Managing DDF

Table of Contents

License

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

Documentation Guide

Introduction

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/

Conventions

The following conventions are used within this documentation:

TIP	This is a **Tip**, used to provide helpful information.
NOTE	This is an **Informational Note**, used to emphasize points, remind users of beneficial information, or indicate minor problems in the outcome of an operation.
WARNING	This is an **Emphasized Note**, used to inform of important information.
IMPORTANT	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., `/admin`) within the documentation assume a locally running installation of DDF. Simply change the hostname if accessing a remote host.

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.

Applications

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

DDF Administrative Application

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

DDF Catalog Application

The DDF Catalog provides a framework for storing, searching, processing, and transforming information. Clients typically perform 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 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 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 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

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.

Administrators

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

Integrators

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

Developers

Developers will build or extend the functionality of the applications.

Quick Start

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.

Quick Start

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.
4. Go to https://localhost:8993/admin.
5. Enter the default username of "admin" (no quotes) and the password of "admin" (no quotes).
6. 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
   ```

   WARNING Other applications may be installed at a later time.
7. After the installation has been configured the instance should be restarted.

8. Go to https://localhost:8993/services and verify five REST services are available: admin, application, metrics, catalog, and catalog/query.

9. Click on the links to each REST service's WADL to see its interface.

10. In the Admin Console (at /admin), configure the system settings.

    a. Enter the username of "admin" (no quotes) and the password "admin" (no quotes).

    b. Select Platform app

    c. Select Platform Global Configuration.

    d. 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. OR

    b. 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" https://localhost:8181/services/catalog
    ```

    **NIX Example*

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

    Where: **-H** adds an HTTP header. In this case, Content-type header `application/json;id=geojson` is added to match the data being sent in the request. **-i** requests that HTTP headers are displayed in the response. **-X** specifies the type of HTTP operation. For this example, it is necessary to POST (ingest) data to the server. **-d** specifies the data sent in the POST request. The @ character is necessary to specify that the data is a file.

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

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

Sample Response

```
HTTP/1.1 201 Created
Content-Length: 0
Date: Mon, 22 Apr 2013 22:02:22 GMT
id: 44dc84da101c4f9d9f751e38d9c4d97b
Location: https://localhost:8993/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 metacard 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 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 metacard based on the metadata parsed from the ingested GeoJson file, and

   e. insert the metacard 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 https://localhost:8993/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. . Open the Web Console (/system/console/metrics). . 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. . Select the browser's back button to return to the Metrics tab. . Select the XLS link for Catalog Queries under the column labeled 1d (one day).

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

Using DDF

Version 2.8.2. Copyright (c) Codice Foundation :imagesdir: target/docs/images :toc: left :branding: DDF :icons: font :example-caption!: :source-highlighter: coderay :data-uri: :title-logo: logo_ddf.png :last-update-label: Copyright (c) Codice Foundation.

This page last updated:

License

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.

Understanding Metadata and Metacards

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

Populating Metacards (during ingest)

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

Searching Metadata

DDF provides the capability to search the Metadata Catalog (MDC) for metadata. There are a number of different types of searches that can be performed on the MDC, and these searches are accessed using one of several interfaces. This section provides are 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 effective time field	Newest to oldest	All Search Types
Distance	The catalog search result's distance field	Nearest to farthest	Point-Radius Spatial searches

Sort Policy	Sorts By	Default Order	Available for
Relevance	The catalog search result's relevance field	Most to least relevant	Contextual

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

WARNING

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

Prerequisites

- Supported platforms are *NIX - Unix/Linux/OSX, Solaris, and Windows.
- JDK8 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/jdk1.8.0
export JAVA_HOME
```

Example of Setting Up JAVA_HOME on Windows

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

WARNING

**NIX*

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

TIP

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
```

WARNING

Restart

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

Restart Command

```
init 6
```

IMPORTANT The Administration Web Console is not compatible with Internet Explorer.

Installing

TIP The *NIX commands listed next to the steps were performed on a default installation of Red Hat Enterprise Linux 5.4. Permission maintenance is not mentioned in this article, but all files should be owned by the running user, regardless of platform.

NOTE

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 running DDF kernel's <INSTALL_DIRECTORY>/deploy directory.

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

Use the DDF Distribution Zip to Install

- 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>.

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

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

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

su - ddf-user
```

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

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

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

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

```
cd D:\
```

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

**NIX*

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

Windows

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

- Change the current directory to the desired install location.

**NIX or Windows*

```
cd <INSTALL_DIRECTORY>
```

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

**NIX*

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

Example: Use Java to Unzip in Windows

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

- Run DDF using the appropriate script.

**NIX*

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

Windows

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

- 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 for status:

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>`.

NOTE	It is recommended that the root user create a new install directory that can be owned by a non-root user (e.g., ddf-user). The non-root user (e.g., ddf-user) can now be used for the remaining installation instructions.

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

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

su - ddf-user
```

- Change the current directory to 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:\
```

- 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>
```

- Change the current directory to the desired install location.

