	None

Security STS

On This Page

- Overview
- Components

Overview

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 add components specific to DDF functionality.

Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
security-sts-clientconfig	security-sts-realm	Security STS Client Config
security-sts-realm	security-sts-realm	Security STS Realm
security-sts-ldaplogin	security-sts-ldaplogin	Security STS LDAP Login
security-sts-ldapclaimshandler	security-sts-server	Security STS LDAP Claims Handler
security-sts-server	security-sts-server	Security STS Server
security-sts-samlvalidator	security-sts-server	Contains the default CXF SAML validator, exposes it as a service for the STS.
security-sts-x509validator	security-sts-server	Contains the default CXF x509 validator, exposes it as a service for the STS.

Security STS Client Config

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

Installation

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

Settings can be found in the web console under **Configuration Security STS Client**.

Configuration Name	Default Value	Additional Information

STS Address	https://server:8993/services/SecurityTokenService	The hostname of the remote server should match the certificate that the server is using.
STS Endpoint Name	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}STS_Port	
STS Service Name	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}SecurityTokenService	
Signature Properties	etc/ws-security/client/signature.properties	
Encryption Properties	etc/ws-security/client/encryption.properties	
STS Properties	etc/ws-security/client/signature.properties	
Claims	<List of Claims>	

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
ddf.catalog.DdfConfigurationWatcher	required	true
org.osgi.service.cm.ConfigurationAdmin	required	false

Exported Services

None

Security STS LDAP Claims Handler

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

Installation

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

Settings

Settings can be found in the web console under **Configuration Security STS LDAP and Roles Claims Handler**.

Configuration Name	Default Value	Additional Information
LDAP URL	ldaps://localhost:1636	
LDAP Bind User DN	cn=admin	

LDAP Bind User Password	secret	This password value is encrypted by default using the Security Encryption application .
LDAP Username Attribute	uid	
LDAP Base User DN	ou=users,dc=example,dc=com	
LDAP Group ObjectClass	groupOfNames	ObjectClass that defines structure for group membership in LDAP. Usually this is groupOfNames or groupOfUniqueNames
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.

Note that currently a server restart is required in order for changes to these settings to take effect.

Implementation Details

Imported Services

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

Exported Services

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

Security STS LDAP Login

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

Configuration

Installation

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

Configuration settings can be found in the web console under **Configuration Security STS LDAP Login**.

Configuration Name	Default Value	Additional Information
LDAP URL	ldaps://localhost:1636	
LDAP Bind User DN	cn=admin	
LDAP Bind User Password	secret	This password value is encrypted by default using the Security Encryption application.
LDAP Username Attribute	uid	
LDAP Base User DN	ou=users,dc=example,dc=com	
LDAP Base Group DN	ou=groups,dc=example,dc=com	
SSL Keystore Alias	server	This alias is used when connecting to the LDAP using SSL (LDAPS).

Implementation Details

Imported Services

None

Exported Services

None

Security STS Realm

On This Page

- [Overview](#)
- [Configuration](#)
 - [Installation](#)
 - [Settings](#)
- [Implementation Details](#)
 - [Imported Services](#)
 - [Exported Services](#)

Overview

The DDF Security STS Realm performs authentication of a user by delegating the authentication request to an STS. This is different than the realms located within the [Security PDP](#) application as those ones only perform authorization and not authentication.

Configuration

Installation

This bundle is installed by default and should not be uninstalled unless the security framework is not being used.

Settings

None

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
ddf.security.encryption.EncryptionService	opt	false

Exported Services

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

Security STS Server

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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.

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

Configuration

Installation

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

Settings

Configuration settings can be found in the web 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.

Implementation Details

Imported Services

Registered Interface	Availability	Multiple
<code>org.apache.cxf.sts.claims.ClaimsHandler</code>	optional	true
<code>org.apache.cxf.sts.token.validator.TokenValidator</code>	optional	true

Exported Services

None

Security PDP

On This Page

- Overview
- Components

Overview

The DDF Security PDP application 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 [Security PEP](#) application.

Components

Bundle Name	Located in Feature	Description/Link to Bundle Page
security-pdp-xacmlrealm	security-pdp-xacml	Security PDP XACML Realm
security-pdp-authzrealm	security-pdp-simple	Security PDP AuthZ Realm

Security PDP AuthZ Realm

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

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. Unlike the [Security PDP XACML Realm](#), this realm does not use XACML and does not delegate decisions to an external processing engine. Decisions are made based on "match-all" and "match-one" logic. The configuration below provides the mapping between user attributes and metocard attributes - 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.

Configuration

Installation

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

Settings

Settings can be found in the web console under **Configuration Security Simple AuthZ Realm**.

Configuration Name	Default Value	Additional Description
Roles	admin	Add all the roles that allow access to restricted actions. Any user that has any one of these roles will be allowed access to restricted actions.
Open Action List		Add any actions that will not be restricted by role. Any action listed here will automatically be allowed to be performed by any user in any role.
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
org.apache.shiro.realm.Realm	ddf.security.pdp.realm.SimpleAuthzRealm	None
org.apache.shiro.authz.Authorizer		

Security PDP XACML Realm

On This Page

- Overview
- Configuration
 - Installation
 - Settings
- Implementation Details
 - Imported Services
 - Exported Services

Overview

The DDF Security PDP XACML realm exposes a realm that creates a XACML request with the incoming authorization information and sends the request to a XACML processing engine. The engine that is sent the request is not hardcoded and is retrieved at runtime by the OSGi service registry. This realm contains an embedded XACML processing engine that handles the requests and policies.

Configuration

Installation

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

Settings

None

Implementation Details

Imported Services

None

Exported Services

Registered Interfaces	Implementation Class	Properties Set
org.apache.shiro.realm.Realm	ddf.security.pep.realm.XACMLRealm	None
org.apache.shiro.authz.Authorizer		

Anonymous Interceptor

Overview

The goal of the AnonymousInterceptor 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 anonymous interceptor detects the missing credentials and inserts an assertion that represents the "anonymous" user. The attributes included in this anonymous user assertion are configured by the administrator to represent any unknown user on the current network.

On This Page

- Overview
- Installing
- Configuring
 - Configuring via the Admin Console

Installing

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

Configuring

Configuring via the Admin Console

1. Navigate to the [DDF Admin Console Configuration](#)
2. Select "Security STS Anonymous Claims Handler"
3. Click the + next to Attributes to add a new attribute
4. Add the following attribute: "<http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier=anonymous>"
5. Repeat the steps above to add another new attribute: "<http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier=anonymous>"
6. Click save
7. Now select "Security Simple AuthZ Realm"
8. Under role, add anonymous so Roles value is: "admin,anonymous"
9. Click save.

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

Extending DDF Security Application

Overview

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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide supports developers creating extensions of the existing framework.

Table of Contents

- Developing Token Validators

Developing Token Validators

Description

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.

Usage

At the moment, validators must be added to the `cxft-sts.xml` blueprint file manually. There is a section labeled as the "Delegate Configuration" in that file. That is where the validator must be added along with the existing validators. In the future, we expect this to be pluggable without changes to the blueprint file.

Update: The validator services that are currently bundled on their own are webSSOTokenValidator, SamlTokenValidator, and x509TokenValidator. Each has its own blueprint.xml file that defines and exports the service. They can be individually turned on/off as a bundle/service.

Securing DDF Security Application

Overview

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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide covers implementations and protocols for enhancing security.

Table of Contents

DDF Security Application Release Notes

Overview

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. More information about the security framework and how everything works as a single security solution can be found on the [Managing Web Service Security](#) page.

This guide covers implementations and protocols for enhancing security.

Table of Contents

- Release Version: security-2.5.1

Release Version: security-2.5.1

Contents

- Overview
- Resolved Issues
- Known Issues
- App Prerequisites
- Third Party Licenses

Overview

This is a bug fix release to address issues discovered in security 2.5.0

- Anonymous Interceptor added

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due
------	-----	---------	----------	----------	----------	--------	------------	---------	---------	-----

 JIRA project doesn't exist or you don't have permission to view it.

[View these issues in JIRA](#)

App Prerequisites

Before the DDF Security Application can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.6.1 must be running

Third Party Licenses

[Third Party License File Notice](#)

DDF Solr Catalog Application

Overview

The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store.

Some notable features of the SCP include:

- Supports extensible metacards
- Fast, simple contextual searching
- Indexes XML Attributes (<https://tools.codic.org/#DDFSolrCatalogApplicationUser'sGuide-IndexingText>) 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

DDF Solr Catalog Application Table of Contents

- [Managing DDF Solr Catalog Application](#) — provides an implementation of the CatalogProvider interface using Apache Solr <http://lucene.apache.org/solr/> as a data store
- [Integrating DDF Solr Catalog Application](#) — DDF Solr Catalog Application
 - [DDF Catalog Solr Embedded Provider](#)
 - [DDF Catalog Solr External Provider](#)
 - [Standalone Solr Server](#) — an Apache Solr instance as a Catalog data store within the distribution
- [Extending DDF Solr Catalog Application](#) — DDF Solr Catalog Application
- [Securing DDF Solr Catalog Application](#) — DDF Solr Catalog Application _security_guide_intro
- [DDF Solr Catalog Application Release Notes](#)
 - [Release Version: solr-2.4.2](#)
 - [Release Version: solr-2.4.4](#)

Managing DDF Solr Catalog Application

On This Page

- Overview
- Solr Catalog Provider Implementations
- Implementation Details
 - Indexing Text
- Whitelist
- Known Issues

Overview

The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store. This guide documents the installation, maintenance, and support of this application.

Solr Catalog Provider Implementations

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.

Whitelist

No packages have been exported by the DDF Solr Catalog Application; therefore, there are no packages approved for use by third parties.

Known Issues

- When searching with the ANY_TEXT field, SCP does not search all text fields within the Catalog Provider. Instead, it searches the METADATA field.
SCP does not fully support spatial capabilities.
- SCP does not support pole wrapping spatial queries.
- SCP ignores the following AttributeDescriptor methods: isIndexed, isTokenized, isMultivalued, isStored. SCP instead indexes, tokenizes, and stores data based on the AttributeFormat, such as it will store and not index all fields labeled as AttributeFormat.BINARY regardless of user instruction. Currently, SCP has no multivalue support even though it is supported by Solr.
- SCP has a 1000 nautical mile limit for nearest neighbor queries. If a point is not provided, the centroid of the shape will be used for distance calculations.
- SCP support for XPath ignores namespaces.
- SCP only supports XPath queries against the metadata attribute. Other XML attributes are not supported.
- Incorrect results may be returned if temporal/spatial queries are OR'ed with XPath queries.

Integrating DDF Solr Catalog Application

Overview

The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store.

This guide supports integration of this application with external frameworks.

Table of Contents

- DDF Catalog Solr Embedded Provider
- DDF Catalog Solr External Provider
- Standalone Solr Server

Prerequisites

Before the DDF Solr Catalog 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

Install

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.

Uninstall

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

```
features:removeurl -u <DDF application's feature repository URL>

Example: features:removeurl -u
mvn:ddf.catalog/catalog-solr-app/2.3.0/xml/features
```

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

Revert the Uninstall

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

1. copying the application's KAR file previously in the <INSTALL_DIRECTORY>/deploy directory, OR
2. 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.

Upgrade

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.Z.kar file to the <INSTALL_DIRECTORY>/deploy directory.
3. Start the application.

DDF Catalog Solr Embedded Provider

On This Page

- Usage
 - Embedded Solr Catalog Provider Pros and Cons
 - When to Use
- Installation
 - Install
 - Verify
- Configuration
 - Embedded Solr Server and Solr Catalog Provider
 - Configurable Properties
 - Solr Configuration Files
 - Move Solr Data to a New Location

Usage

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 solution. However, it does not provide a Solr Admin GUI (<http://wiki.apache.org/solr/SolrAdminGUI>) or a "REST-like HTTP/XML and JSON API." If that is necessary, see Standalone Solr Server.

Embedded Solr Catalog Provider Pros and Cons

Feature	Pro	Con
Scalability		<ul style="list-style-type: none">• Does not scale. Only runs one single server instance.• Does not allow the middle tier to be scaled.
Flexibility	<ul style="list-style-type: none">• 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.	<ul style="list-style-type: none">• Can only be interfaced using Java
(Administrative) Tools	<ul style="list-style-type: none">• External open source tools like Luke will work to allow admins to check index contents• JMX metrics reporting is enabled	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Does not open any ports which means no ports have to be secured.	
Performance	<ul style="list-style-type: none">• Requires no HTTP or network overhead• Near real-time indexing• Can understand complex queries	
Backup/Recovery	<ul style="list-style-type: none">• Can manually or through custom scripts back up the indexes	<ul style="list-style-type: none">• Must copy files when server is shutdown

When to Use

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

Installation

Install

By default, the DDF Solr Catalog application installs the External Solr Provider. Uninstall it by uninstalling the `catalog-solr-external-provider` feature, then install the `catalog-solr-embedded-provider` feature.

DDF Solr Catalog Application features for Embedded Solr Catalog installation

```
ddf@local>features:uninstall catalog-solr-external-provider  
ddf@local>features:install catalog-solr-embedded-provider
```

Verify

Verify the catalog-solr-embedded-provider feature has been successfully installed and the catalog-solr-external-provider feature has been uninstalled.

DDF Solr Catalog Application features for Embedded Solr Catalog Provider configuration

```
ddf@local>features:list | grep solr-app  
[installed ] [2.4.4] [           ] catalog-solr-embedded-provider  
solr-app-2.4.4 [           ] Catalog Provider with locally  
Embedded Solr Server, implemented using Solr 4.7.2.  
[uninstalled] [2.4.4] [           ] catalog-solr-external-provider  
solr-app-2.4.4 [           ] Catalog Provider to interface with  
an external Solr 4.7.2 Server  
[installed ] [2.4.4] [           ] solr-app  
solr-app-2.4.4 [           ] The Solr Catalog Provider (SCP) is  
an implementation of the CatalogProvider interface using Apache Solr 4.7.2  
as a data store.\nThe SCP supports extensible metacards, fast/simple  
contextual searching, indexes xml attributes/CDATA sections/XML text  
elements, contains full XPath support, works with an embedded local Solr  
Server, and also works with a standalone Solr Server.:DDF Solr Catalog
```

Verify the DDF Solr Catalog Application bundle is Active for the Embedded Solr Catalog Provider configuration.

DDF Solr Catalog Application active bundles for Embedded Solr Catalog Provider configuration

```
ddf@local>list | grep -i solr  
[ 374] [Active ] [           ] [           ] [     80] DDF :: Platform ::  
Solr :: Server :: Standalone War (2.6.1)  
[ 375] [Active ] [Created ] [           ] [     80] DDF :: Catalog ::  
Solr :: Embedded :: Provider (2.4.4)
```

Configuration

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 <DISTRIBUTION_INSTALLATION_DIRECTORY>/data/solr by default. A user does not have to specify any parameters. In addition, 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 using the normal processes described in the [Configuring DDF](#) section. The configurable properties for the SCP are accessed from the **Catalog Embedded Solr Catalog Provider** configurations in the Web Console.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Data Directory File Path	dataDirectoryPath	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 <DISTRIBUTION_INSTALLATION_DIRECTORY>/data/solr.</p> <p>If dataDirectoryPath is left blank (empty string), it will default to <DISTRIBUTION_INSTALLATION_DIRECTORY>/data/solr.</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 <DISTRIBUTION_INSTALLATION_DIRECTORY>/scp/solr_data.</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
Force Auto Commit	forceAutoCommit	Boolean / Checkbox	<p>Performance Impact Only in special cases should auto-commit be forced. Forcing auto-commit makes the search results visible immediately.</p>		No

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

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.

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 Web 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.

Changes Require a Distribution Restart

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

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.

DDF Catalog Solr External Provider

On This Page

- Usage
 - Solr Catalog External Provider Pros and Cons
 - When to Use
- Installation
 - Install
 - Verify
- Configuration
 - Configurable Properties
- Implementation Details
 - Indexing Text

Usage

The Solr Catalog External 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](#).

Solr Catalog External Provider Pros and Cons

Feature	Pro	Con
Scalability	<ul style="list-style-type: none">• 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."	<ul style="list-style-type: none">• Solr Cloud Catalog Provider not implemented yet.
Flexibility	<ul style="list-style-type: none">• 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.	
(Administrative) Tools	<ul style="list-style-type: none">• Contains Solr Admin GUI, which allows admins to query, check health, see metrics, see configuration files and preferences, etc.• External open source tools like Luke will work to allow admins to check index contents.• JMX metrics reporting is enabled.	
Security	<ul style="list-style-type: none">• Inherits app server security.	<ul style="list-style-type: none">• Web Console must be secured and is openly accessible.• REST-like HTTP/XML and JSON APIs must be secured.• Current Catalog Provider implementation requires sending unsecured messages to Solr. Without a coded solution, requires network or firewall restrictions in order to secure.
Performance	<ul style="list-style-type: none">• If scaled, high performance.• Near real-time indexing.	<ul style="list-style-type: none">• Possible network latency impact• Extra overhead when sent over HTTP. Extra parsing for XML, JSON, or other interface formats.• Possible limitations upon requests and queries dependent on HTTP server settings.
Backup/Recovery	<ul style="list-style-type: none">• Built-in recovery tools that allow in-place backups (does not require server shutdown).• Backup of Solr indexes can be scripted.	<ul style="list-style-type: none">• Recovery is performed as an HTTP request.

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](#).

Installation

Install

By default, the DDF Solr Catalog application installs the External Solr Provider via the `catalog-solr-external-provider` feature. This will not install any Solr Servers. Installing this feature will provide a user an "unconfigured" Solr Catalog Provider. See the Configuration section for how to configure this external Solr Catalog Provider to connect to an external Solr Server.