**NIX or Windows*

```
cd <INSTALL_DIRECTORY>
```

- 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\jdk1.8.0\bin\jar.exe" xf ddf-kernel-X.Y.zip
```

- Configure global properties in <INSTALL_DIRECTORY>/etc/system.properties .system.properties

```
org.codice.ddf.system.protocol=https://
org.codice.ddf.system.hostname=localhost
org.codice.ddf.system.httpsPort=8993
org.codice.ddf.system.httpPort=8181
org.codice.ddf.system.port=8993
org.codice.ddf.system.rootContext=/services

# Set the system information properties
org.codice.ddf.system.siteName=ddf.distribution
org.codice.ddf.system.siteContact=
org.codice.ddf.system.version=<latest version>
org.codice.ddf.system.organization=Codice Foundation
```

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

Additional Configuration Step

```
# Jetty Configuration
org.ops4j.pax.web.config.file={karaf.home}/etc/jetty.xml
```

- Run the DDF kernel using the appropriate script.

**NIX*

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

Windows

```
<INSTALL_DIRECTORY>/ddf-kernel-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 for status:

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] {branding} :: Distribution :: Web
Console (2.3.0)
                                       Hosts: 76
[ 113] [Active     ] [Created     ] [       ] [   80] {branding} :: Distribution ::
Console Branding Plugin (2.3.0)
```

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

DDF Kernel's Installed Features List

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

WARNING

DDF Application Installation Dependencies

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

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

- Install the Platform application by following the Platform Application Installation instructions.
- Install any optional applications that may be needed for the desired configuration.
  - Catalog Application Installation instructions

- Security Application Installation instructions
- Content Application Installation instructions
- Spatial Application Installation instructions
- Solr Catalog Application Installation instructions are included in each of the Solr Catalog configurations. Refer to the appropriate section for the desired Solr Catalog Provider configuration's installation instructions.
  - Embedded Solr Catalog Provider
  - External Solr Catalog Provider
  - Standalone Solr Server

- Proceed to Configuration.

(Option 2/Part 2) Configuration

1. Open a compatible web browser and log in to the Administrator Console (https://localhost:8993/admin) with username "admin" and password "admin" (no quotes).
2. Select the Configuration tab in the Administrator Console.
   a. 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 Catalog Framework 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
```

WARNING	*If displayed, the *DDF* Distribution :: Web Console* entry should be in the **Resolved** state. This is expected. The DDF Distribution Web Console is an OSGi bundle fragment. Bundle fragments are distinguished from other bundles in the command line console list by a new line under the its bundle status that states `Hosts`, followed by a bundle number. Bundle fragments remain in the **Resolved** state and can never move to the **Active** state.

Example: Bundle Fragment in the Command Line Console

```
[ 261] [Resolved   ] [            ] [       ] [   80] {branding} :: Distribution :: Web
Console (2.2.0)
                                    Hosts: 76
```

NOTE	For a complete list of installed features/bundles see the DDF Included Features document.

DDF Directory Contents after Installation

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.

Configuring

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

Configuring via the Admin Console

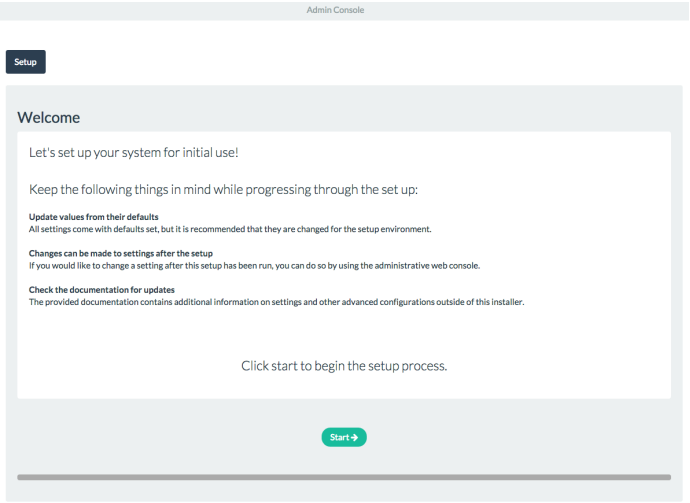
Accessing the Admin Console

Open the admin portal. * https://localhost:8993/admin * Enter Username and Password.

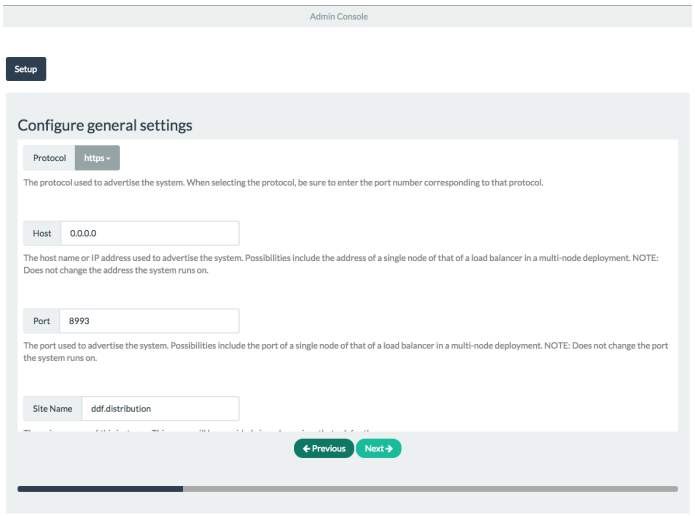
NOTE	The default username/password in admin/admin. To change this, refer to Password Management page.

Initial Configuration

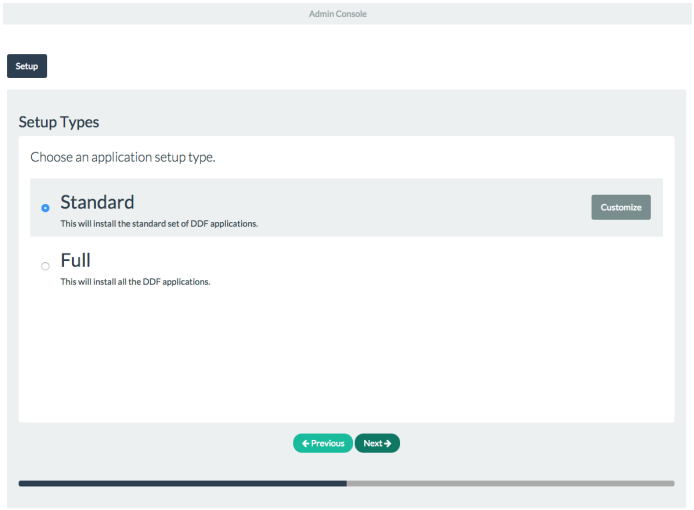
The first time the DDF administrator portal runs, the initial configuration steps appear. Click Start to begin.



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



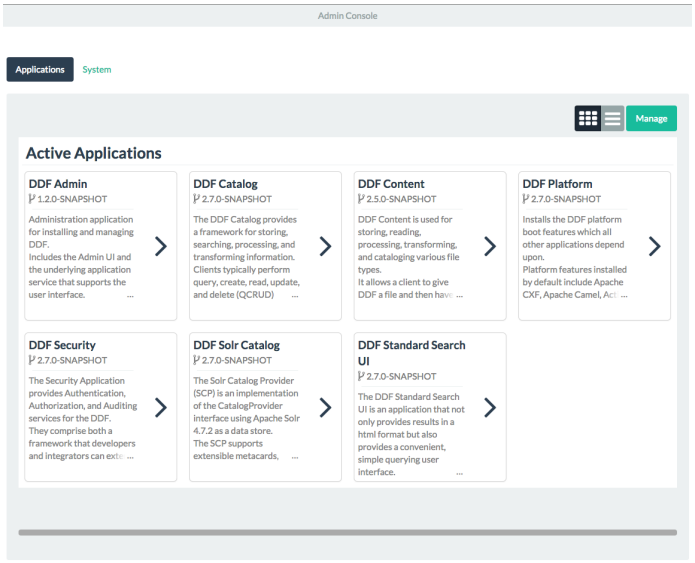
The final step of initial configuration is a display of all applications installed and their current status. This tree structure demonstrates how several applications depend on other applications.

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

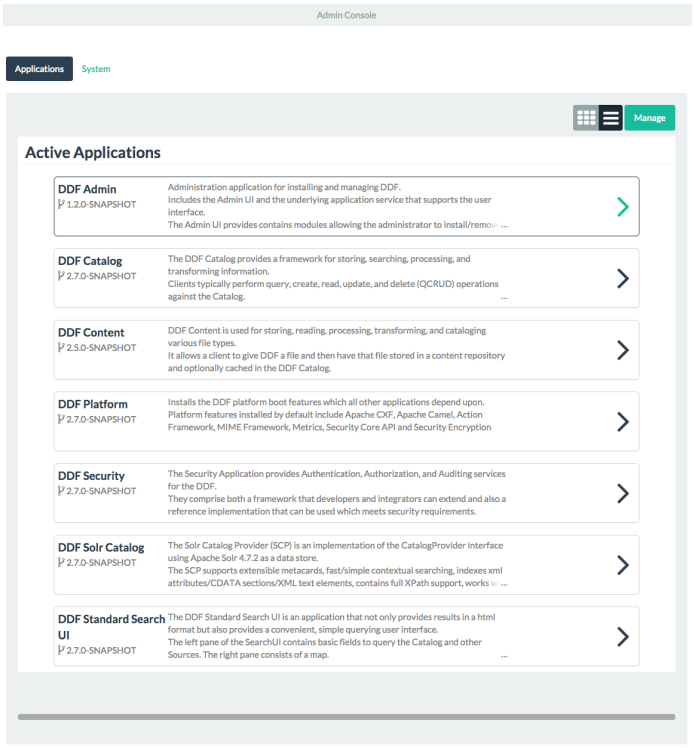
Tile View

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



List View

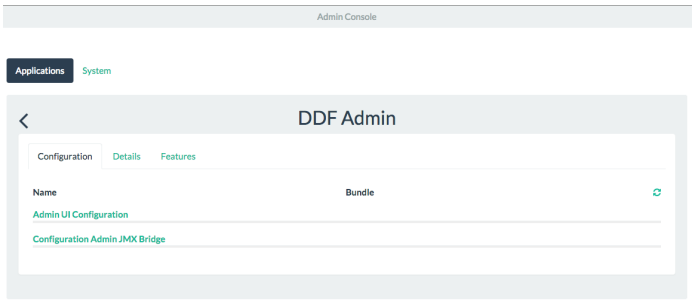
Optionally, active applications can be displayed in a list format by clicking the list view button.