DDF Solr Catalog Application features for External Solr Catalog installation

```
ddf@local>features:uninstall catalog-solr-embedded-provider  
ddf@local>features:install catalog-solr-external-provider
```

Verify

Verify the `catalog-solr-external-provider` feature is installed and the `catalog-solr-embedded-provider` feature is uninstalled.

DDF Solr Catalog Application features for External Solr Catalog Provider configuration

```
ddf@local>features:list | grep solr-app  
[uninstalled] [2.4.4] [           ] catalog-solr-embedded-provider  
solr-app-2.4.4 [           ] Catalog Provider with locally  
Embedded Solr Server, implemented using Solr 4.7.2.  
[installed] [2.4.4] [           ] catalog-solr-external-provider  
solr-app-2.4.4 [           ] Catalog Provider to interface with  
an external Solr 4.7.2 Server  
[installed] [2.4.4] [           ] solr-app  
solr-app-2.4.4 [           ] The Solr Catalog Provider (SCP) is  
an implementation of the CatalogProvider interface using Apache Solr 4.7.2  
as a data store.\nThe SCP supports extensible metacards, fast/simple  
contextual searching, indexes xml attributes/CDATA sections/XML text  
elements, contains full XPath support, works with an embedded local Solr  
Server, and also works with a standalone Solr Server.:DDF Solr Catalog
```

Verify the DDF Solr Catalog Application bundle is Active for the External Solr Catalog Provider configuration.

DDF Solr Catalog Application active bundles for External Solr Catalog Provider configuration

```
ddf@local>list | grep -i solr  
[ 374] [Active] [           ] [           ] [     80] DDF :: Platform ::  
Solr :: Server :: Standalone War (2.6.1)  
[ 376] [Active] [Created] [           ] [     80] DDF :: Catalog ::  
Solr :: External :: Provider (2.4.4)
```

Configuration

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 defaults to a local Solr installation. 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	url	String	HTTP URL of the standalone, preconfigured Solr 4.x Server.	Connecting over HTTPS When connecting to an external Solr server over https, ensure that certificates are properly configured so that the two servers can securely communicate.	https://localhost:8993/solr	Yes
Force Auto Commit	forceAutoCommit	Boolean / Checkbox		Connecting over HTTP When connecting to an external Solr server over http, update the Web Context Policy Manager settings to remove /solr from the Authentication Types and add /solr to the White Listed Contexts. If the external Solr is running on another DDF, the same settings must be configured there as well.	Unchecked/False	No

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.

Standalone Solr Server

On This Page
<ul style="list-style-type: none"> • Overview • Usage • Installation <ul style="list-style-type: none"> • Prerequisites • Install • Verify • Configuration • Solr Standalone Server Meta Catalog Backup <ul style="list-style-type: none"> • Back Up Data from the Solr Server Standalone Metadata Catalog • Restore Data to the Solr Server Standalone Metadata Catalog • Suggestions for Managing Backup and Recovery

Overview

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.

Usage

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.

Installation

Prerequisites

Before a DDF instance can be configured as a Standalone Solr Server, the DDF Kernel must be running. It is also dependent on the DDF Platform, DDF Catalog, and DDF Solr Catalog applications. 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.

Install

By default, the feature for the Standalone Solr Server, simply named `solr`, which is contained in the DDF Platform application, is installed out of the box. The feature will copy the Solr configuration files in the distribution home directory then deploy the configured Solr war. Verification that the server started correctly can be performed by visiting the Solr Admin interface at <http://localhost:8181/solr>.

Verify

To verify the DDF is successfully configured to be a Standalone Solr Server, verify the appropriate feature is installed.

DDF Solr Catalog Application installed features for Standalone Solr configuration

```
ddf@local>features:list | grep solr
[uninstalled] [2.4.4] catalog-solr-embedded-provider
solr-app-2.4.4 Catalog Provider with locally
Embedded Solr Server, implemented using Solr 4.7.2.
[uninstalled] [2.4.4] catalog-solr-external-provider
solr-app-2.4.4 Catalog Provider to interface with
an external Solr 4.7.2 Server
[installed ] [2.4.4] solr-app
solr-app-2.4.4 The Solr Catalog Provider (SCP) is
an implementation of the CatalogProvider interface using Apache Solr 4.7.2
as a data store.\nThe SCP supports extensible metacards, fast/simple
contextual searching, indexes xml attributes/CDATA sections/XML text
elements, contains full XPath support, works with an embedded local Solr
Server, and also works with a standalone Solr Server.::DDF Solr Catalog
[installed ] [2.6.1] solr
platform-app-2.6.1 DDF Standalone Solr Server
```

Verify the appropriate bundle is Active for the Standalone Solr Server:

DDF Solr Catalog Application's active bundles for Standalone Solr Server configuration

```
ddf@local>list | grep -i solr
[ 374] [Active ] [ ] [ ] [ ] [ 80] DDF :: Platform :: Solr :: Server :: Standalone War (2.6.1)
```

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 'http://localhost:8181/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

<http://localhost:8181/solr>

Configuration

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.

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:

Solr Backup Response

```
<?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. All files within the index directory should be deleted.
5. Copy the files from the chosen backup directory into the index directory.
6. 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.

Extending DDF Solr Catalog Application

Overview

The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store.

This guide supports developers creating extensions of the existing framework.

Table of Contents

Securing DDF Solr Catalog Application

Overview

The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/>) as a data store. This guide covers implementations and protocols for enhancing security.

Table of Contents

DDF Solr Catalog Application Release Notes

Overview

This page contains the release notes for recent version of DDF Solr catalog application.

Versions

- Release Version: solr-2.4.4
- Release Version: solr-2.4.2

Release Version: solr-2.4.2

Contents

- New Capabilities
- Resolved Issues

- Known Issues
- App Prerequisites
- App Dependencies
- Third Party Licenses

New Capabilities

- Additional Temporal Filter Support
 - TEqual
 - After
 - PropertyGreaterThan (on dates)
 - PropertyLessThan (on dates)
 - PropertyGreaterThanOrEqualTo (on dates)
 - PropertyLessThanOrEqualTo (on dates)
- Solr Server moved to ddf-platform

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

App Prerequisites

Before the DDF Solr Application 2.4.2 can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.5.0 must be running
- the DDF Catalog App 2.5.0 must be running

App Dependencies

```
com.vividsolutions:jts:jar:1.12
commons-collections:commons-collections:jar:3.2.1
commons-io:commons-io:jar:2.1
commons-lang:commons-lang:jar:2.6
ddf.catalog.core:catalog-core-api-impl:jar:2.3.0
ddf.catalog.core:catalog-core-commons:jar:2.3.0
ddf.measure:measure-api:jar:2.3.0
ddf.thirdparty:lse:jar:0.3_4
javax.servlet:servlet-api:jar:2.5
joda-time:joda-time:jar:1.6.2
org.apache.solr:solr-core:jar:4.6.0
org.apache.solr:solr-solrj:jar:4.6.0
org.apache.solr:solr:war:4.6.0
org.codehaus.woodstox:wstx-asl:jar:3.2.7
org.parboiled:parboiled-java:jar:1.1.3
xerces:xercesImpl:jar:2.9.1
```

Third Party Licenses

[Third Party License File Notice](#)

Release Version: solr-2.4.4

Contents

- [New Capabilities](#)
- [Resolved Issues](#)
- [Known Issues](#)
- [App Prerequisites](#)
- [Third Party Licenses](#)

New Capabilities

- Secure Solr Admin by default

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------



Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
-----	---------	------	---------	---------	-----	----------	----------	----------	--------	------------

 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.

App Prerequisites

Before the DDF Solr Application 2.4.4 can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.6.1 must be running
- the DDF Catalog App 2.6.1 must be running

Third Party Licenses

[Third Party License File Notice](#)

DDF Spatial Application

Overview

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 Spatial Application Table of Contents

- [Managing 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.
 - [Installing DDF Spatial Application](#)
- [Integrating DDF Spatial Application](#)
 - [Integrating DDF with CSW](#)
 - [Integrating DDF with KML](#)
 - [Integrating DDF with WFS](#)
- [Extending DDF Spatial Application](#)
- [Securing DDF Spatial Application](#)
- [DDF Spatial Application Release Notes](#)
 - [Release Version: spatial-2.6.1](#)

Managing DDF Spatial Application

Overview

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.

- [Installing DDF Spatial Application](#)

Whitelist

While packages have been exported by the DDF Spatial Application, none of these exported packages are approved for use by third parties. These exported packages can be changed at any time.

Installing DDF Spatial Application

Overview

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.

On This Page

- Overview
- Prerequisites
- Installing
- Verifying
- Uninstalling
 - Reverting the Uninstall
- Upgrading

Prerequisites

Before the DDF Spatial Application can be installed:

- the [DDF Kernel](#) must be running
- the [DDF Platform Application](#) must be installed
- the [DDF Catalog Application](#) must be installed

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.

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

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.

Uninstalling

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

```
features:removeurl -u <DDF application's feature repository URL>

Example:   features:removeurl -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 `features:list` command output.

The repository URLs for installed applications can be obtained by entering:

```
features:listrepositories -u
```

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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf .catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

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.

```
features:install admin-app
```

4. Complete the steps in the Verify section above to determine if the upgrade was successful.

Integrating DDF Spatial Application

Overview

This guide supports integration of this application with external frameworks.

Integrating DDF with CSW

Overview

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

On This Page

- Overview
 - CSW v2.0.2 Endpoint
 - CSW v2.0.2 Source

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 (<http://www.opengeospatial.org/standards/cat>).

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 GetCapabilties 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 (<http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd>).

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=GetCapabilities
```

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 CSW-Discovery XML Schema (<http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd>).

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

```
<Capabilities xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ns2="http://www.opengis.net/ogc"
  xmlns:ns3="http://www.opengis.net/gml"
  xmlns:ns4="http://www.w3.org/1999/xlink"
  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/"
  xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0.2"
  xsi:schemaLocation="http://www.opengis.net/csw
  /ogc/csw/2.0.2/CSW-discovery.xsd">
  <ns5:ServiceIdentification>
    <ns5:Title>Catalog Service for the Web</ns5:Title>
    <ns5:Abstract>DDF CSW Endpoint</ns5:Abstract>
    <ns5:ServiceType>CSW</ns5:ServiceType>
    <ns5:ServiceTypeVersion>2.0.2</ns5:ServiceTypeVersion>
  </ns5:ServiceIdentification>
```

```
<ns5:ServiceProvider>
  <ns5:ProviderName>DDF</ns5:ProviderName>
  <ns5:ProviderSite/>
  <ns5:ServiceContact/>
</ns5:ServiceProvider>
<ns5:OperationsMetadata>
  <ns5:Operation name="GetCapabilities">
    <ns5:DCP>
      <ns5:HTTP>
        <ns5:Get ns4:href="http://localhost:8181/services/csw"/>
        <ns5:Post ns4:href="http://localhost:8181/services/csw">
          <ns5:Constraint name="PostEncoding">
            <ns5:Value>XML</ns5:Value>
          </ns5:Constraint>
        </ns5:Post>
      </ns5:HTTP>
    </ns5:DCP>
    <ns5:Parameter name="sections">
      <ns5:Value>ServiceIdentification</ns5:Value>
      <ns5:Value>ServiceProvider</ns5:Value>
      <ns5:Value>OperationsMetadata</ns5:Value>
      <ns5:Value>Filter_Capabilities</ns5:Value>
    </ns5:Parameter>
  </ns5:Operation>
<ns5:Operation name="DescribeRecord">
  <ns5:DCP>
    <ns5:HTTP>
      <ns5:Get ns4:href="http://localhost:8181/services/csw"/>
      <ns5:Post ns4:href="http://localhost:8181/services/csw">
        <ns5:Constraint name="PostEncoding">
          <ns5:Value>XML</ns5:Value>
        </ns5:Constraint>
      </ns5:Post>
    </ns5:HTTP>
  </ns5:DCP>
  <ns5:Parameter name="typeName">
    <ns5:Value>csw:Record</ns5:Value>
  </ns5:Parameter>
  <ns5:Parameter name="OutputFormat">
    <ns5:Value>text/xml</ns5:Value>
    <ns5:Value>application/xml</ns5:Value>
  </ns5:Parameter>
  <ns5:Parameter name="schemaLanguage">
    <ns5:Value>http://www.w3.org/XMLSchema</ns5:Value>
    <ns5:Value>http://www.w3.org/XML/Schema</ns5:Value>
    <ns5:Value>http://www.w3.org/2001/XMLSchema</ns5:Value>
    <ns5:Value>http://www.w3.org/TR/xmlschema-1/</ns5:Value>
  </ns5:Parameter>
</ns5:Operation>
<ns5:Operation name="GetRecords">
  <ns5:DCP>
    <ns5:HTTP>
      <ns5:Get ns4:href="http://localhost:8181/services/csw"/>
```

```
<ns5:Post ns4:href="http://localhost:8181/services/csw">
  <ns5:Constraint name="PostEncoding">
    <ns5:Value>XML</ns5:Value>
  </ns5:Constraint>
</ns5:Post>
</ns5:HTTP>
</ns5:DCP>
<ns5:Parameter name="ResultType">
  <ns5:Value>hits</ns5:Value>
  <ns5:Value>results</ns5:Value>
  <ns5:Value>validate</ns5:Value>
</ns5:Parameter>
<ns5:Parameter name="OutputFormat">
  <ns5:Value>text/xml</ns5:Value>
  <ns5:Value>application/xml</ns5:Value>
</ns5:Parameter>
<ns5:Parameter name="OutputSchema">
  <ns5:Value>http://www.opengis.net/cat/csw/2.0.2</ns5:Value>
</ns5:Parameter>
<ns5:Parameter name="typeNames">
  <ns5:Value>csw:Record</ns5:Value>
</ns5:Parameter>
<ns5:Parameter name="ConstraintLanguage">
  <ns5:Value>Filter</ns5:Value>
</ns5:Parameter>
</ns5:Operation>
<ns5:Operation name="GetRecordById">
  <ns5:DCP>
    <ns5:HTTP>
      <ns5:Get ns4:href="http://localhost:8181/services/csw"/>
      <ns5:Post ns4:href="http://localhost:8181/services/csw">
        <ns5:Constraint name="PostEncoding">
          <ns5:Value>XML</ns5:Value>
        </ns5:Constraint>
      </ns5:Post>
    </ns5:HTTP>
  </ns5:DCP>
  <ns5:Parameter name="OutputSchema">
    <ns5:Value>http://www.opengis.net/cat/csw/2.0.2</ns5:Value>
  </ns5:Parameter>
  <ns5:Parameter name="OutputFormat">
    <ns5:Value>text/xml</ns5:Value>
    <ns5:Value>application/xml</ns5:Value>
  </ns5:Parameter>
  <ns5:Parameter name="ResultType">
    <ns5:Value>hits</ns5:Value>
    <ns5:Value>results</ns5:Value>
    <ns5:Value>validate</ns5:Value>
  </ns5:Parameter>
  <ns5:Parameter name="ElementSetName">
    <ns5:Value>brief</ns5:Value>
    <ns5:Value>summary</ns5:Value>
    <ns5:Value>full</ns5:Value>
```

```
</ns5:Parameter>
</ns5:Operation>
<ns5:Parameter name="service">
  <ns5:Value>CSW</ns5:Value>
</ns5:Parameter>
<ns5:Parameter name="version">
  <ns5:Value>2.0.2</ns5:Value>
</ns5:Parameter>
</ns5:OperationsMetadata>
<ns2:Filter_Capabilities>
  <ns2:Spatial_Capabilities>
    <ns2:GeometryOperands>
      <ns2:GeometryOperand>ns3:Point</ns2:GeometryOperand>
      <ns2:GeometryOperand>ns3:Polygon</ns2:GeometryOperand>
      <ns2:GeometryOperand>ns3:LineString</ns2:GeometryOperand>
    </ns2:GeometryOperands>
    <ns2:SpatialOperators>
      <ns2:SpatialOperator name="BBOX" />
      <ns2:SpatialOperator name="Beyond" />
      <ns2:SpatialOperator name="Contains" />
      <ns2:SpatialOperator name="Crosses" />
      <ns2:SpatialOperator name="Disjoint" />
      <ns2:SpatialOperator name="DWithin" />
      <ns2:SpatialOperator name="Intersects" />
      <ns2:SpatialOperator name="Overlaps" />
      <ns2:SpatialOperator name="Touches" />
      <ns2:SpatialOperator name="Within" />
    </ns2:SpatialOperators>
  </ns2:Spatial_Capabilities>
  <ns2:Scalar_Capabilities>
    <ns2:LogicalOperators/>
    <ns2:ComparisonOperators>
      <ns2:ComparisonOperator>Between</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>NullCheck</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Like</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>EqualTo</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Greater Than</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Greater Than Equal To</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Less Than</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Less Than Equal To</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Equal To</ns2:ComparisonOperator>
      <ns2:ComparisonOperator>Not Equal To</ns2:ComparisonOperator>
    </ns2:ComparisonOperators>
  </ns2:Scalar_Capabilities>
  <ns2:Id_Capabilities>
    <ns2:EID/>
  </ns2:Id_Capabilities>
```

```
</ns2:Filter_Capabilities>  
</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-uri]),xmlns([prefix=namespace-uri])*
- Service – The service being used, in this case it is fixed at CSW.
- Version – The version of the service being used (2.0.2).
- OutputFormat – The requester wants the response to be in this intended output. Currently, only one format is supported (application/xml). If this parameter is supplied, it is validated against the known type. If this parameter is not supported, it passes through and returns the XML response upon success.
- OutputSchema – The 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&NAMESPACE=xmlns(http://www.opengis.net/cat/csw/2.0.2)&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2
```

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"  
    outputSchema="http://www.opengis.net/cat/csw/2.0.2"  
    xmlns="http://www.opengis.net/cat/csw/2.0.2">  
</DescribeRecord>
```

DescribeRecordResponse

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

Sample - DescribeRecordResponse

```
<DescribeRecordResponse xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2/CSW-discovery.xsd"  
    xmlns="http://www.opengis.net/cat/csw/2.0.2"  
    xmlns:ns2="http://www.opengis.net/ogc"  
    xmlns:ns3="http://www.opengis.net/gml"  
    xmlns:ns4="http://www.w3.org/1999/xlink"
```

```

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/"
xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SchemaComponent targetNamespace="http://www.opengis.net/cat/csw/2.0.2"
schemaLanguage="http://www.w3.org/XML/Schema">
        <xsd:schema elementFormDefault="qualified" id="csw-record"
targetNamespace="http://www.opengis.net/cat/csw/2.0.2" version="2.0.2"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:dct="http://purl.org/dc/terms/"
xmlns:ows="http://www.opengis.net/ows">
            <xsd:annotation>
                <xsd:appinfo>

<dc:identifier>http://schemas.opengis.net/csw/2.0.2/record.xsd</dc:identifier>
                </xsd:appinfo>
                <xsd:documentation xml:lang="en">This schema defines the basic
record types that must be supported
                    by all CSW implementations. These correspond to full, summary, and
                    brief views based on DCMI metadata terms.</xsd:documentation>
            </xsd:annotation>
            <xsd:import namespace="http://purl.org/dc/terms/">
schemaLocation="rec-dcterms.xsd"/>
            <xsd:import namespace="http://purl.org/dc/elements/1.1/">
schemaLocation="rec-dcmes.xsd"/>
            <xsd:import namespace="http://www.opengis.net/ows">
schemaLocation="../../ows/1.0.0/owsAll.xsd"/>
            <xsd:element abstract="true" id="AbstractRecord"
name="AbstractRecord" type="csw:AbstractRecordType"/>
            <xsd:complexType abstract="true" id="AbstractRecordType"
name="AbstractRecordType"/>
            <xsd:element name="DCMIRecord"
substitutionGroup="csw:AbstractRecord" type="csw:DCMIRecordType" />
            <xsd:complexType name="DCMIRecordType">
                <xsd:annotation>
                    <xsd:documentation xml:lang="en">This type encapsulates all
                    of the standard DCMI metadata terms,
                        including the Dublin Core refinements; these terms may be
                    mapped
                        to the profile-specific information model.</xsd:documentation>
                </xsd:annotation>
                <xsd:complexContent>
                    <xsd:extension base="csw:AbstractRecordType">
                        <xsd:sequence>
                            <xsd:group ref="dct:DCMI-terms" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>
        </xsd:schema>
    </SchemaComponent>

```

```

        </xsd:complexType>
        <xsd:element name="BriefRecord"
substitutionGroup="csw:AbstractRecord" type="csw:BriefRecordType"/>
        <xsd:complexType final="#all" name="BriefRecordType">
            <xsd:annotation>
                <xsd:documentation xml:lang="en">This type defines a brief
representation of the common record
                    format. It extends AbstractRecordType to include only the
                    dc:identifier and dc:type properties.</xsd:documentation>
            </xsd:annotation>
            <xsd:complexContent>
                <xsd:extension base="csw:AbstractRecordType">
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1"
ref="dc:identifier"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="1"
ref="dc:title"/>
                            <xsd:element minOccurs="0" ref="dc:type"/>
                            <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="ows:BoundingBox"/>
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>
            <xsd:element name="SummaryRecord"
substitutionGroup="csw:AbstractRecord" type="csw:SummaryRecordType"/>
            <xsd:complexType final="#all" name="SummaryRecordType">
                <xsd:annotation>
                    <xsd:documentation xml:lang="en">This type defines a summary
representation of the common record
                    format. It extends AbstractRecordType to include the core
                    properties.</xsd:documentation>
                </xsd:annotation>
                <xsd:complexContent>
                    <xsd:extension base="csw:AbstractRecordType">
                        <xsd:sequence>
                            <xsd:element maxOccurs="unbounded" minOccurs="1"
ref="dc:identifier"/>
                            <xsd:element maxOccurs="unbounded" minOccurs="1"
ref="dc:title"/>
                                <xsd:element minOccurs="0" ref="dc:type"/>
                                <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="dc:subject"/>
                            <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="dc:format"/>
                                <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="dc:relation"/>
                            <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="dct:modified"/>
                            <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="dct:abstract"/>
                            <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="dct:spatial"/>

```

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

```
</DescribeRecordResponse>
```

GetRecords Operation

The GetRecords operation is the principal means of searching the catalog. The matching entries may be included with the response. The client may assign a requestId (absolute URI). A distributed search is performed if the DistributedSearch element is present and the catalog is a member of a federation. Profiles may allow alternative query expressions. There are two types of request types: one for GET and one for POST. Each request has the following common data parameters:

- Namespace – In POST operations, namespaces are defined in the XML. In GET operations, namespaces are defined in a comma-separated list of the form xmlns([prefix=]namespace-url),xmlns([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.
- 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&outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/csw/2.0.2&NAMESPACE=xmlns(csw=http://www.opengis.net/cat/csw/2.0.2)&resultType=results&typeNames=csw:Record&ElementSetName=brief&ConstraintLanguage=CQL_TEXT&constraint=AnyText Like '%25'
```

GetRecords HTTP POST

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

GetRecords XML Request

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

GetRecordsResponse

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

Sample - GetRecordsResponse

```
<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>182fb33103414e5ccb06f8693b526239</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>
```

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. 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-uri)(,xmlns([prefix=]namespace-uri))*
- 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.
- 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=fd7ff1535dfe47db8793b550d4170424,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>182fb33103414e5ccb06f8693b526239</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>
```

CSW Record to Metocard Mapping

CSW Record Field	Metocard Field	Brief Record	Summary Record	Record
dc:title	title	1-n	1-n	0-n
dc:creator				0-n
dc:subject			0-n	0-n
dc:description				0-n
dc:publisher				0-n
dc:contributor				0-n
dc:date	modified			0-n
dc:type	metadata-content-type	0-1	0-1	0-n
dc:format			0-n	0-n
dc:identifier	id	1-n	1-n	0-n
dc:source	source-id			0-n
dc:language				0-n
dc:relation			0-n	0-n
dc:coverage				0-n
dc:rights				0-n
dct:abstract			0-n	0-n
dct:accessRights				0-n
dct:alternative	title			0-n
dct:audience				0-n
dct:available				0-n
dct:bibliographicCitation	id			0-n
dct:conformsTo				0-n
dct:created	created			0-n
dct:dateAccepted	effective			0-n
dct:Copyrighted	effective			0-n
dct:dateSubmitted	modified			0-n
dct:educationLevel				0-n
dct:extent				0-n
dct:hasFormat				0-n
dct:hasPart				0-n
dct:hasVersion				0-n
dct:isFormatOf				0-n
dct:isPartOf				0-n
dct:isReferencedBy				0-n
dct:isReplacedBy				0-n
dct:isRequiredBy				0-n
dct:issued	modified			0-n
dct:isVersionOf				0-n
dct:license				0-n
dct:mediator				0-n
dct:medium				0-n

dct:modified	modified		0-n	0-n
dct:provenance				0-n
dct:references				0-n
dct:replaces				0-n
dct:requires				0-n
dct:rightsHolder				0-n
dct:spatial	location		0-n	0-n
dct:tableOfContents				0-n
dct:temporal	effective + " - " + expiration			0-n
dct:valid	expiration			0-n
ows:BoundingBox		0-n	0-n	0-n

Install and Uninstall

The CSW endpoint can be installed and uninstalled using the normal processes described in the [Configuration](#) section.

Configuration

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

Known Issues

None

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.

Using

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

Installing and Uninstalling

The CSW 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 CSW source are accessed from the **CSW Federated Source** Configuration in the Web Console or Admin Console.

Configure the CSW Source

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	id	String	Unique Name of this Source.	CSW	Yes
CSW URL	cswUrl	String	URL to the Catalogue Services for the Web site that will be queried by this source		Yes
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 the CN Check for the server certificate	false	Yes
Force Longitude/Latitude coordinate order	isLonLatOrder	Boolean	Force Longitude/Latitude coordinate order	false	Yes

Use posList in LlinearRing		Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	Yes
Effective Date maps to	effectiveDateMapping	String	<p>The field in the CSW Record that should be mapped to a Metocard's effective date.</p> <p>This field will have a default mapping, but the user can change this to be any date formatted field in a CSW Record. Relevant CSW fields would include dateSubmitted,</p> <p>created and modified. If no value is specified, the default value of created will be used.</p> <p>Note that the same CSW Record field cannot be used more than once in these date mapping properties.</p>	created	No
Created Date maps to	createdDateMapping	String	<p>The field in the CSW Record that should be mapped to a Metocard's created date.</p> <p>This field will have a default mapping, but the user can change this to be any date formatted field in a CSW Record. Relevant CSW fields would include dateSubmitted,</p> <p>created and modified. If no value is specified, the default value of dateSubmitted will be used.</p> <p>Note that the same CSW Record field cannot be used more than once in these date mapping properties.</p>	dateSubmitted	No
Modified Date maps to	modifiedDateMapping	String	<p>The field in the CSW Record that should be mapped to a Metocard's effective date.</p> <p>This field will have a default mapping, but the user can change this to be any date formatted field in a CSW Record. Relevant CSW fields would include dateSubmitted,</p> <p>created and modified. If no value is specified, the default value of created will be used.</p> <p>Note that the same CSW Record field cannot be used more than once in these date mapping properties.</p>	modified	No
Resource URI maps to	resourceUriMapping	String	CSW field to map to Metocard's resource URI to retrieve product associated with the CSW record.	source	No
Thumbnail maps to	thumbnailMapping	String	CSW field to map to Metocard's thumbnail URI to retrieve thumbnail data associated with the CSW record.	references	No
Content type maps to	contentTypemapping	String	CSW field to map to Metocard's content type.	type	No
Content Types	contentTypeNames	List of Strings	A list of content types that can be searched on. The user can add any content types to the list, e.g., doc, or even wildcarded types. The list of content types currently in the CSW source will be added to this list during configuration when the GetCapabilities response is returned.		No

Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1)	5	Yes
Connection Timeout	connectionTimeout	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	30000	Yes
Receive Timeout	receiveTimeout	Integer	Amount of time (in milliseconds) to attempt to establish a connection before timing out.	60000	Yes
Output schema	outputSchema	String	Output Schema	http://www.opengis.net/cat/csw/2.0.2	Yes
Force CQL Text as the Query Language	isCqlForced	Boolean	Force CQL Text	false	Yes
Forced Spatial Filter Type	forceSpatialFilter	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	None	No

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.

Integrating DDF with KML

Overview

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

On This Page

- Overview
 - KML Network Link Endpoint
 - KML Query Response Transformer
 - KML Metacard Transformer
 - KML Style Mapper

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.)

 Unknown macro: 'plantuml'

Using

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

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

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

Example Output

```

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

```

When configured to do so, the KML endpoint can serve up a KML style document. The request below will return the configured KML style document. For more information on how to configure the KML style document, see [Configuration](#).

```
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. For more information on how to configure the KML Icons document, see [Configuration](#).

```

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

```

Installing and Uninstalling

The `spatial-kml-networklinkendpoint` feature is installed by default with the [Spatial App](#).

Configuring

This KML Network Link endpoint has the ability to serve up custom KML style documents and Icons to be used within that document.

The KML style document must be a valid XML document containing a KML style.

The KML Icons should be placed in a single level directory and must be an image type (png, jpg, tif, etc.).

The Description will be displayed as a pop-up from the root network link on Google Earth. This may contain the general purpose of the network and URLs to external resources.

Configurable Properties

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

Known Issues

None.

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.

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>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
          &lt;td
            width="400">$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
      </BalloonStyle>
    </Style>
  </Document>
</kml>
```

```
</Style>
<Style id="bluehighlight">
  <LabelStyle>
    <scale>1.0</scale>
  </LabelStyle>
  <LineStyle>
    <color>99ff0000</color>
    <width>6.0</width>
  </LineStyle>
  <PolyStyle>
    <color>99ff0000</color>
    <fill xsi:type="xs:boolean"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
  </PolyStyle>
  <BalloonStyle>

    <text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
      &lt;td
        width="400"&gt;$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
    </BalloonStyle>
  </Style>
  <StyleMap id="default">
    <Pair>
      <key>normal</key>
      <styleUrl>#bluenormal</styleUrl>
    </Pair>
    <Pair>
      <key>highlight</key>
      <styleUrl>#bluehighlight</styleUrl>
    </Pair>
  </StyleMap>
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description>&lt;!DOCTYPE html&gt;
      &lt;html&gt;
        &lt;head&gt;
          &lt;meta content="text/html; charset=windows-1252"
            http-equiv="content-type"&gt;
            &lt;style media="screen" type="text/css"&gt;
              .label {
                font-weight: bold
              }
              .linkTable {
                width: 100%
              }
              .thumbnailDiv {
                text-align: center
              }
              img {
                max-width: 100px;
                max-height: 100px;
                border-style:none
              }
            &lt;/style&gt;
          &lt;/head&gt;
          &lt;body&gt;
            <h3>Multi Polygon</h3>
            <table border="1" style="width:100%; border-collapse: collapse;">
              <tr>
                <td style="width: 10%; padding: 5px;"><a href="#">Multi Polygon</a>
                <td style="width: 10%; padding: 5px;"><a href="#">Multi Polygon</a>
              </tr>
            </table>
            <img alt="A multi-colored polygon representing a complex geographical area." data-bbox="100 100 800 500"/>
          &lt;/body&gt;
        &lt;/html&gt;
      &lt;/description&gt;
    </Placemark>

```

```
        }
    &lt;/style&gt;
    &lt;/head&gt;
    &lt;body&gt;
        &lt;div class="thumbnailDiv"&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource"&gt;&lt;img alt="Thumbnail" src="data:image/jpeg; charset=utf-8; base64, CA=="&gt;&lt;/a&gt;&lt;/div&gt;
        &lt;table&gt;
            &lt;tr&gt;
                &lt;td class="label"&gt;Source:&lt;/td&gt;
                &lt;td&gt;ddf.distribution&lt;/td&gt;
            &lt;/tr&gt;
            &lt;tr&gt;
                &lt;td class="label"&gt;Created:&lt;/td&gt;
                &lt;td&gt;Wed Oct 30 09:46:29 MDT 2013&lt;/td&gt;
            &lt;/tr&gt;
            &lt;tr&gt;
                &lt;td class="label"&gt;Effective:&lt;/td&gt;
                &lt;td&gt;2014-01-07T14:48:47-0700&lt;/td&gt;
            &lt;/tr&gt;
        &lt;/table&gt;
        &lt;table class="linkTable"&gt;
            &lt;tr&gt;
                &lt;td&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=html"&gt;View Details...&lt;/a&gt;&lt;/td&gt;
                &lt;td&gt;&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource"&gt;Download...&lt;/a&gt;&lt;/td&gt;
            &lt;/tr&gt;
        &lt;/table&gt;
        &lt;/body&gt;
    &lt;/html&gt;
</description>
<TimeSpan>
    <begin>2014-01-07T21:48:47</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>
        </Polygon>
    </MultiGeometry>

```

```
</Polygon>
<Polygon>
  <outerBoundaryIs>
    <LinearRing>
      <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0
100.0,0.0 100.2,0.2 100.8,0.2
          100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
    </LinearRing>
  </outerBoundaryIs>
</Polygon>
</MultiGeometry>
</MultiGeometry>
```

```
</Placemark>
</Document>
</kml>
```

Installing and Uninstalling

The `spatial-kml-transformer` feature is installed by default with the [Spatial App](#).

Configuring

None

Implementation Details

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

Known Issues

None.

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.

Using

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

```
http://localhost:8181/services/catalog/0103c77e66d9428d8f48fab939da528e?tr
ansform=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">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description>&lt;!DOCTYPE html&gt;
&lt;html&gt;
  &lt;head&gt;
    &lt;meta content="text/html; charset=windows-1252"
http-equiv="content-type"&gt;
    &lt;style media="screen" type="text/css"&gt;
      .label {
        font-weight: bold
      }
      .linkTable {
        width: 100%
      }
    
```

```
.thumbnailDiv {
    text-align: center
}
img {
    max-width: 100px;
    max-height: 100px;
    border-style:none
}
</style>
</head>
<body>
    <div class="thumbnailDiv">&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource">&lt;img alt="Thumnail" src="data:image/jpeg; charset=utf-8; base64, CA=="&gt;&lt;/a&gt;&lt;/div&gt;
    <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>&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=html">View Details...&lt;/a&gt;&lt;/td>;
            <td>&lt;a href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48fab939da528e?transform=resource">Download...&lt;/a&gt;&lt;/td>;
        </tr>
    </table>
</body>
</html>
</description>
<TimeSpan>
    <begin>2014-01-07T21:58:16</begin>
</TimeSpan>
<Style id="bluenormal">
    <LabelStyle>
        <scale>0.0</scale>
    </LabelStyle>
    <LineStyle>
        <color>33ff0000</color>
```

```
<width>3.0</width>
</LineStyle>
<PolyStyle>
<color>33ff0000</color>
<fill xsi:type="xs:boolean"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
</PolyStyle>
<BalloonStyle>

<text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
t;&lt;td
width="400"&gt;${[description]}&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
</BalloonStyle>
</Style>
<Style id="bluehighlight">
<LabelStyle>
<scale>1.0</scale>
</LabelStyle>
<LineStyle>
<color>99ff0000</color>
<width>6.0</width>
</LineStyle>
<PolyStyle>
<color>99ff0000</color>
<fill xsi:type="xs:boolean"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
</PolyStyle>
<BalloonStyle>

<text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
t;&lt;td
width="400"&gt;${[description]}&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
</BalloonStyle>
</Style>
<StyleMap id="default">
<Pair>
<key>normal</key>
<styleUrl>#bluenormal</styleUrl>
</Pair>
<Pair>
<key>highlight</key>
<styleUrl>#bluehighlight</styleUrl>
</Pair>
</StyleMap>
<MultiGeometry>
<Point>
<coordinates>102.0,2.0</coordinates>
</Point>
<MultiGeometry>
<Polygon>
<outerBoundaryIs>
```

```
<LinearRing>
  <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0
102.0,2.0</coordinates>
</LinearRing>
</outerBoundaryIs>
</Polygon>
<Polygon>
  <outerBoundaryIs>
    <LinearRing>
      <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0
100.0,0.0 100.2,0.2 100.8,0.2
      100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
    </LinearRing>
  </outerBoundaryIs>
</Polygon>
</MultiGeometry>
```

```

</MultiGeometry>
</Placemark>
</kml>

```

Installing and Uninstalling

The `spatial-kml-transformer` feature is installed by default with the [Spatial App](#).