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

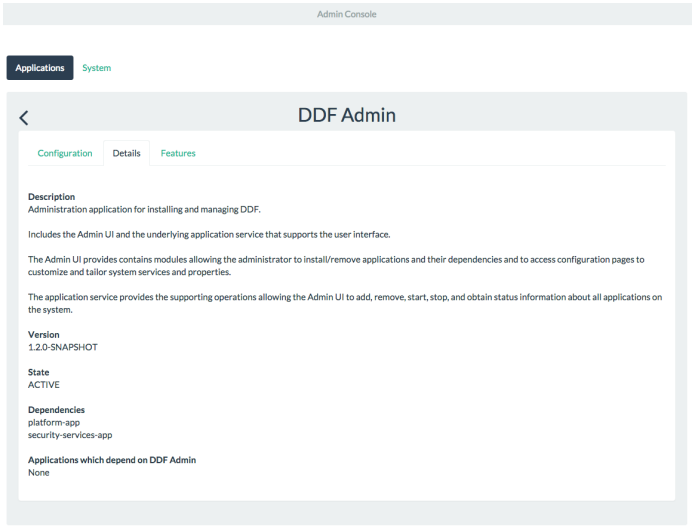
Configuration

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



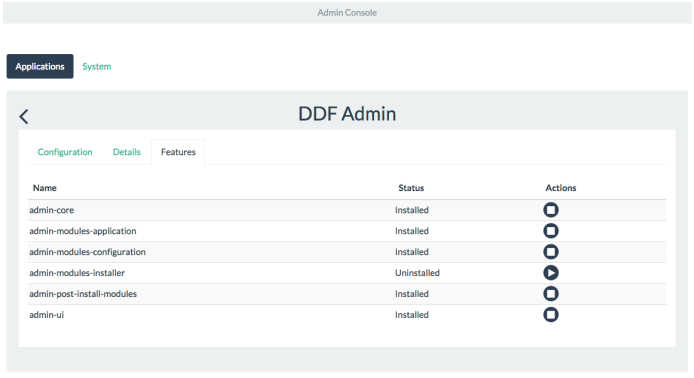
Details

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



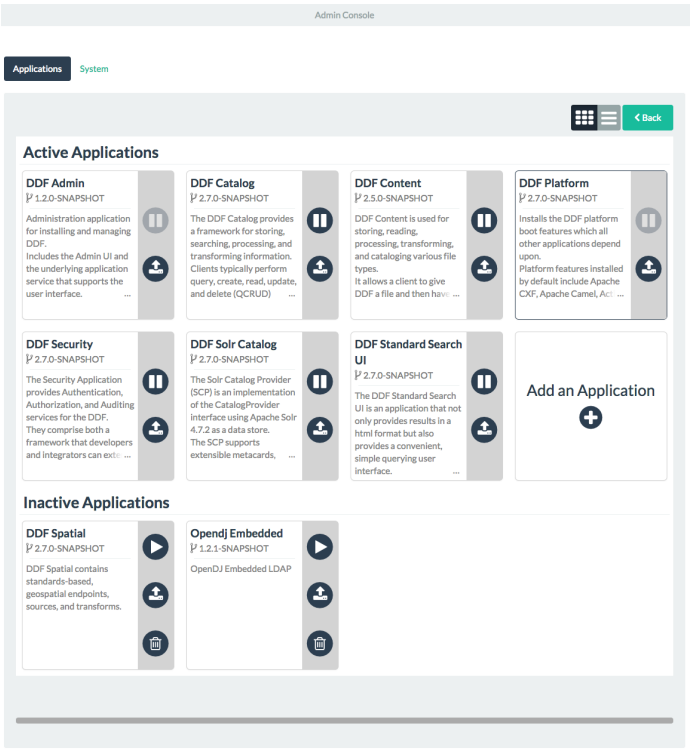
Features

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.

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

Adding Applications

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

Removing Applications

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

Upgrading Applications

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

System Settings Tab

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

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

Configuring DDF

Configure DDF Global Settings

Global configuration settings are configured via the properties file system.properties. These properties can be manually set by editing this file or set via the installer.

Configurable Properties

Title	Property	Type	Description	Default Value	Required
Protocol	org.codice.ddf.system.protocol	String	Default protocol that should be used to connect to this machine.	https://	yes
Host	org.codice.ddf.system.hostname	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.	localhost	yes
Default Port	org.codice.ddf.system.port	String	The default port used to advertise the system. This should match either the http or https port. NOTE: Does not change the port the system runs on.	8993	yes
HTTP Port	org.codice.ddf.system.httpPort	String	The http port used by the system. NOTE: This **DOES** change the port the system runs on.	8181	yes
HTTPS Port	org.codice.ddf.system.httpsPort	String	The https port used by the system. NOTE: This **DOES** change the port the system runs on.	8993	yes

Root Context	org.codice.ddf.system.rootContext	String	The the base or root context that services will be made available under.	/services	yes ////
Trust Store	trustStore	String	The trust store used for outgoing SSL connections. Path is relative to ddf.home.	etc/keystores/clientTruststore.jks	yes
Trust Store Password	trustStorePassword	String	The password associated with the trust store.	changeit (encrypted)	yes
Key Store	keyStore	String	The key store used for outgoing SSL connections. Path is relative to karaf.home.	etc/keystores/clientKeystore.jks	yes
Key Store Password	keyStorePassword	String	The password associated with the key store.	changeit (encrypted)	yes ////
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
Site Contact	site contact	String	The email address of the site contact.		no

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).