Configuring

None

Implementation Details

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

Known Issues

None.

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 (<https://developers.google.com/kml/documentation/kmlreference#styleurl>).

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.

Configuring

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

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Attribute Name	attributeName	String	The name of the metocard attribute to match against (e.g., title, metadata-content-type).		Yes
Attribute Value	attributeValue	String	The value of the metocard attribute.		Yes
Style URL	styleUrl	String	The full qualified URL to the KML style (e.g., http://example.com/styles#myStyle).		Yes

Example Values

Attribute Name = title

Attribute Value = MutliPolygon

Style URL = <http://example.com/kml/style#sampleStyle>

Example Output

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns:ns2="http://www.google.com/kml/ext/2.2"
      xmlns="http://www.opengis.net/kml/2.2"
      xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">

```

```
xmlns:ns3="http://www.w3.org/2005/Atom">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description>&lt;!DOCTYPE html&gt;
&lt;html&gt;
  &lt;head&gt;
    &lt;meta content="text/html; charset=windows-1252"
http-equiv="content-type"&gt;
    &lt;style media="screen" type="text/css"&gt;
      .label {
        font-weight: bold
      }
      .linkTable {
        width: 100%
      }
      .thumbnailDiv {
        text-align: center
      }
      img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    &lt;/style&gt;
  &lt;/head&gt;
  &lt;body&gt;
    &lt;div class="thumbnailDiv"&gt;&lt;a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103
c77e66d9428d8f48fab939da528e?transform=resource"&gt;&lt;img alt="Thumbnail"
src="data:image/jpeg;charset=utf-8;base64, CA=="&gt;&lt;/a&gt;&lt;/div&gt;
    &lt;table&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Source:&lt;/td&gt;
        &lt;td&gt;ddf.distribution&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Created:&lt;/td&gt;
        &lt;td&gt;Wed Oct 30 09:46:29 MDT 2013&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Effective:&lt;/td&gt;
        &lt;td&gt;2014-01-07T14:58:16-0700&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
    &lt;table class="linkTable"&gt;
      &lt;tr&gt;
        &lt;td&gt;&lt;a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103
c77e66d9428d8f48fab939da528e?transform=html"&gt;View
Details...&lt;/a&gt;&lt;/td&gt;
        &lt;td&gt;&lt;a
href="http://localhost:8181/services/catalog/sources/ddf.distribution/0103
c77e66d9428d8f48fab939da528e?transform=resource"&gt;Download...&lt;/a&gt;&
```

```
lt;/td&gt;
    &lt;/tr&gt;
    &lt;/table&gt;
    &lt;/body&gt;
&lt;/html&gt;
</description>
<TimeSpan>
    <begin>2014-01-07T21:58:16</begin>
</TimeSpan>
<styleUrl>http://example.com/kml/style#sampleStyle</styleUrl>
<MultiGeometry>
    <Point>
        <coordinates>102.0,2.0</coordinates>
    </Point>
    <MultiGeometry>
        <Polygon>
            <outerBoundaryIs>
                <LinearRing>
                    <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0
102.0,2.0</coordinates>
                </LinearRing>
            </outerBoundaryIs>
        </Polygon>
        <Polygon>
            <outerBoundaryIs>
                <LinearRing>
                    <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0
100.0,0.0 100.2,0.2 100.8,0.2
                    100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
                </LinearRing>
            </outerBoundaryIs>
        </Polygon>
    </MultiGeometry>
```

```

</MultiGeometry>
</Placemark>
</kml>

```

Installing and Uninstalling

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

Implementation Details

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

Known Issues

None.

Integrating DDF with WFS

Overview

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.

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

On This Page

- Overview
- Working with WFS Sources
 - WFS Features
- WFS v1.0.0 Source
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues
- WFS v2.0.0 Source
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues
- Mapping WFS Feature Properties to Metocard Attributes
 - Using
 - Installing and Uninstalling
 - Configuring
 - Known Issues

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.

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 `Converter` (<http://xstream.codehaus.org/javadoc/com/thoughtworks/xstream/converters/Converter.html>) interface provided by the XStream (<http://xstream.codehaus.org/>) 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.

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

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

2. 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;  
    }  
}
```

3. 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());  
}
```

4. Export theConverterFactory 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.locationtech.geogig.catalog.common.converter.factory.FeatureConverterFactory"/>
</blueprint>
```

For more information about registering services, see [Working with OSGi](#).

WFS v1.0.0 Source

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

Using

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

Also see [Working with WFS Sources](#).

Installing and Uninstalling

The WFS 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 WFS Source are accessed from the **WFS Federated Source** Configuration in the Admin Console.

Configuring WFS Source

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	id	String	Unique name of the Source.	WFS_v1_0_0	Yes
WFS URL	wfsUrl	String	URL to the Web Feature Service (WFS) that will be queried by this source (see below).		Yes
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.		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
Receive Timeout	receiveTimeout	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	Yes

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.

As an example, GeoServer 2.5 syntax might look like:

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

In this case, the WFS URL would be

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

Known Issues

None.

WFS v2.0.0 Source

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

Using

Use the WFS Source if querying a WFS version 2.0.0 compliant service.

Also see [Working with WFS Sources](#).

Installing and Uninstalling

The WFS 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 WFS 2.0.0 Source are accessed from the **WFS 2.0.0 Federated Source** Configuration in the Admin Console.

Configuring WFS 2.0.0 Source

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Source ID	id	String	Unique name of the source	WFS_v2_0_0	Yes
WFS URL	wfsUrl	String	URL to the endpoint implementing the Web Feature Service (WFS) 2.0.0 spec.		Yes
Disable CN Check	disableCnCheck	Boolean	Disable CN Check for the server certificate. This should only be used when testing.	false	Yes
Coordinate Order	coordinateOrder	String	Coordinate order that remote source expects and returns spatial data in.	Lat/Lon	Yes
Disable Sorting	disableSorting	Boolean	When selected, the system will not specify sort criteria with the query. This should only be used if the remote source is unable to handle sorting even when the capabilities states 'ImplementsSorting' is supported.	false	Yes
Username	username	String	Username for the WFS service.		No

Password	password	String	Password for the WFS service.		No
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
Connection Timeout	connectionTimeout	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds	30000	Yes
Receive Timeout	receiveTimeout	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	Yes

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.

As an example, GeoServer 2.5 syntax might look like:

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

In this case, the WFS URL would be

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

Known Issues

None.

Mapping WFS Feature Properties to Metocard Attributes

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

Using

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

Also see [Working with WFS Sources](#) for more advanced use cases.

Installing and Uninstalling

The WFS MetocardMapper 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 WFS MetocardMapper are accessed from the **Metocard to WFS Feature Map** Configuration in the Admin Console.

Configuring WFS MetocardMapper

Configurable Properties

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

Metocard Attribute to WFS Feature Property Mapping	metocardAttrToFeaturePropMap	String	Metocard Attribute to WFS Feature Property Mapping. Format is metocardAttribute=featureProperty		Yes
Temporal Sort By Feature Property	sortByTemporalFeatureProperty	String	When Sorting Temporally, Sort By This Feature Property.		No
Relevance Sort By Feature Property	sortByRelevanceFeatureProperty	String	When Sorting By Relevance, Sort By This Feature Property.		No
Distance Sort By Feature Property	sortByDistanceFeatureProperty	String	When Sorting By Distance, Sort By This Feature Property.		No

Example Configuration

There are two ways to configure the MetocardMapper, one is to use the Configuration Admin available via the Web Admin Console. Additionally, a feature.xml file can be created and copied into the "deploy" directory. The following shows how to configure the MetocardMapper 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 featurePropToMetocardAttrMap line through the use of comma as a delimiter.

Unknown Attachment

Below is an example of a MetocardMapper configuration within a feature.xml file:

```
<feature name="geoserver-states" version="${project.version}" 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.MetocardMapper-geoserver.http://www.openplans.org/topp.states">
        featureType = {http://www.openplans.org/topp}states
        service.factoryPid =
        org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetocardMapper
            featurePropToMetocardAttrMap = states.STATE_NAME=title
    </config>
</feature>
```

Known Issues

None.

Extending DDF Spatial Application

Overview

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 developers creating extensions of the existing framework.

Securing DDF Spatial Application

Overview

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 covers implementations and protocols for enhancing security.

DDF Spatial Application Release Notes

- Release Version: spatial-2.6.1

Release Version: spatial-2.6.1

Contents

- New Capabilities
- Resolved Issues
- Known Issues
- App Prerequisites
- Third Party Licenses

New Capabilities

- CSW Metocard Transformer
- CSW Source and Endpoint support Metocard XML

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
<p> Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</p>										

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
<p> Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</p>										

App Prerequisites

Before the DDF Spatial Application 2.6.1 can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.6.1 must be running

Third Party Licenses

[Third Party License File Notice](#)

DDF Standard Search UI

Overview

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

DDF Standard Search UI Table of Contents

- Using DDF Standard Search UI — DDF Standard Search UI _user-guide-intro
- Managing DDF Standard Search UI — The Standard Search UI is a user interface that enables users to search a catalog and associated sites for content and metadata.
 - Installing DDF Standard Search UI
 - Troubleshooting DDF Standard Search UI
- Integrating DDF Standard Search UI
- Extending DDF Standard Search UI
- Securing DDF Standard Search UI
- DDF Standard Search UI Release Notes
 - Release Version: ui-2.6.1

Using DDF Standard Search UI

On This Page

- Overview
- Search
 - Search Criteria
 - Results
 - Enhanced Search
 - Record Summary
 - Record Details
 - Record Actions
 - Save a Search
- Workspaces
 - Create a Workspace
- Notifications and Activities
 - Notifications
 - Activities
 - Downloads
- Uploading New Resources
- Maps
 - 3D View
 - 2D View
 - Columbus View
 - Known Issues

Overview

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

The right pane consists of a map. The Help page is accessed from the Search UI by clicking the help icon (



) in the menu bar then **Help Topics**.

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 search criteria to a workspace, so the query can be executed again.

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

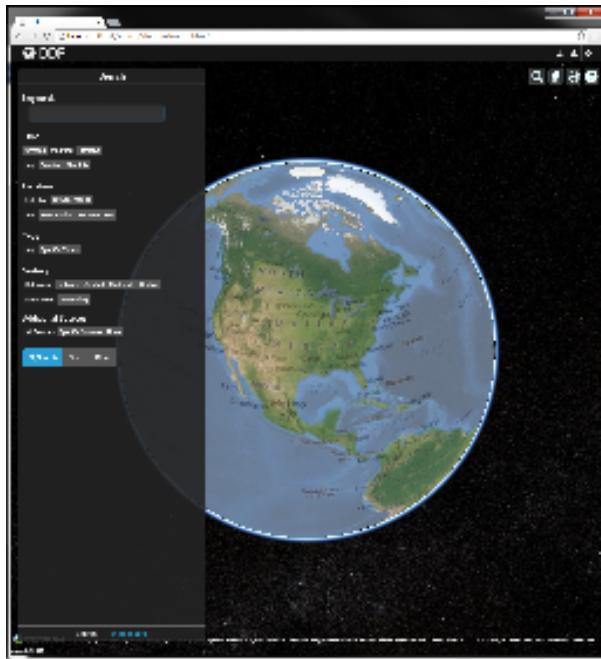
workspaces feature uses the same search criteria that are provided in the Search tab, and it also allows the user to create custom searches that can be saved for later execution. The right-side pane displays a map that, when records are found, shows the location of where the record was retrieved.

Navigate to <http://localhost:8181/search> to open the Standard Search UI.

Search Criteria

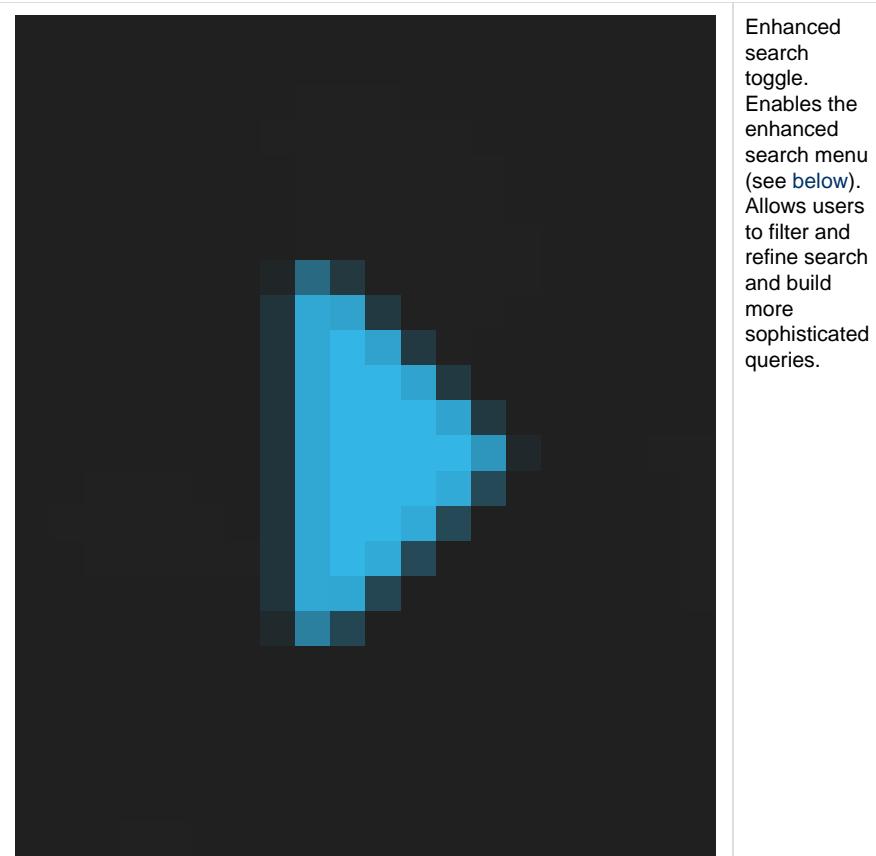
The Search UI queries a Catalog using the following configurable criteria.

Criteria	Description
Keywords	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 Metocard.CONTENT_TYPE values.
Sorting	Sort results by relevance, distance, created time, modified time or effective time.
Additional Sources	Select All Sources or Specific Sources .
All Sources	Create an enterprise search. All federations are queried.
Specific Sources	Search a specific source(s). If a source is unavailable, it is displayed in red text.



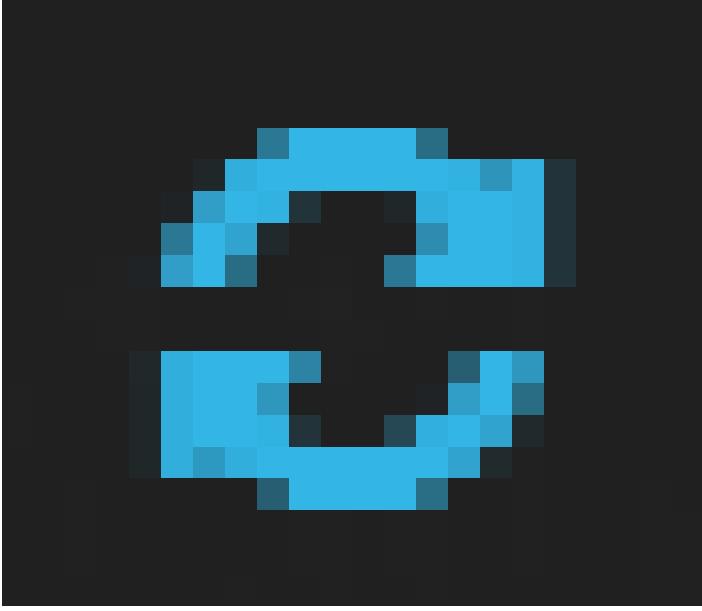
Results

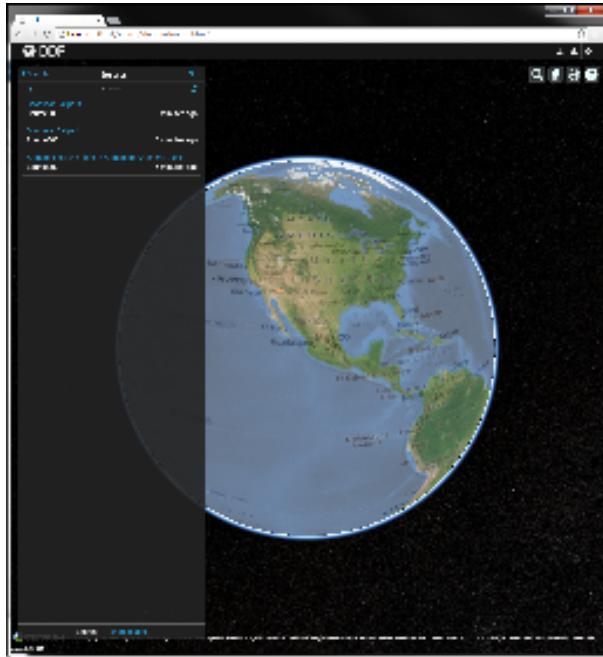
Item	Description



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.