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

Install Features Using the Admin Console

1. Open the admin console.
   a. `https://localhost:8993/admin`
   b. Enter Username and Password.
2. Select the appropriate application.
3. Select the Features tab.
4. Select the play arrow under the Actions column for the feature that should be installed.
5. Wait for the **Status** to change from **Uninstalled** to **Installed**.

Uninstall Features

1. Open the admin console.
   a. `https://localhost:8993/admin`
   b. Enter Username and Password.
2. Select the appropriate application.
3. Select the stop icon under the Actions column for the feature that should be uninstalled.
4. Wait for the **Status** to change from **Installed** to **Uninstalled**.

Add Feature Repositories

1. Open the web administration console.
   a. `https://localhost:8993/system/console`
   b. Enter Username and Password
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

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

Follow these steps to configure DDF using the system console.

NOTE | 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).

NOTE	*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`

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

```
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
```

1. Install the desired feature.
   `ddf@local>features:install ddf-source-dummy`

2. 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
```

1. Check the bundle status to verify the service is started.
   `ddf@local>list`

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

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

Uninstall Features

1. 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
```

1. Uninstall the feature.
   `ddf@local>features:uninstall ddf-source-dummy`

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

1. Verify that the feature has uninstalled properly.
   `ddf@local>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

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

HTTP Port Configuration

IMPORTANT	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

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

Enable and Configure HTTP to HTTPS Proxy

What it does: This feature proxies http to https.

When to use: Use this feature when you have legacy clients that can't use HTTPS.

How to install this feature using the DDF terminal:

```
Note: If DDF has not been installed, use the   How to install this feature using the
AdminUI   guide found below
```

1.) Launch the DDF application.

2.) In the DDF console, type the command "features:install platform-http-proxy"

How to install this feature using the AdminUI:

1.) Navigate your browser to https://localhost:8993/admin/index.html. The admin console appears.

1.a) If this is not a new install, skip to step 9

2.) At the admin console, click on "Start" to begin the setup process. The general settings page will

appear.

3.) Configure the general settings by entering in values for the “Protocol”, “Host”, “Port”, “Site Name”, and “Organization.” Descriptions for these settings can be found on the “Configure general settings” page.

4.) Click “Next” and the “Setup Types” page appears.

5.) Choose whichever setup type that corresponds to the suite of applications you would like installed.

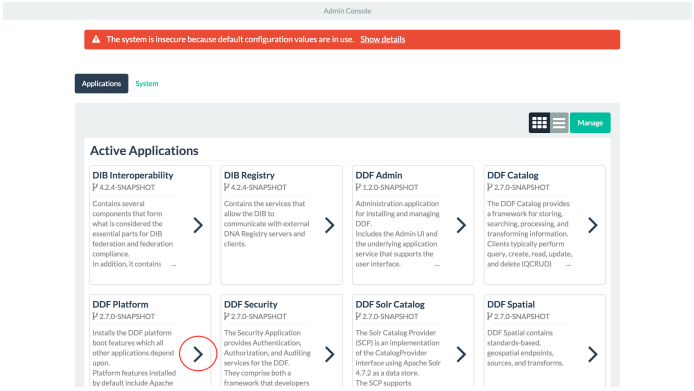
6.) [Optional] Click “Customize” and a window appears allowing you to customize which features will be included in your installation.

7.) Click “Next” and the installation begins

8.) Once the installation is finished, click “Finish” to conclude the setup and the page will refresh.

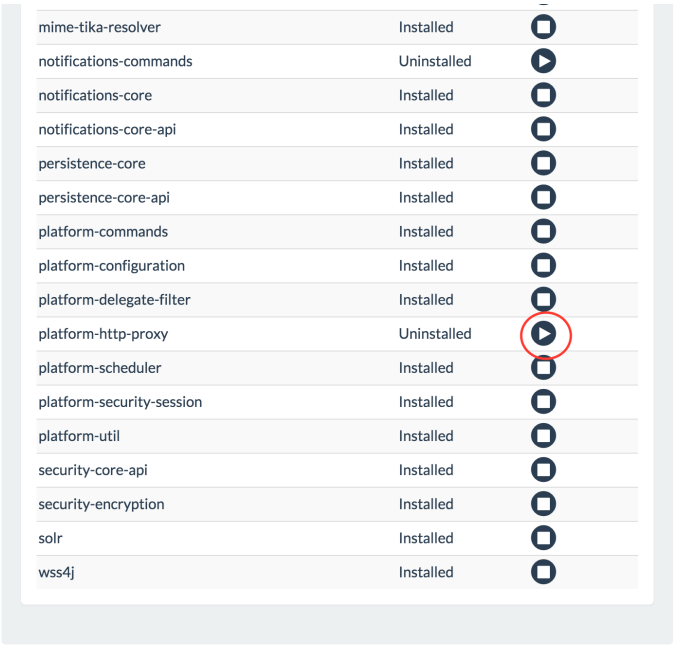
9.) On the left hand side of the screen, select the “Applications” tab.

10.) Under “Active Applications” choose the “>” arrow under “DDF Platform”. This will bring up the configuration page for the DDF Platform application. This arrow is shown here circled in red:



11.) Under the “DDF Platform” heading, choose the “Features” tab. The features tab shows us all the features we can install to the currently selected app.

12.) Scroll through the list of features and find “platform-http-proxy.” The feature will be listed as “Uninstalled” click on the arrow button to the right of the word “Uninstalled” in order to install the platform-http-proxy. This arrow button is shown here circled in red:



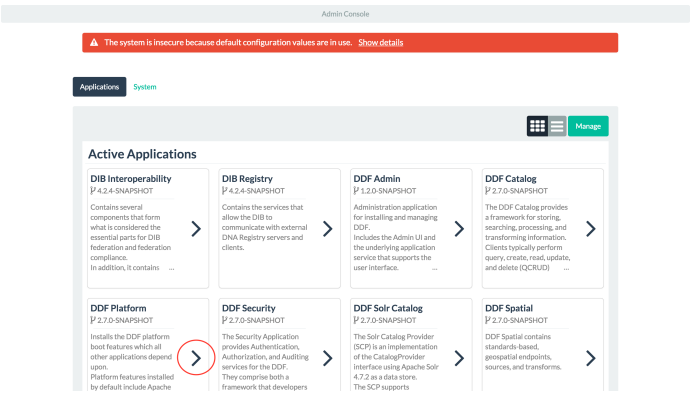
Configuring the proxy:

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

1.) Navigate your browser to https://localhost:8993/admin/index.html. The admin console appears.

2.) On the left hand side of the screen, select the “Applications” tab.

3.) Under “Active Applications” choose the “>” arrow under “DDF Platform”. This will bring up the configuration page for the DDF Platform application. This arrow is shown here circled in red:



4.) Under the “DDF Platform” heading, ensure that the “Configuration” tab is selected.

5.) Under the “Name” heading, select “HTTP to HTTPS Proxy Settings” and the settings menu appears.

6.) Under “Hostname”, enter in the Hostname to use for HTTPS connection in the proxy.

7.) Click “Save changes” to save the Hostname.

Enable SSL for Clients

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

Property	Sample Value	Description
javax.net.ssl.trustStore	etc/keystores/serverTruststore.jks	The java keystore that contains the trusted public certificates for Certificate Authorities (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/serverKeystore.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

Property	Sample Value	Description
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

Configuring DDF with New Certificates

DDF ships with a default security certificate configured to identify the DDF instance machine as "localhost." This allows the DDF distribution to be unzipped and run immediately in a secure manner. If the installer was used to install the DDF and a hostname other than 'localhost' was given, a new cert for the hostname was generated and added to the keystore. If the hostname was left as 'localhost' or the hostname was changed after installation, in order to access the DDF instance from another machine over HTTPS (now the default for many services) the default certificates need to be replaced with a certificate that uses the fully qualified hostname of the server running the DDF instance.

Important Terms

Term	Definition	Example
DDF_HOME	The path to the unzipped DDF distribution	/opt/ddf/ddf-2.6.0
alias	The nickname given to a certificate within a keystore to make it easily identifiable. Normally the alias should be the DDF instance’s FQDN.	localhost

Term	Definition	Example
certificate	A combination of an entity's identity information with the entity's public key. The entity can be a person, organization, or something else, but in this case the entity is a computer on the network. To be valid, a certificate must be digitally (cryptographically) signed by a certifcate authority. By signing a certificate, the CA attests that the public key truly belongs to the entity and no one else. See also PKIX.	<FQDN>.crt
CN	Common Name - The FQDN of the DDF instance as defined within the Certificate.	search.codice.org
certification path	A list of certificates, starting with the server's certificate and followed certificate of the CA who signed the server's CSR. The list of certificates continues, with each subsequent certificate belonging to the CA that signed the current CA's certificate. This chain continues until it reaches a trusted anchor, or root CA certificate. The chain establishes a link between the trust anchor and the server's certificate. See IETF RFC 4158 for details.	
chain of trust	See certification path.	
CSR	Certificate Signing Request. A certificate that has not yet been signed by a certificate auhority.	<FQDN>.csr
digital certificate	See **certificate**.	
FQDN	Fully Qualified Domain Name	search.codice.org
HTTPS	Hyper-Text Transfer Protocol Secure. An encrypted alternative to HTTP. The HTTP connection is encrypted over TLS. See IETF RFC 2818 for more information.	https://

Term	Definition	Example
JKS	Java **keystore**. A dictionary of cryptographic objects (e.g. private keys, certificates) referenced by an **alias**. The JKS format is specific Java.	
keystore	Refers to either a JKS keystore or a PKCS#12 keystore. For the purposes of these instructions a keystore is always a file.	
keytool	The Java keytool is a key and certificate management command line utility.	
openssl	The openssl program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell.	
PKCS#12	Personal Information Exchange Syntax. A standard that allows certifcates, private keys, and optional attributes to be combined into a single file. See IETF RFC 7292 for more information.	<FQDN>.p12
PKIX	A public key infrastructure also know as X.509. It is documented in the IEFT RFC 5280 and defines what a **certificate** is.	
PORT	TCP Port of service	8993
security certificate	See **certificate**.	
TLS	Transport Layer Security protocol. Provides privacy and data integrity between client and server. See IETF RFC 5246 for more information.	