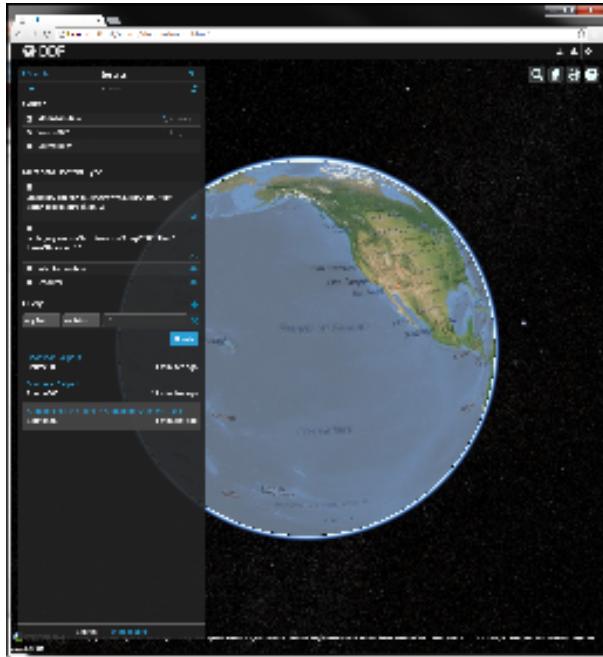
<p>Records list</p> 	<p>Shows the results of the search. The following information is displayed for each record:</p> <p>Title – The title of the record is displayed in blue text. Select the title of the record to view more details.</p> <p>Source – The gray text displayed below the record title is the data source and the amount of time that has passed since the record was created.</p>
	<p>Refreshes the list of results. The most relevant records continue to be displayed at the top of the results list.</p>



Enhanced Search

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

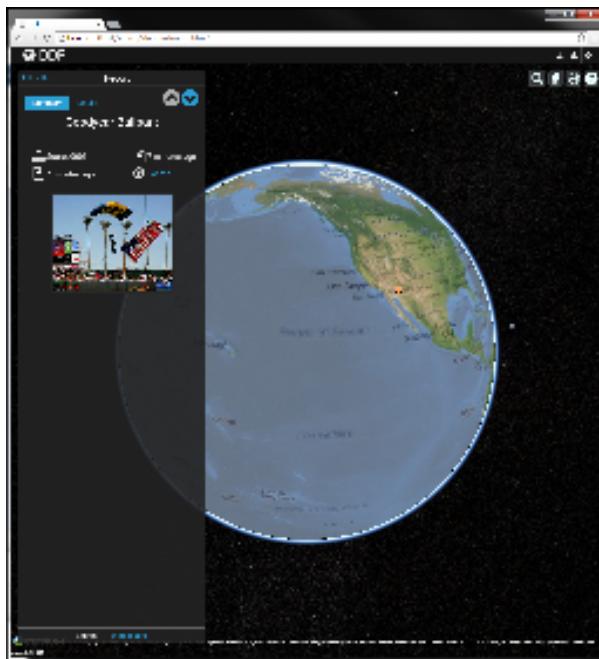
Item	Description
Source	List of all sources searched with check boxes to allow users to refine searches to the most relevant sources.
Metadata Content Type	List of metadata content types found in the search, with the ability to select and deselect content types.
Query	<p>The Query builder enables users to construct a very granular search query.</p> <p>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, before, after, etc.)</p> <p>Click the + to add further constraints to the query, or x to remove.</p> <p>Click Search to use the new query.</p>
Search Button	Executes enhanced search on new parameters specified or query built above.



Record Summary

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

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

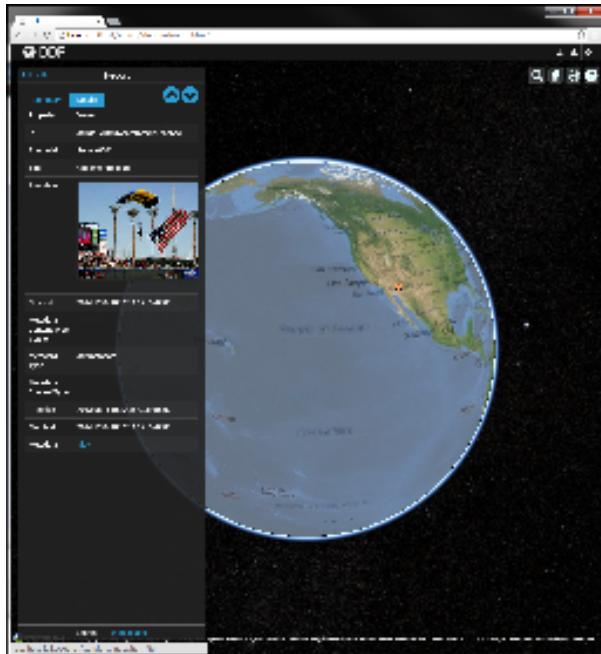


Record Details

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.
Metadata content type version	The version of the metadata associated with the record.
Metocard Type	The type of metocard associated with the record.

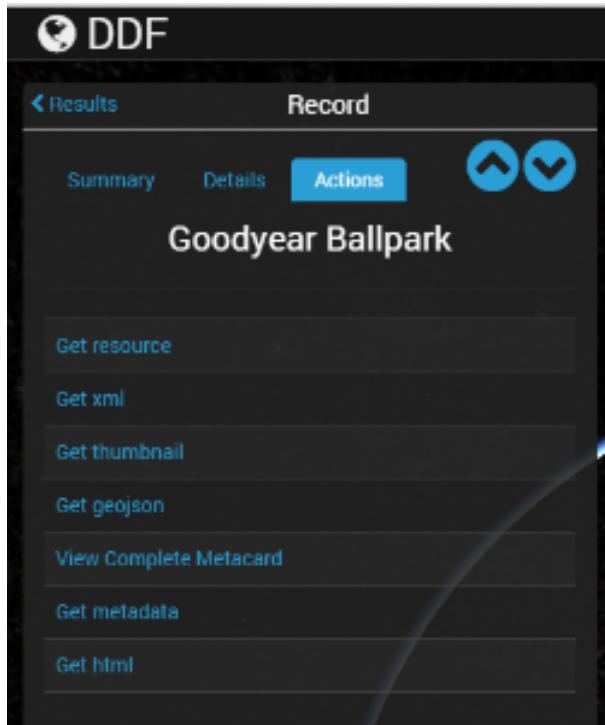
Metadata content type	The type of the metadata associated with the record.
Resource size	The size of the resource, if available.
Modified	How long ago the record was modified.
Download	When applicable, a download link for the product associated with the record is displayed. The size of the product is also displayed, if available. If the size is not available, Unknown Size is displayed.
Metadata	Shows a representation of the metadata XML, if available.



Record 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](#).



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.

DDF

Search Results >

Keywords

Time
 Created Modified Effective
 Any Relative Absolute

Location
 Lat / Lon USNG / MGRS
 Any Polygon Point-Radius Bounding Box

Type
 Any Specific Types

Sorting
 Modified Created Effective Relevance Distance
 Descending Ascending

Additional Sources
 All Sources Specific Sources None

Search Save Clear

Search Workspaces

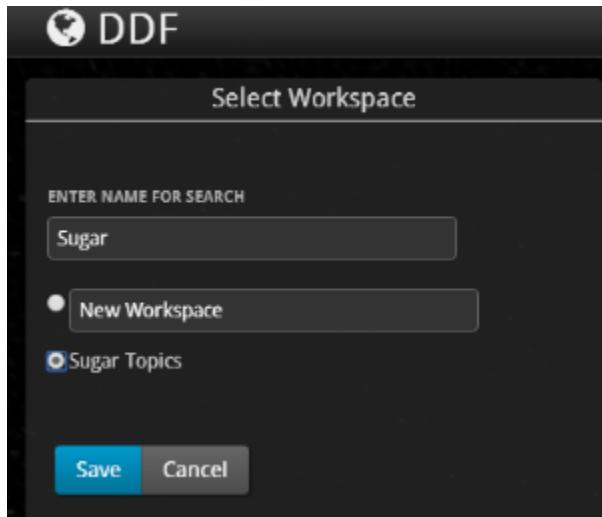
Example

Query parameters are highly configurable. An example scenario could be monitoring activity in a certain country, such as Luxembourg. Location-based search could be as simple as a bounding box around the desired area. The polygon search allows fine-tuning along geographic features or national borders. A second, contextual query can field results mentioning the subject country originating outside the borders.

Grouping similar searches into workspaces simplifies workflow and automates those queries that need to run repeatedly or frequently.



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.

Download Link and Product Size

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 **Unknown Size**, indicating that the size is unknown or not applicable.

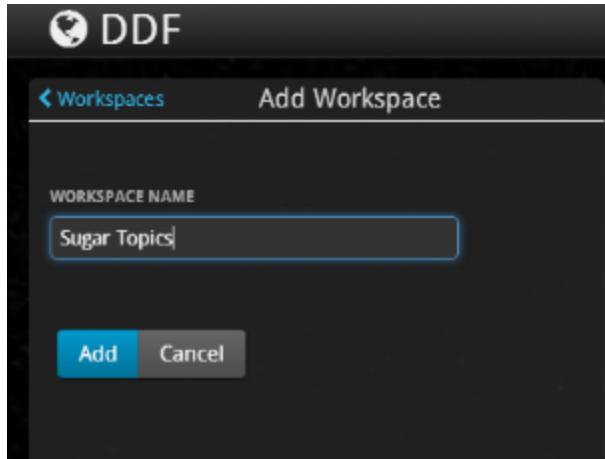
However, if the administrator has enabled caching on DDF and has installed the `catalog-core-resourcesizeplugin` PostQuery Plugin, and if the product has been retrieved, it has been cached and the size of the product can be determined based on the cached file's size. Therefore, subsequent query results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

Workspaces

Each user can create multiple workspaces and assign each of them a descriptive name. Each workspace can contain multiple saved searches and have 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 DDF is restarted. Currently, workspaces are private and cannot be viewed by other users.

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 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**.

Query Name

Keywords

Time

Created Modified Effective

Any Relative Absolute

Location

Lat / Lon USNG / MGRS

Any Polygon Point-Radius Bounding Box

Type

Any Specific Types

Sorting

Modified Created Effective Relevance Distance

Descending Ascending

Additional Sources

All Sources Specific Sources None

1 of 1 selected

Scheduling

Not Scheduled Scheduled

Action Buttons

Save & Search Clear Cancel

Navigation

Search Workspaces

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.

Workspaces **Sugar Topics** **Edit**

SEARCHES **+**

Search Name	Created	Count
Sugar	a few seconds ago	1

View Saved Records (0) >

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.

Notifications and Activities

Notifications

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



).

Currently, the notifications provide information about product retrieval only. After a user initiates a resource download, they receive notifications when the download completed, failed, canceled, or is being retried.

Refer to the the [Configuring Notifications](#) and the [Asynchronous Notifications and Activities](#) sections of the DDF documentation for more detailed information.

A notification popup will remain visible until it is dismissed or the browser is refreshed. Once a notification is dismissed, it cannot be retrieved again.

Activities

Similar to notifications, activities appear as pop-up windows inside the Search UI. Activities are only available when the administrator has enabled "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. 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. Refer to the example of the Activity menu below.

If caching is enabled, a progress bar is displayed in the Activity (Product Retrieval) drop-down menu until the action being performed is complete. Refer to the [Asynchronous Notifications and Activities](#) section of the DDF documentation for more information.

Downloads

Downloads from the UI are currently managed by the user-specific browser's download manager, and will also be displayed as notifications when canceled or completed.

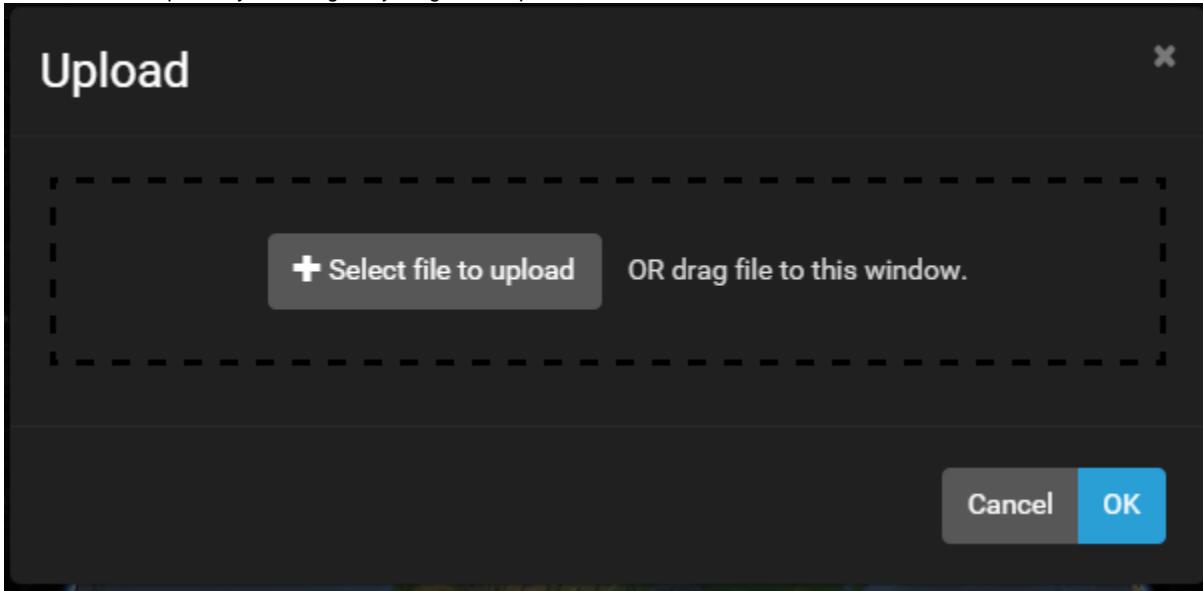
Uploading New Resources

In order to ingest new resources to the local DDF repository, the user can select the upload icon (

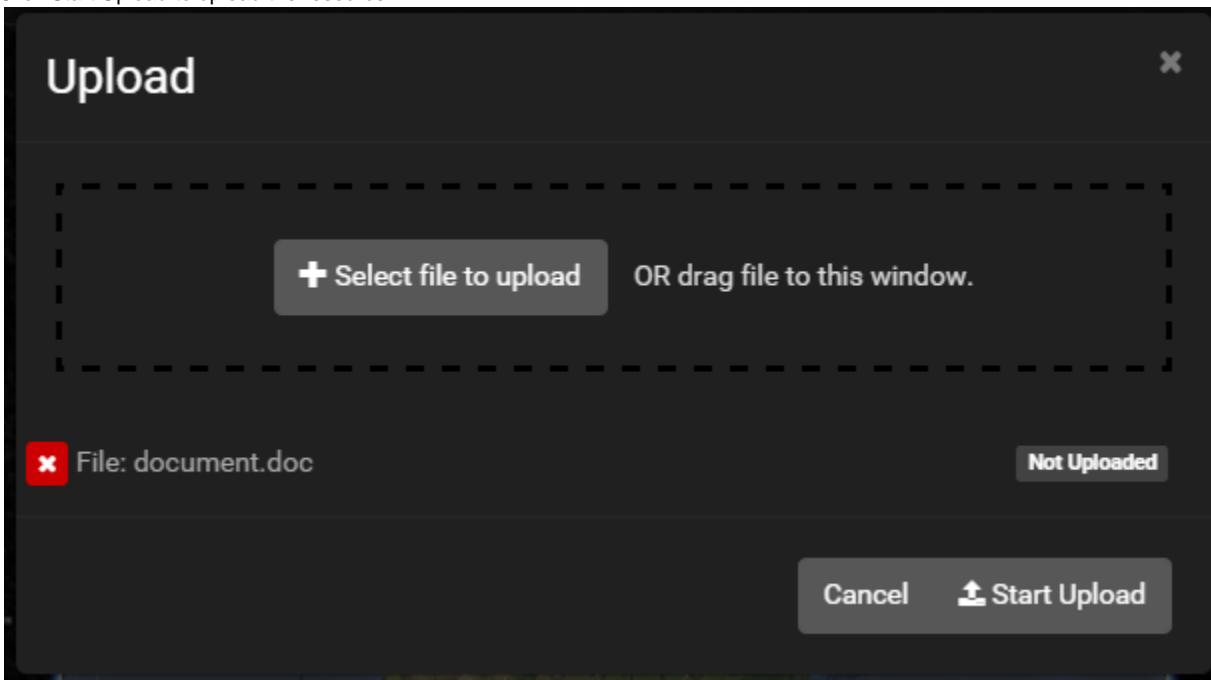


) in the top right corner of the window to launch the Upload dialog. This dialog allows you to upload a resource and input metocard attributes about this resource.

1. Select the file to upload by browsing or by drag and drop.



2. Click Start Upload to upload the resource.



3. Update metocard attributes and Save the metocard.

Upload



+ Select file to upload

OR drag file to this window.

File: document.doc

Success!