Update DDF Configuration

Configure DDF Web Service Providers

By default Solr, STS server, STS client and the rest of the services use the system property 'org.codice.ddf.system.hostname' which is defaulted to 'localhost' and not to the fully qualified domain

name of the DDF instance. Assuming the DDF instance is providing these servcies, the configuration must be updated to use the DDF instance's **fully qualified domain name** as the service provider.

This can be done by editing the system.properties in <INSTALL_DIRECTORY>/etc/

IMPORTANT	The process of changing the configuration can cause users to loose access to the DDF instance via the Web. This includes losing access to the **Admin UI** (https://localhost:8993/admin) and the **Felix console** (https://localhost:8993/system/console). Without access to these bundles, it is not possible to configure the DDF instance through a Web browser.
TIP	Even if Web access is lost, the DDF instance can still be configured using the DDF command line console.

1. Start the DDF instance, if it is not already running.
2. Go to https://localhost:8993/admin and step through the installer setting the hostname to the instance's FQDN and the port to correct value.

Configure Files in HOME Directory Hierarchy

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

- In `<DDF_HOME>/etc/user.properties`, modify the line:

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

To be:

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

- Next ,configure `<DDF_HOME>/etc/system.properties`

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

Create Private Key and Certificate Signing Request

NOTE These steps assume that `openssl` is installed on the machine as a command line utility.

Open a console or terminal and change directory

```
$> cd <DDF_HOME>/etc/certs
```

Create a new RSA key pair (public and private keys) and use the public key to create a certificate signing request. This happens with a single command to `openssl`

```
$> openssl req -newkey rsa:2048 -keyout <FQDN>.key -out <FQDN>.csr
```

The command will encrypt the private key to protect it. The command prompts for a pass phrase that will decrypt the private key:

```
writing new private key to '....key'
Enter PEM pass phrase:
```

Enter a pass phrase (this documentation uses the pass phrase `changeit`).

The command then prompts for identity information:

- For country, use the same two-letter ISO code as the certificate authority. For the Demo CA, use "US".
- For the "Common Name" or "CN" enter the Fully Qualified Domain Name (FQDN) of the system. An example of a FQDN would be "search.codice.org".

This command creates two files:

- `<FDQN>.csr`, the certificate signing request. The CSR includes the public key that matches the private key.
- `<FQDN>.key`, a private key cryptogrpahically linked to the CSR

Sign the CSR to Create a Valid Certificate

NOTE This step assume Demo CA signs the CSR. If the CSR is signed by a different CA, skip this step.

```
$> openssl ca -config openssl-demo.cnf -policy policy_anything -passin "pass:secret" -in
<FQDN>.csr -out <FQDN>.crt
```

The command prompts:

```
    >Certificate is to be certified until Jul 23 21:06:37 2016 GMT (365 days)
    >Sign the certificate? [y/n]:
```

Enter **y** and select return. It responds:

```
    >1 out of 1 certificate requests certified, commit? [y/n]
```

Enter **y** and select return again.

```
    >Write out database with 1 new entries
    >Data Base Updated
```

A new `.crt` file is writen to the current directory. By default, the certificate is valid for one year. This period can be changed using `openssl` command line options or by editing the `openssl` configuration file, `openssl-demo.cnf`.

Create PKCS#12 File

The private key, certificate need to be combined into a new structure that can be imported into the keystore. A chain of trust file also needs to be created and combined with the other objects. A file that conforms to the PKCS#12 specification is created from these objects as an intermediary step in the process.

Creating the Chain of Trust

- If the certificate was signed by the Demo CA, skip this step. It is not necessary in because `openssl` adds the issuer's (signer's) certificate to the PKCS12 file to create the complete chain of trust.
- Establasihing a **chain of trust** is imporant if a root CA did not directly sign the CSR. It is important for establishing a connection between different branches/orgganizations/departments.

NOTE ==== Although a single certificate can have only one issuer (cannot have more than one CA signature), different certificate chains can exist for the same target certificate because more than one certificate can exist containing the same subject and public key. ====

- Concatenate the certificate from the issuer, all intermediate certificate authorities, and the root authority into a text file.

```
$>  cat <root CA PEM> <intermediate CA 1 PEM> <intermediate CA 2 PEM> > chain.txt
```

This command created the file `chain.txt`. This file an intermediate file used to create the PKCS#12 file.

WARNING

If a certificate chain file was created, add the following options to the next command:

```
-chain -CAfile chain.txt
```

Create the PKCS#12 file. The argument for `-passin` is the password that was used to encrypt the private key. The argument for `-passout` sets the password used to decrypt the PKCS#12 file.

```
$> openssl pkcs12 -in <FQDN>.crt -inkey <FQDN>.key -certfile demoCA/cacert.pem -out
<FQDN>.p12 -export -name <FQDN> -passin "pass:changeit" -passout "pass:changeit"
```

TIP

Use `openssl` can verify a chain of trust for a certificate. For example:

```
openssl verify -CAfile ./demoCA/cacert.pem <FQDN>.crt
```

Install Signed Certificate Into Key Store

Use the Java keytool to import the signed certificate into the server keystore file. The keytool needs the password word to decrypt the PKCS#12 file (`-srcstorepass`) and the password to decrypt the existing server keystore.