Title

My Document

Effective Time

2015-01-27T11:30:39.706-0500

Content Type

application/msword

Thumbnail

Upload new thumbnail



Save Changes

No Change

Delete Metocard

Cancel

OK

4. Select OK. The resource will be stored and the metocard will be generated. The contents of the resource will be used to generate the metadata if the format of the resource is recognized.

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



.)

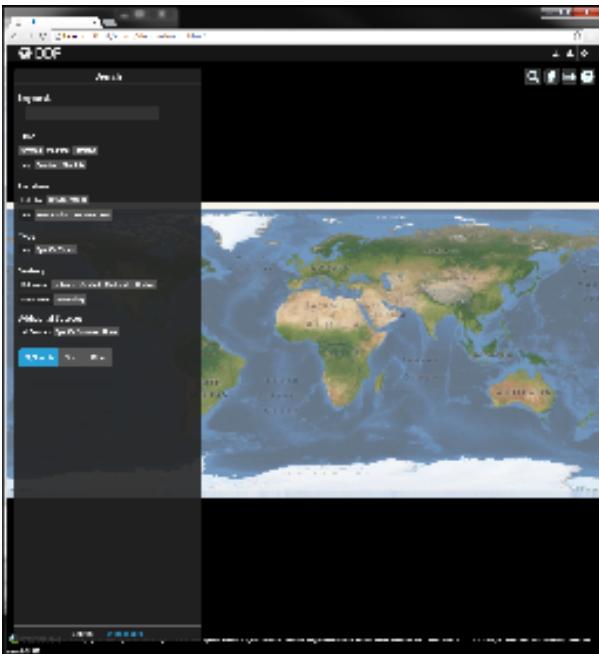
3D View

The 3D view is a fully interactive globe view.



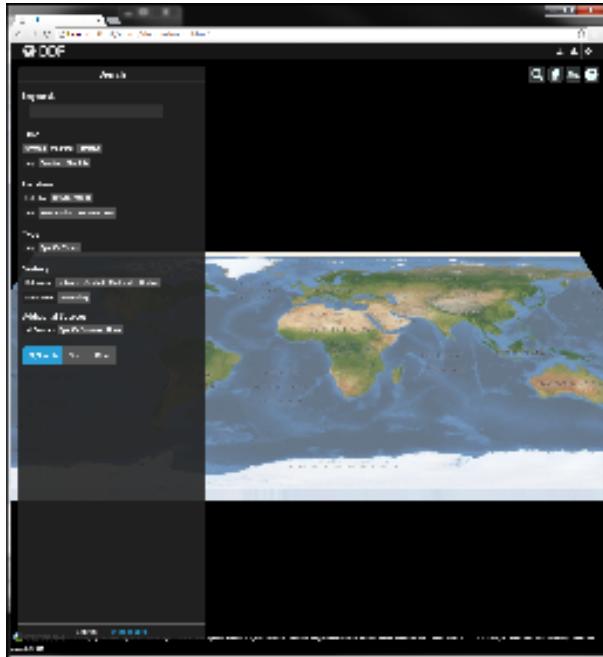
2D View

The 2D view is a flattened map view. It is preferable in environments that do not support the more resource-intensive globe view.



Columbus View

This View is another alternative format for viewing search results' locations.



Known Issues

- Stale search results, i.e., records that have been deleted, may be displayed in query search results sometimes. This occurs when the query cache has not been kept in sync with the catalog provider as records were deleted from the provider. If this occurs, the administrator should be notified to clean the query cache following the instructions for [Troubleshooting DDF Standard Search UI](#).

Managing DDF Standard Search UI

Overview

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

Table of Contents

- [Overview](#)
- [Installing DDF Standard Search UI](#)
- [Troubleshooting DDF Standard Search UI](#)

Installing DDF Standard Search UI

Overview

This page describes:

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

- [Overview](#)
- [Prerequisites](#)
- [Installing](#)
 - [Verifying Installation](#)
- [Configuring](#)
 - [Configurable Properties](#)
- [Uninstalling](#)
 - [Uninstalling manually](#)
 - [Reverting the Uninstall](#)
- [Upgrading](#)
 - [Upgrading manually](#)

Prerequisites

Before the DDF Search UI application can be installed:

- the [DDF Kernel](#) must be running.
- the [DDF Platform Application](#) must be installed.
- the [DDF Catalog Application](#) must be installed.

Installing

The Search UI application is installed by default

If using the Admin application, this app can be installed [via the Admin Console](#) or the [System Console](#) (at <http://localhost:8181/system/console/features>). Otherwise, follow steps below.

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.

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

Verifying Installation

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.

Configuring

Configure individual features within the application with [Admin Console](#).

Configurable Properties

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: OSM (OpenStreetMap), AGM (ArcGisMap), BM (BingMap), WMS (WebMapService), WMT (WebMapTile), TMS (TileMapService), GE (GoogleEarth).</p> <p>Example: <code>{"type" "WMS" "url" "http://example.com" "layers" ["layer1" "layer2"] "parameters" {"FORMAT" "image/png" "VERSION" "1.1.1"} "alpha" 0.5}</code></p>	yes
Terrain Providers	terrainProvider	String	<p>Terrain provider to use for height data. Valid types are: CT (CesiumTerrain), AGS (ArcGisImageServer), VRW (VRTheWorld).</p> <p>Example: <code>{"type" "CT" "url" "http://example.com"}</code></p>	no
Map Projection	projection	String	Projection of imagery providers	no

Connection timeout	timeout	Integer	The WMS connection timeout.	yes
Show sign in	signIn	Boolean	Whether or not to authenticate users.	no
Show tasks	task	Boolean	Whether or not to display progress of background tasks.	no
Show Gazetteer	gazetteer	Boolean	Whether or not to show gazetteer for searching place names.	no
Show Uploader	ingest	Boolean	Whether or not to show upload menu for adding new metadata.	no
Type Name Mapping	typeNameMapping	String[]	The mapping of content types to displayed names.	no

Uninstalling

If using the Admin application, applications can be removed [via the Admin Console](#).

Uninstalling manually

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

```
features:removeurl -u <DDF application's feature repository URL>
```

Example: `features:removeurl -u mvn:ddf.ui.search/search-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 `features:list` command output.

The repository URLs for installed applications can be obtained by entering:

```
features:listrepositories -u
```

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

```
features:addurl <DDF application's feature repository URL>
features:install <DDF application's main feature>
```

Example:

```
ddf@local>features:addurl
mvn:ddf.catalog/catalog-app/2.3.0/xml/features
ddf@local>features:install catalog-app
```

Upgrading

Upgrading to a newer version of the app can be performed by the [Admin Console](#).

Upgrading manually

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.

```
features:install admin-app
```

4. Complete the steps in the Verify section above to determine if the upgrade was successful.

Troubleshooting DDF Standard Search UI

Overview

This page describes known issues that may be encountered with the Standard Search UI and how to address them.

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 still being returned from the cache. When this occurs records in the cache can be manually deleted using the URL commands listed below from a browser. In these command URLs, metocard_cache is the name of the Solr query cache. These command URL examples are for a non-secured install. If the DDF is secured, then the command URLs would start with <https://localhost:8993/> assuming the secure port is configured for 8993.

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

Deletion of all records in Solr query cache

```
http://localhost:8181/solr/metocard_cache/update?stream.body=<delete><quer
y>*:*</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><quer
y>original_id_txt: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><quer
y>title_txt:*remote*</query></delete>&commit=true
```

Integrating DDF Standard Search UI

Overview

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

This guide supports integration of this application with external frameworks.

Extending DDF Standard Search UI

Overview

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

This guide supports developers creating extensions of the existing framework.

Securing DDF Standard Search UI

Overview

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

This guide covers implementations and protocols for enhancing security.

DDF Standard Search UI Release Notes

- Release Version: ui-2.6.1

Release Version: ui-2.6.1

Contents

- New Capabilities
- Resolved Issues
- Known Issues
- App Prerequisites
- Third Party Licenses

New Capabilities

- Enhanced Search

- New Sorting options (e.g. Modified)
- Metocard Action support
- Upload Resource
- Ability to customize types

Resolved Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.										

Known Issues

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	Resolution
 Unable to locate JIRA server for this macro. It may be due to Application Link configuration.										

App Prerequisites

Before the DDF UI Application can be installed:

- the DDF Kernel must be running
- the DDF Platform App 2.6.1 must be running
- the DDF Catalog App 2.6.1 must be running

Third Party Licenses

[Third Party License File Notice](#)

DDF Release Versions

- [DDF 2.6.0 Release Versions](#)

DDF 2.6.0 Release Versions

On This Page	
<ul style="list-style-type: none"> • Application Updates • Repository Updates <ul style="list-style-type: none"> • DDF Libs • Codice Thirdparty • Applications Included in Distribution 2.6.0 	

Application Updates

Repository	Application	Version	Description

ddf-admin	admin-app	1.1.1	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-catalog	catalog-app	2.6.1	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-content	content-app	2.4.2	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-parent		3.0.0	Parent cleanup																				
ddf-platform	platform-app	2.6.1	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-security	security-services-app	2.5.1	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-solr	solr-app	2.4.3	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-spatial	spatial-app	2.6.1	<table border="1"> <thead> <tr> <th>Key</th><th>Summary</th><th>Type</th><th>Created</th><th>Updated</th><th>Due</th><th>Assignee</th><th>Reporter</th><th>Priority</th><th>Status</th></tr> </thead> <tbody> <tr> <td colspan="10">⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.</td></tr> </tbody> </table>	Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status	⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status														
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.																							
ddf-stomp		1.0.0																					

Key	Summary	Type	Created	Updated	Due	Assignee	Reporter	Priority	Status
⚠️ Unable to locate JIRA server for this macro. It may be due to Application Link configuration.									
distribution		2.6.0							

Repository Updates

The following repositories contain libraries that are individually versioned and released.

DDF Libs

Library	Version	Description	Comments	Status	Assigned to
notifications	1.1.0			RELEASED	
httpproxy	1.0.1			RELEASED	
activities	0.3.0			NOT CHANGED	

Codice Thirdparty

Library	Version	Description	Status

Applications Included in Distribution 2.6.0

Name	Version
platform-app	2.6.1
security-services-app	2.5.1
catalog-app	2.6.1
content-app	2.4.2
solr-app	2.4.3
opendj-embedded	1.1.0
spatial-app	2.6.1
search-ui-app	2.6.1
admin-app	1.1.1

How-to articles

Add how-to article

How-to article

Provide step-by-step guidance for completing a task.

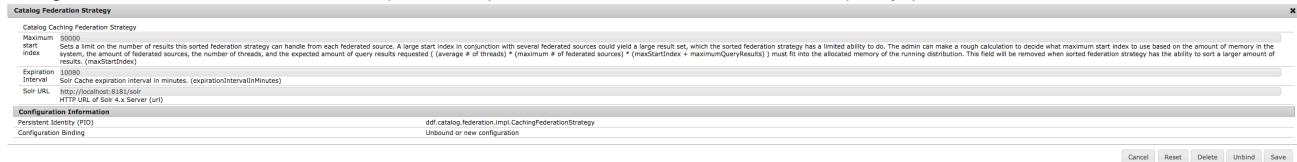
Add how-to article

How to Change the Solr Cache Expiration Interval

These steps describe the process of changing the Solr Cache expiration interval.

Step-by-step guide

1. Go to the admin console: <http://localhost:8181/system/console>
2. Select OSGI -> Configuration from the top menu.
3. Navigate to the Catalog Federation Strategy configuration item.
4. Configure the Expiration Interval (in minutes). Note: the default is 10080 minutes (7 days).



Replace "localhost" with System Certificates

DDF ships with default certificates configured to identify the local machine as "localhost." This allows one to unzip DDF and run immediately in a secure manner. However, in order to access the DDF instance from another machine over HTTPS (now the default for many services) without receiving warnings or failures, the default certificates need to be replaced with a certificate reflecting the name of the host machine as reachable from other network nodes. This article the process involved in creating new certificates, installing them into the DDF keystores, and configuring various components of DDF to utilize the new certificates. The general steps described in detail below consist of:

- Creating new certificates for each DDF host machine/VM
- Changing URLs in various configurations to reference the host by name
- Updating the certificates used by DDF

Step-by-step guide

Related articles

Content by label

There is no content with the specified labels



Index

Space Index

0-9 ... 0	A ... 19	B ... 3	C ... 39	D ... 61	E ... 25
F ... 3	G ... 0	H ... 5	I ... 25	J ... 1	K ... 0
L ... 1	M ... 14	N ... 0	O ... 4	P ... 3	Q ... 1
R ... 22	S ... 47	T ... 3	U ... 9	V ... 0	W ... 2
X ... 2	Y ... 0	Z ... 0	!@#\$... 0		

0-9

A

Admin If you use Adminis Qui Anonym Ove sec Appenc Ove Applica Apri ope Applica Thi hov Applica Pre \$ap Applica If th pre Applica Rei con Applica To \$ap Sta Applica To AC Applica DD Applica ddf Applica Ove Assurin Intr ver Asynch Not Cor

Config
Ov
mir
Config
Ov
App
Config
Ov
Config
Ov
Config
Ov
Config
Ov
Config
Ov
cor
Fee
Config
Ov
Acc
Cor
Config
Ov
Client
Config
Cor
files
Config
Ov
Cor
adr
Consolid
Ov
to r
Content
Content
dor
/-=-
Content
con
cre
Content
the
+---
Content
Arc
| ^ |
Content
Cre
con
Me
Content
rep
+---
Content
pro
cat:
Creatin
Cre
for
CSW E
CSW S
Current

DDF Data Migration	Overview 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 pr	Extendi Ov cre Extendi Ov inst Extendi Ov que Extendi Ov woi Extendi Ov acti
DDF Developer's Guide	Describes methods to develop new capabilities for DDF, both for the Catalog and at the Framework level.	
	Overview This guide discusses the several extension points and components permitted by the Distributed Data Framework (ApplicationName) Catalog API. U	
DDF Development Prerequisites	Overview Development requires full knowledge of the DDF Catalog. ApplicationName 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 inter	
DDF Directory Contents after Installation	Major Directories During DDF installation, the major directories shown in the table below are created, modified, or replaced in the destination directory. Directory Name Description bin Scripts to start and stop DDF data The working directory of the syste	
DDF Home	Welcome to the home of the DDF. The Quick Start describes how to get up and running quickly. DDF Introduction Please visit the DDF Administrator's Guide and the DDF Developer's Guide for more information. Building Building DDF How to Run * Unzip the distr	
DDF HTTP Proxy	Overview The DDF HTTP Proxy provides a reverse proxy mechanism to allow a DDF service to connect with external HTTP-based services and web pages via an internal URL. The proxy creates an endpoint on DDF, which is accessible via HTTP or HTTPS. This endo	
DDF Included Features	DDF Features A feature provides certain, modular functionality that can be easily installed or removed from the DDF. A brief description of that functionality for each feature is included below. Each feature is comprised of a collection of bundles that	
DDF Integrator's Guide	Overview This guide provides instructions for configuring, maintaining and operation components of the Distributed Data Framework. DDF Introduction Contents	
DDF Introduction	Overview Distributed Data Framework (DDF) is an agile and modular integration framework. It is primarily focused on data integration, enabling clients to insert, query and transform information from disparate data sources via the DDF Catalog. A Catalog A	
DDF Is Unresponsive to Incoming Requests	ApplicationName Is Unresponsive to Incoming Requests Problem: ApplicationName is unresponsive to incoming requests. An example of the log file when this problem is encountered: Feb 7, 2013 10:51:33 AM org.apache.karaf.main.SimpleFileLock lock INFO: lockin	
DDF Load Balancer	provides utility that allows incoming traffic to be distributed over multiple instances of DDF, via HTTP or HTTPS Overview Contained within DDF is a Load Balancer utility that allows incoming traffic to be distributed over multiple instances of DDF. The D	
DDF Mime Framework	Mime Type Mapper The MimeMapper is the entry point in ApplicationName for resolving file extensions to mime types, and vice versa. MimeMappers are used by the ResourceReader to determine the file extension for a given mime type in aid of retrievin	
DDF Platform Application	Overview 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. These building blocks include subsets of: Karaf (http://karaf.apache.org)	
DDF Platform Application Release Notes	Overview This page lists Release notes for various versions of ApplicationName.	
DDF Registry Application	Overview The registry application offers APIs and services that allow DDF nodes to dynamically discover other nodes and automatically enable federation between them. The DDF Registry code is currently a very functional, but initial prototype that was cr	
DDF Registry Application Release Notes	Overview DDF Registry Application There are currently no released versions of the DDF Registry application.	
DDF Release Versions		
DDF Roadmap	DDF Roadmap.png	

DDF Security Application

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

DDF Security Application Install and Uninstall

Overview Prerequisites Before the DDF Security application can be installed: the DDF Kernel must be running and the DDF Platform Application must be installed. Install Before installing a DDF application, verify that its prerequisites have been met. Copy the

DDF Security Application Release Notes

Overview DDF Security Application _security_guide_intro

DDF Solr Catalog Application

Overview The Solr Catalog Provider (SCP) is an implementation of the CatalogProvider interface using Apache Solr (<http://lucene.apache.org/solr/> <http://lucene.apache.org/solr/>) as a data store. Some notable features of the SCP include: Supports extensible

DDF Solr Catalog Application Release Notes

Overview This page contains the release notes for recent version of ApplicationName Solr catalog application.

DDF Spatial Application

Overview The ApplicationName 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 Spatial Application Release Notes

DDF Standard Search UI

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

DDF Standard Search UI Release Notes

DDF Static Analysis and Build

Github Badges Github/Markdown support the use of badges to indicate status on a project. This page will be used to track potential badges to add to DDF and their status. Name Description Status Travis CI Continuous Integration Builds Added to all Repositories

DDF STOMP

Overview The DDF STOMP application allows query subscription messages to be sent to the DDF server via STOMP protocol. Subscription query messages are defined in JSON format following a defined schema.

These messages allow for the management of subscriptions

DDF User's Guide

Overview This guide provides instructions to install, start, and stop the DDF . DDF Introduction Contents

DDF-libs

DDF-libs is a repository for library modules in DDF. Typically the modules in this repository are re-usable across different components of DDF.

Decision Making

Overview DDF, like the Codice Foundation, heavily leverages processes from Apache Software Foundation. Making decisions by voting is no exception. Therefore, the ASF voting process <http://www.apache.org/foundation/voting.html> serves as a primary reference

Deploying a Bundle

Deploying a Bundle A bundle is typically installed in two ways: As a feature or 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 ApplicationName. This directory

Deployment Guidelines

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

Developer Prerequisites

Prerequisites Understand the desired component for development as described in the DDF Catalog section. Review existing implementations in the source code and application documentation. Have an IDE and the ability to create OSGi bundles. Understand the Us

Developer Use of OGC Filter

answers frequently asked questions about the use of the OGC Filter in DDF Catalog Queries and Subscriptions Related Topic: OGC Filter with DDF Frequently Asked Questions in the User's Guide. Frequently Asked Questions How do I create a filter? First, get

Developing Action Components (Action Framework)

describes how and why to create Action Components Overview 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

Developing Catalog Components

Overview This section describes how to create Catalog components. Use in conjunction with the Javadoc to begin extending the DDF Catalog. Describes how to create Catalog components. Used in conjunction with the Javadoc to begin extending the DDF Catalog.

Developing DDF Applications

Overview 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 Web Console or command line console. DDF applications also consist of one o

Developing Token Validators

Description 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 o

Development Recommendations

provides general developer tips and recommendations for OSGi bundle development Javascript Avoid using Console.log Package Names Use singular package names. Author Tags Author tags are discouraged from being placed in the source code, as they can be a bar

Directory Permissions

DDF is installed in the DDF_HOME directory. Directory Permissions on Windows Restrict access to sensitive files by ensuring that the only users with access privileges are administrators. Right-click on the file or directory noted below then select Full

Distribution Will Not Start

Distribution Will Not Start Problem: ApplicationName will not start when calling the start script defined during installation. Solution: Complete the following procedure. 1. Verify that Java is correctly installed. java -version 2. Should return something

Documentation Downloads

Overview This page contains complete documentation for each of the ApplicationName application in a downloadable, printable pdf format. Get Adobe Reader <http://get.adobe.com/reader/>. Downloads Admin.pdf Catalog.pdf Content.pdf Platform.pdf Registry.pdf Se

Documentation Guide

This page establishes conventions used throughout the documentation. Overview This document serves to guide users, administrators and developers through the ApplicationName. Documentation Updates The most current Distributed Data Framework (DDF) document

F

Fat Bundles

Decision Status [NOT VOTED, APPROVED, DENIED, TABLED] Approved Date of Decision Nov 13, 2013 Contributions by Ashraf Barakat (editor) Bruce Beyeler Phillip Klinefelter Michael Menousek Description Decision Decision is to adopt the best practice of creatin

Federation UI

Overview The federation user interface is a convenient way to manage federated data sources for the DDF. Federation enables including remote sources, including other ApplicationName installations in queries. For a full description of Federation, see Exten

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 (<http://servicemix>).

G

H

Hardening

Overview These instructions demonstrate how to harden a ApplicationName system for a more secure installation. The web administration console is not compatible with Internet Explorer 7. Disable the Web Console To harden ApplicationName for security purp

How to become active in the DDF Community

Get involved and Contribute <http://www.codice.org/contributing> via forums <https://groups.google.com/forum/?fromgroups#!forum/ddf-developers>, edit the documentation, work on the issue tracker <http://tools.codice.org/jira/browse/DDF>, and submit pull requ

How to Change the Solr Cache Expiration Interval

These steps describe the process of changing the Solr Cache expiration interval. Step-by-step guide Go to the admin console: <http://localhost:8181/system/console> <http://localhost:8181/system/console> Select OSGi -> Configuration from the top menu.

How-to articles

I

Include

Index

index.h

Ple

</s

Ingestir

Ove

file

Installir

Thi

ger

Installir

L	<p>LDAP Administration</p> <p>Overview OpenDJ provides a number of tools for LDAP administration. Refer to the OpenDJ Admin Guide (http://opendj.forgerock.org/opendj-server/doc/admin-guide/). Download the Admin Tools OpenDJ (V</p>	M	<p>Management Overview</p> <p>Descriptor Definition</p> <p>Management Description</p> <p>Management Overview</p> <p>Management Overview</p> <p>Management Overview</p> <p>Management Overview</p> <p>Management Overview</p> <p>Management Overview</p> <p>Metrics Collection</p> <p>Metrics Collection</p> <p>Metrics Allocation</p> <p>Metrics Application</p> <p>Module Overview</p> <p>Module Import</p> <p>MTS Configuration</p>
N		O	<p>OGC Functionality</p> <p>Thing</p> <p>Free</p> <p>Queue</p> <p>OpenShift Container</p> <p>shc</p> <p>OpenShift</p> <p>OpenW</p> <p>Open</p> <p>Pass</p> <p>OSGi Service</p> <p>pro</p> <p>reg</p>
P	<p>Persistence Commands</p> <p>Persistence Commands Title Namespace Description ApplicationName:: Persistence :: Core :: Commands store The Persistence Shell Commands are meant to be used with any PersistentStore implementations.</p>	Q	<p>Quick Start</p> <p>Overview</p> <p>enc</p>

They provide the ability to query and delete entries from the database.

Platform Commands

Platform Commands Title Namespace Description ApplicationName Platform Commands platform The ApplicationName Platform Shell Commands provide generic platform management functions The Platform Commands are installed when the Platform application is installed.

Platform Global Settings

The system-wide configuration settings used throughout DDF Overview The Platform Global Settings are the system-wide configuration settings used throughout DDF to specify the information about the machine hosting DDF. Configuration Configuration can be performed.

R

Redaction and Filtering

Overview Redaction and filtering are performed in a Post Query plugin that occurs after a query has been performed. How it Works Each metocard result will contain security attributes that are pulled from the metadata record after being processed by a Post.

Release Guide

How to create and announce a DDF Release Required tools Use Maven 3.0.3 Use git client 1.8 or greater Sample Versioning SNAPSHOT RELEASE 1.0.0-SNAPSHOT 1.0.0 1.1.0-SNAPSHOT 1.1.0 Creating the Release //change directory to repo cd ddf-repo //ensure you a

Release Version: catalog-2.5.0

New Capabilities Resource Caching Reliable Resource Retrieval Notifications Activities Ability to Cancel Resource Retrieval is associated with a user Resolved Issues Known Issues App Prerequisites Before the DDF Catalog Application 2.5.0 can be installed.

Release Version: catalog-2.5.1

Contents Overview This release contains bug fixes that were discovered in ddf-catalog 2.5.0 Bug Fixes Known Issues App Prerequisites Before the DDF Catalog Application 2.5.1 can be installed: the DDF Kernel must be running the DDF Platform App 2.5.1.

Release Version: catalog-2.6.1

New Capabilities Sources UI Catalog Backup Plugin Resolved Issues Known Issues App Prerequisites Before the DDF Catalog Application 2.6.1 can be installed: the DDF Kernel 2.6.1 must be running the DDF Platform App 2.6.1 must be running Third Party Lic.

Release Version: content-2.4.0

Contents App Dependencies

Release Version: content-2.4.2

Contents Resolved Issues Known Issues

Release Version: ddf-admin-1.0.1

Contents App Dependencies commons-collections-3.2.1.jar jolokia-osgi-1.2.1.jar json-smart-1.1.1.jar ops4j-base-util-property-1.4.0.jar org.apache.aries.jmx.core-1.1.1.jar org.apache.karaf.bundle.core-3.0.0.1.jar org.apache.karaf.bundle.springstate-3.0

Release Version: ddf-admin-1.1.1

Contents Summary The Administration UI was updated to include a configuration module for managing applications, features, and configurations. Resolved Issues Known Issues

Release Version: ddf-catalog-2.4.1

Contents App Dependencies abdera-client-1.1.3.jar abdera-core-1.1.3.jar abdera-extensions-geo-1.1.3.jar abdera-extensions-opensearch-1.1.3.jar abdera-i18n-1.1.3.jar abdera-server-1.1.3.jar axiom-api-1.2.10.jar commons-codec-1.4.jar commons-collections

Release Version: ddf-platform-2.4.0

Contents App Dependencies abdera-core-1.1.3.jar abdera-extensions-main-1.1.3.jar abdera-i18n-1.1.3.jar abdera-parser-1.1.3.jar apache-mime4j-core-0.7.2.jar c3p0-0.9.1.1.jar cal10n-api-0.7.4.jar camel-blueprint-2.12.1.jar camel-core-2.12.1.jar camel-fr

Release Version: platform-2.3.1

Contents New Capabilities DDF Platform Application Initial Release Added support for accessing metrics tab in admin console via SSL Resolved Issues Known Issues Included Features Feature Name Bundles Installed Description Installed by Default ac

Release Version: platform-2.5.0

Contents New Capabilities Added a persistent store Solr Server (migrated from the Solr application) Resolved Issues Known Issues App Prerequisites Before the DDF Platform Application can be installed: the DDF Kernel must be running App Dependenc

Release Version: platform-2.5.1

Contents Overview This release is a bug fix release of platform 2.5.0 Resolved Issues Known Issues App Prerequisites Before the DDF Platform Application can be installed: the DDF Kernel must be running App Dependencies com.codahale.metrics:met

S

Search

Overview

that

Security

Overview

Release Version: platform-2.6.1

Contents Overview This release is a bug fix release of platform 2.6.0 Added EXI Compression Resolved Issues Known Issues App Prerequisites Before the DDF Platform Application can be installed: the DDF Kernel must be running

Release Version: security-2.5.1

Contents Overview This is a bug fix release to address issues discovered in security 2.5.0 Anonymous Interceptor added Resolved Issues Known Issues App Prerequisites Before the DDF Security Application can be installed: the DDF Kernel must be running

Release Version: solr-2.4.2

Contents New Capabilities Additional Temporal Filter Support TEqual After PropertyGreaterThan (on dates) PropertyLessThan (on dates) PropertyGreaterThanOrEqualTo (on dates) PropertyLessThanOrEqualTo (on dates) Solr Server moved to ddf-platform Resolved

Release Version: solr-2.4.4

Contents New Capabilities Secure Solr Admin by default Resolved Issues Known Issues App Prerequisites Before the DDF Solr Application 2.4.4 can be installed: the DDF Kernel must be running the DDF Platform App 2.6.1 must be running the DDF Catalog

Release Version: spatial-2.6.1

Contents New Capabilities CSW Metacard Transformer CSW Source and Endpoint support Metacard XML Resolved Issues Known Issues App Prerequisites Before the DDF Spatial Application 2.6.1 can be installed: the DDF Kernel must be running the DDF Platform

Release Version: ui-2.6.1

Contents New Capabilities Enhanced Search New Sorting options (e.g. Modified) Metacard Action support Upload Resource Ability to customize types Resolved Issues Known Issues App Prerequisites Before the DDF UI Application can be installed: the DDF Ke

Replace "localhost" with System Certificates

DDF ships with default certificates configured to identify the local machine as "localhost." This allows one to unzip DDF and run immediately in a secure manner. However, in order to access the DDF instance from another machine over HTTPS (now the default

Resource Description

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

Ov
con

Securi
Ov

with

Securi
Ov

use

Securi
Ov

ser

Securi
Ov

enc

Securi
Ov

anc

Securi
Ov

fea

Securi
Ov

cor

Securi
Ov

attr

Securi
Ov

doe

Securi
Ov

infc

retr

Securi
Ov

typ

Securi
Ov

use

Securi
Ov

lifec

Securi
Ov

(ST

Loc

Securi
Ov

This

Securi
Ov

fror

Securi
Ov

aut

Securi
Ov

diff

Securi
Ov

cor

Securi
Ov

		<p>pro (ST</p> <p>Software For fun</p> <p>Solr Ca imp imp fea</p> <p>Standar an . to r</p> <p>Standar The</p> <p>Starting Ov con</p> <p>StopPr Fai Plu</p> <p>Storage pro /-=-</p> <p>Subscri Sut App sub</p>
T		<p>Team Project Management Committee Bruce Beyeler, Lockheed Martin Corp. Phil Klinefelter, Connexta LLC Michael Menousek, Connexta LLC (Chair) Shaun Morris, Lockheed Martin Corp. Matt Ramey, Cohesive Integrations LLC Scott Tustison, Connexta LLC Keith Wire, Con</p> <p>Troubleshooting Overview This section is a compilation of successful troubleshooting steps to take while installing or maintaining ApplicationName.</p> <p>Troubleshooting DDF Standard Search UI Overview This page describes known issues that may be encountered with the Standard Search UI and how to address them. Deleted Records Are Being Displayed In The Standard Search UI's Search Results When queries are issued by the Standard Search UI, the qu</p>
V		<p>W</p> <p>Whitebo All reg</p> <p>Wiki Pa</p>

X	<p>X.509 PublicKey SAML Security Token Request/Response</p> <p>Overview In order to obtain a SAML assertion to use in secure communication to ApplicationName, 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 t</p> <p>XACML Policy Decision Point (PDP)</p> <p>Overview After unzipping the DDF distribution, place the desired XACML policy in the <distribution root>/etc/pdp/policies directory. This is the directory in which the PDP will look for XACML policies every 60 seconds. A sample XACML policy is located at</p>	Y
Z		!@#\$

Community

- Team
- Community Contributions
- How to become active in the DDF Community
- Decision Making
- Release Guide
- DDF Roadmap
- DDF Static Analysis and Build

Team

Project Management Committee

- Bruce Beyeler, Lockheed Martin Corp.
- Phil Klinefelter, Connexta LLC
- Michael Menousek, Connexta LLC (Chair)
- Shaun Morris, Lockheed Martin Corp.
- Matt Ramey, Cohesive Integrations LLC
- Scott Tustison, Connexta LLC
- Keith Wire, Connexta LLC

Committers

- Ashraf Barakat, Self
- Ian Barnett, Self
- Daniel Figliola, Connexta LLC
- Damon Jones, Lockheed Martin Corp.
- Jesse Kim, Lockheed Martin Corp.
- Jay McNallie, Lockheed Martin Corp.
- William Miller, Lockheed Martin Corp.
- Hugh Rodgers, Lockheed Martin Corp.
- Jason Smith, Connexta LLC
- Khoa Tran, Lockheed Martin Corp.
- Dave Willison, Aviture

Contributors

- Kyle Dyer, Connexta LLC
- Jeremy Glasser, Aviture
- Brad Hards, Sigma Bravo
- Jeren Hicks, Aviture
- Brendan Hoffman, Connexta LLC
- Rick Larsen, Connexta LLC
- Mark Lucas, Aviture
- Mike Macaulay, Aviture
- Michael Nguyen, Lockheed Martin Corp.

- Megan Plachecki, Lockheed Martin Corp.
- Sam Snyder, Aviture
- Harrison Tarr, Lockheed Martin Corp.
- Lindsay Tustison, Connexta LLC
- Michael Verret, Connexta LLC
- Jeff Vetraino, Cohesive Integrations LLC

Community Contributions

On This Page

- Procedure for Contributing

Procedure for Contributing

If contributing as a community member, complete the following steps.

1. If the ticket does not exist, create a JIRA ticket on <https://tools.codice.org/jira/> to track changes being made.
2. Fork the DDF code on GitHub.
3. Clone the forked DDF.
4. Create a feature branch in the forked DDF.
5. Make changes.
6. Add the remote repository to clone:

```
git remote add codice-ddf https://github.com/codice/ddf.git
```

7. Fetch the code from the master Codice/DDF repository:

```
git fetch codice-ddf
```

8. Rebase the changes onto the latest commit from Codice/DDF's master branch:

```
git rebase codice-ddf/master
```

9. Create a pull request in GitHub to pull changes from the fork into Codice/DDF's master branch.
10. Ensure Travis CI indicates that the merge will have no issues.

Additional information can be found here:

- <https://www.openshift.com/wiki/github-workflow-for-submitting-pull-requests>.
- <http://stackoverflow.com/a/14681796>
- <https://gun.io/blog/how-to-github-fork-branch-and-pull-request/>
- <http://codeinthehole.com/writing/pull-requests-and-other-good-practices-for-teams-using-github/>
- <https://help.github.com/articles/syncing-a-fork>

How to become active in the DDF Community

- Steps to Becoming a Contributor to Documentation and Source Code
- Getting Started on Development
- Getting Karma on JIRA and Confluence

Get involved and [Contribute via forums](#), edit the documentation, work on the [issue tracker](#), and submit pull requests on GitHub.

Once you're actively contributing, one of our [Team](#) members may invite you to be a committer (after a vote has been called). When that happens, if you accept, the following process begins.

Note that becoming a committer is not just about submitting some pull requests; it is also about helping with the development and user [Discussion Forums](#), helping with documentation, and working on the issue tracker.

Steps to Becoming a Contributor to Documentation and Source Code

1. Fill out and submit the [Codice Individual Contributor License Agreement](#).
2. Submit a pull request with desired changes (for source code only).

Getting Started on Development

See the [Developer's Guide](#) for developer tips and information about contributing your changes and [source code formatting](#).

Getting Karma on JIRA and Confluence

After being an active participant, you can mail the dev list and request JIRA/Confluence karma. We can then grant the necessary karma, so you can start working on JIRA issues and editing the wiki.

Decision Making

On This Page
<ul style="list-style-type: none">• Overview• Voting<ul style="list-style-type: none">• Voting Implications• Expressing Votes• Voting Time Period

Overview

DDF, like the Codice Foundation, heavily leverages processes from Apache Software Foundation. Making decisions by voting is no exception. Therefore, the [ASF voting process](#) serves as a primary reference.

Voting

A call to vote may be necessary for code modifications, releases, or procedural changes. A call to vote will be posted on the developers [mailing list](#).

All subscribing members may cast a vote. However, the only binding votes are those cast by the [Team](#).

This project follows a no veto process when a vote is called, meaning that a -1 will stop a decision until all vetoers withdraw their -1 votes.

Voting Implications

Voting requires a level of commitment. Voting members are not only stating their opinion, they are also agreeing to help do the work.

Expressing Votes

+1	Agree, yes, or this should be done
0	Abstain, no opinion.
-1	No. This will constitute as a veto. This must be accompanied by a technical reason or explanation otherwise this vote is void. The person who vetos must withdraw this vote for the decision to be approved. The person can be lobbied to change their vote.

Voting Time Period

Votes must be submitted within 72 hours of the call to vote.

Release Guide

How to create and announce a DDF Release

Required tools

- Use Maven 3.0.3
- Use git client 1.8 or greater

Sample Versioning

SNAPSHOT	RELEASE
1.0.0-SNAPSHOT	1.0.0
1.1.0-SNAPSHOT	1.1.0

Creating the Release

```

//change directory to repo
cd ddf-repo

//ensure you are on the master branch
git co master

//verify no files are checked out
git status

//verify the latest
git pull --rebase origin

//version according to versioning guidance
// if releasing master, next development version increments minor version
// if from a minor branch, next development version increments patch
version
mvn release:prepare

//verify new tag name
git tag

//push the new master and tag
git push origin master --tags

//Clean release files
mvn release:clean

//check out the new tag that was created
git co tag

//Deploy the artifacts into the Codice Nexus
mvn deploy

//reminder to not stay in the tag
git co master

```

Starting the vote

To: "DDF PMC" <ddf-pmc@googlegroups.com>

Subject: Vote on DDF X.Y.Z. Release

Hi - We resolved N number of issues this release

<https://tools.codice.org/jira/browse/DDF>

There are still outstanding issues:

<https://tools.codice.org/jira/secure/IssueNavigator.jspa?mode=hide&requestId=10602>

Please vote to approve this release

+1 = Approve

-1 = Veto the release (Requires specific reasons)

This vote will be open for 72 hours

Community Awareness Message

To: "DDF Developers Forum" <ddf-announcements@googlegroups.com>
Subject: DDF Repo XYZ release

Hi,

The DDF Team is pleased to announce the release and deployment of DDF application artifacts.

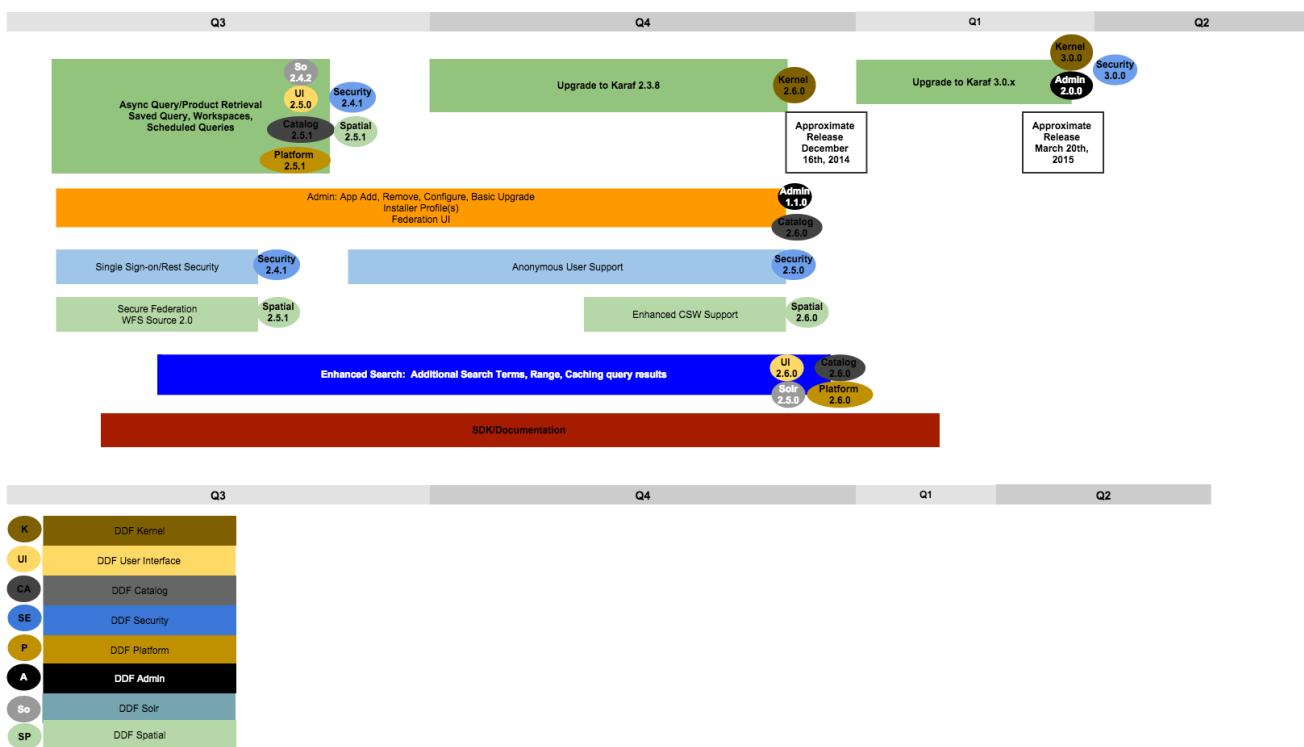
For an idea of what changed, see <https://tools.codice.org/wiki/display/DDF/2.3.0.ALPHA4>

The DDF zips are still available and can be found at
<http://artifacts.codice.org/content/repositories/releases/ddf/distribution/ddf>

Thanks,

-The DDF team

DDF Roadmap



DDF Static Analysis and Build

Github Badges

Github/Markdown support the use of badges to indicate status on a project. This page will be used to track potential badges to add to DDF and their status.

Name	Description	Status
Travis CI	Continuous Integration Builds	Added to all Repositories

Coveralls	Unit Test Coverage. Tested using Cobertura.	Tested on forked ddf-platform
Coverity	Static Analysis	Added to all Repositories. Links to scan.coverity.com projects.
VersionEye	Version Dependency notifications.	Tested on forked ddf-platform. Is not resolving Maven properties. For example: \${project.version}

<https://github.com/shaundmorris/ddf-platform/blob/master/README.md>

Static Analysis Tools

Name	Description	Status
Checkstyle	Rule enforcement (e.g. code formatter)	Included in all repositories
Findbugs	Static Analysis	Tested on ddf-platform

CSW Endpoint

Wiki Page Template

Use the following template if you are creating a new page for the DDF wiki, including the table of contents located within a panel (the first item on this page):

On This Page
<ul style="list-style-type: none"> • Overview • Some Topic <ul style="list-style-type: none"> • Some Sub-Topic <ul style="list-style-type: none"> • Some Sub Sub-Topic • Labels <ul style="list-style-type: none"> • user-guide and admin-guide • user-guide-content • Capitalization • Tables • Procedures • Common Mistakes <ul style="list-style-type: none"> • I'm Talking to You • Don't Ask Nicely • Two through 65489946 • Login vs. Log in • I.E. vs. E.G. • Which vs. That

Overview

This is the first topic, it's an H1, and it's required. Put an explanation of the topic that this page describes here and name the heading "Overview".

Some Topic

Some Sub-Topic

Some Sub Sub-Topic

Always use topics in order. Do not create a heading 1 then skip to heading 3, for example.

Labels

The Wiki provides a "label" function to tag the bottom of a page with categorical information that can be used to automatically generate pages on specific topics or for specific user categories. Generally, every page should have at least one label, but can support more. Some of these labels are used to pull pages into dynamic content areas and make the documentation structure more flexible.

user-guide and admin-guide

These labels are used on the top-level guide pages to generate custom pages of guides. Additionally, there are **int-guide** and **dev-guide** tags as well.

user-guide-content

These labels are used to mark content within one of the guides.

Capitalization

The DDF wiki uses the Chicago Manual of Style when capitalizing headings. All words are capitalized with the exception of articles (a, the, an, etc.) and prepositions (through, below, above, within, etc.). Here is a link to a website that correctly capitalizes headers, if there is some confusion: <http://titlecapitalization.com/>.

Tables

Use the following table format:

- Center headers
- Bold headers
- Use headers similar to the ones below

Item	Description	Other Information
Blah	A description of the item I'm using because it's important.	Optional other information

Procedures

Number each procedure and try to start the sentence with a verb phrase. Try not to place more than one action in a step. Place screenshots below the corresponding step, if necessary. Also, after the action in the step is performed, it's nice to give a description of the outcome. Example:

Complete the following procedure to log on to the system.

1. Power on the system. A splash screen opens (outcome).
2. Select the appropriate username in the list. The password window opens (outcome).
3. Enter the password for the user.
4. Select the **Log In** button. The desktop is displayed.
5. Open a terminal window.

Do NOT put a step, procedure, or important screenshot in an Info, Warning, or Note box. Too many of these boxes creates overemphasis, which makes people ignore them.

This Is a Note

Buttons, fields, and selection items should be emphasized in **bold** font, without any other emphasis. Examples:

Incorrect: Select the *Log In* button.

Incorrect: Select the "Log In" button.

Incorrect: Select the Log In button.

Correct: Select the **Log In** button.

6. Type any code examples inside a code block macro. Example:

```
<p>Code blocks are cool.</p>
```

7. If a code block is not necessary for the information being provided, use monospace text. Refer to step 8 for an example.
8. Type `cd /home/run` in the terminal window.
9. To create monospace text, highlight the text to change then select the **More** drop-down menu in the formatting section of the page.
10. Use a numerical indent only when the indent provides steps. Do not use numerical indents for informational purposes; use bullets.

Example:

- a. Create a new folder in `/home/run`.
- b. Name the folder `baseball`.
- A bullet is used for information and lists only. Do not put a procedure here.
- Bulleted lists and steps (including sub-steps) should have two or more points.

Common Mistakes

The mistakes explained in the following sections are often made when creating technical documentation.

I'm Talking to You

Try to keep the content of wiki pages anonymous (i.e., vague), if possible. Avoid using the word "you." Try to use a more inclusive term, such as "the user."

Don't Ask Nicely

Don't ask the reader to please do something. Tell them to do something. Examples:

- **Incorrect:** Log in to the system as an administrator. Please do not log in as a robot.
- **Correct:** Create a new user. Do not name the user after your grandmother.

Two through 65489946

Numbers zero through nine should be typed out (e.g., one, eight, nine, etc.). Numbers greater than zero should be shown numerically (e.g., 10, 24, 65489946). However, if a sentence starts with a number, type out the number or rearrange the sentence.

Login vs. Log in

Unfortunately, these terms are often used incorrectly. The term "login" refers to a user's ID. "Login" is a noun, not a verb. Use the term "log in" if performing an action. Examples:

- **Incorrect:** Login to the system to begin a fantastic voyage.
- **Correct:** Log in to the system to activate the time machine.
- **Correct:** Log in to the system by typing the correct login information into the **Login** field.

I.E. vs. E.G.

These initials are not interchangeable. I.e. is Latin and translates to "that is." An easy way to remember how to use i.e. is to think of the definition as "in essence." Use "i.e." when clarifying the preceding phrase or term. E.g. can be thought of as "example given." Use "e.g." when providing an example of the preceding phrase or term. Also, use a comma before and after i.e. and e.g. If the phrase following i.e. or e.g. is a complete

sentence, use parentheses to separate it from the rest of the sentence, or use a semicolon (;) before the i.e. or e.g. and a comma after i.e. or e.g.
Examples:

- **Incorrect:** The best way to capture a unicorn is by luring it with sweet treats, i.e., candy corn.
- **Correct** The best way to capture a unicorn is by luring it with sweet treats, e.g., candy corn.
- **Incorrect:** When I order pizza, I ask for all of the toppings (i.e., cheese).
- **Correct:** When I order pizza, I ask for all of the toppings (e.g., cheese, pepperoni, and bell peppers).
- **Incorrect:** Mysterious meat is the main ingredient in my favorite foods, i.e., hot dogs.
- **Correct:** Mysterious meat is the main ingredient in my favorite foods, e.g., Spam, hot dogs, and balogna.
- **Correct:** I like to take out trolls with a claymore, i.e., a directional weapon that explodes shrapnel.

Which vs. That

Use "which" when creating an "aside" for emphasis in a sentence. Asides are almost always enclosed by commas. When an aside is removed from a sentence, the sentence still makes sense. Use "that" everywhere else. Examples:

- Go to the garage and fill up the lawn mower that is green with gasoline. (This sentence implies that there is more than one lawn mower in the garage. The one to be filled up with gas is the one that is green in color, and the other lawn mowers in the garage are not green.)
- Go to the garage and fill up the lawn mower, which is green, with gasoline. (This sentence says that there is one lawn mower in the garage, and it is green in color. Notice that if you remove the aside from the sentence, it still makes sense: Go to the garage and fill up the lawn mower with gasoline.)
- **Correct:** The troll that is mean is sitting under the bridge. There is another troll, which is crying, sitting on the row boat.
- **Correct:** The candy that gives me a toothache is now in the garbage can. Candy corn, which gives me a toothache, is what I ate for dinner.

Administrator's Quick Start

On This Page

- Overview
- Prerequisites
- Install DDF
- Catalog Capabilities
- Use of the Content Framework
- Metrics Reporting

Overview

This quick tutorial will demonstrate:

- Installation
- Catalog Capabilities: Ingest and query using every endpoint
- Use of the Content Framework
- Metrics Reporting

Prerequisites

Review [Prerequisites](#) to ensure all system prerequisites are met.

Install DDF

1. Install DDF by unzipping the [zip file](#). 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 in place of this.
2. Start DDF by running the <DISTRIBUTION_INSTALL_DIR>/bin/ddf script (or ddf.bat on Windows).
3. Verify the distribution is running.
 - a. Go to <https://localhost:8993/admin>.
 - b. Enter the default username of "admin" (no quotes) and the password of "admin" (no quotes).
4. Follow the [install instructions](#) for more extensive install guidance, or use the command line console (which appears after the <DISTRIBUTION_INSTALL_DIR>/bin/ddf script starts) to install a few applications as mentioned below.

```
app:start catalog-app  
app:start content-app  
app:start solr-app
```

Other applications may be installed at a later time.

5. Go to <http://localhost:8181/services> and verify five REST services are available: admin, application, metrics, catalog, and catalog/query.
6. Click on the links to each REST service's WADL to see its interface.
7. In the Web Console (at </system/console/configMgr>), configure the system settings.
 - a. Enter the username of "admin" (no quotes) and the password "admin" (no quotes).
 - b. Select **Platform Global Configuration**.
 - c. Enter the port and host where the distribution is running.

Catalog Capabilities

1. Create an entry in the Catalog by ingesting a [valid GeoJson file](#) (attached to this page). This ingest can be performed using:
 - a. A REST client, such as Google Chrome's Advanced REST Client.
 - b. OR by using the following curl command to POST to the Catalog REST CRUD endpoint.

Windows Example

```
curl.exe -H "Content-type: application/json;id=geojson" -i -X  
POST -d @"C:\path\to\geojson_valid.json"  
http://localhost:8181/services/catalog
```

*NIX Example

```
curl -H "Content-type: application/json;id=geojson" -i -X POST -d  
@geojson_valid.json http://localhost:8181/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 2013 22:02:22 GMT
id: 44dc84da101c4f9d9f751e38d9c4d97b
Location:
http://localhost:8181/services/catalog/44dc84da101c4f9d9f751e38d9c4d97b
Server: Jetty(7.5.4.v20111024)
```

2. 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.
3. Verify the catalog entry exists by executing a query via the OpenSearch endpoint.
4. Enter the following URL in a browser </services/catalog/query?q=ddf>. A single result, in Atom format, should be returned.

Use of the Content Framework

Using the Content 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 [Web Console](#), select the Configuration tab.
2. Select the **Content Directory Monitor**.
3. Set the directory path to **inbox**.
4. Click the **Save** button.
5. Copy the attached [geojson_valid.json](#) file to the <DISTRIBUTION_INSTALL_DIR>/inbox directory.

The Content 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 that XML metadata for text searching is not automatically generated from GeoJson fields.

6. Verify GeoJson file was stored using the Content REST endpoint.
 - a. Install the feature `content-rest-endpoint` using the [Features tab](#) in the Web Console.
 - b. 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 <DISTRIBUTION_INSTALL_DIR>/content/store directory and copying the sub-directory in this folder (there should only be one).

*NIX Example

```
curl -X GET
http://localhost:8181/services/content/c90147bf86294d46a9d35ebbd44992c5
```

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

Metrics Reporting

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

1. Open the Web Console (</system/console/metrics>).
2. Select the **PNG** link for **Catalog Queries** under the column labeled **1h** (one hour). A graph of the catalog queries that were performed in the last hour is displayed.
3. Select the browser's back button to return to the Metrics tab.
4. Select the **XLS** link for **Catalog Queries** under the column labeled **1d** (one day).

Handy Tip

Based on the browser's configuration, the .xls file will be downloaded or automatically displayed in Excel.

CSW Source