NOTE

This example assumes the passwords are `changeit`. This example also assumes the server keystore is in the default location and has the default name, `<DDF_HOME>/etc/keystores/serverKeystore.jks`

```
$> keytool -importkeystore -srckeystore <FQDN>.p12 -srcstoretype pkcs12 -destkeystore
<DDF_HOME>/etc/keystores/serverKeystore.jks -srcalias <FQDN> -deststorepass changeit
-srcstorepass changeit
```

TIP

Check the contents of a Java keystore with the keytool:

```
keytool -list -keystore <{branding}_HOME>/etc/keystore/serverKeystore.jks
```

Add `-v` to list the entire contents of the key store entries.

```
keytool -list -v -keystore <{branding}_HOME>/etc/keystore/serverKeystore.jks
```

Remove localhost Key Entry

The server key store comes configured with a key for localhost. It should be removed when a certificate with the server's FQDN certificate is installed. If the localhost key exists, the Solr server uses the that key to sign messages. The browser and other client processes will halt when trying to connect to Solr because the client expects common name to be the Solr processes's <FQDN>. Use the keytool to remove the localhost entry:

```
$> keytool -delete -keystore <{branding}_HOME>/etc/keystore/serverKeystore.jks -alias
localhost
```

Import CA Certificates into the Keystore

- The Demo CA certificate is already imported into the keystore file that is included in the DDF distribution. If the FQDN certificate was signed by the Demo CA, skip this step.

Import CA Certificates into the Trustsore

- The Demo CA certificate is already imported into the truststore file that is included in the DDF distribution. If the FQDN certificate was signed by the Demo CA, skip this step.
- The truststore is created from the CA certificates. The CA should be the only entry needed in the trust store.

```
# import each CA into truststore
$> keytool -import -trustcacerts -alias <CA alias> -file <CA PEM> -keystore
serverTruststore.jks
```

NOTE	When federating with other DDF instances that do not share the same CA, you will need to import the CA certs from the other federated instances and they will need to import yours.

Restart and Test

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

WARNING	If the server's fully qualifiied domain name is not recognized, the name may need to be added to the netowrk's DNS server.

TIP

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

```
ping <FQDN>
```

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

```
127.0.0.1    <FQDN>
```

Configuring a Java Keystore for Secure Communications

NOTE

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

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`

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

Create a Truststore

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

- Add system console to whitelisted contexts.
  - `https://localhost:8993/admin`
    - Select DDF Security.
    - Select **Configuration** tab.
    - Select the **Web Context Policy Manager**.
    - Add `/system/console` to the Whitelisted Contexts

NOTE

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

Enable Backup Plugin: true

Assumes hostname of 'ddf'

- Configure Catalog External Solr Catalog Provider
  - Change the HTTP URL to: `https://ddf:8993/solr`
- Configure Persistent Store
  - Solr URL: `https://ddf:8993/solr`
- Configure Catalog Federation Strategy
  - Change Solr URL to: `https://ddf:8993/solr`
- Configure Security STS Client
  - Change the STS WSDL Address to: `https://ddf:8993/services/SecurityTokenService?wsdl`
- Configure Security STS Server
  - SAML Assertion Lifetime: `86400`
  - Change Token Issuer to: `ddf`
  - Change Signature Username to: `ddf`
  - Change Encryption Username to: `ddf`
- Configure Security STS LDAP Login

    - `features:install security-sts-ldaplogin`
    - LDAP URL: `ldaps://ddf:1636`
    - SSL Keystore Alias: `ddf`
    - Configure Platform Global Configuration
    - Protocol: `https`
    - Host: `ddf`
    - Port: `8993`
- Configure the Web Context Policy Manager
    - In White Listed Contexts, add /sso
- Configuring the Embedded LDAP

IMPORTANT

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

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

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

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

    - `ds-cfg-key-store-type: JKS`
    - `ds-cfg-key-store-pin: password`
    - `cn: JKS`
- The default password in `config.ldif` for `serverTruststore.jks` is `changeit`. This needs to be modified to password.
    - `ds-cfg-trust-store-file: ../../keystores/serverTruststore.jks`
    - `ds-cfg-trust-store-pin: password`
    - `cn: JKS`
- If using the default keystores and certificates, start the `opendj-embedded app:start opendj-embedded`
- Shutdown DDF
    - `<ddf-home>/bin/shutdown -f`
- Add the newly created keystore and truststore
    - put the newly created `serverKeystore.jks` in `<ddf-home>/etc/keystores`
    - put the newly created `serverTruststore.jks` in `<ddf-home>/etc/keystores`
- Configure system properties
    - In `<ddf-home>/etc/system.properties`, modify the keystore and truststore paths and passwords as appropriate.

system.properties

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

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

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

- Configure users properties
  - In `<ddf-home>/etc/users.properties`, change `localhost=localhost,admin` to `ddf=ddf,admin`
- Start ddf
  - For Windows, run `<ddf-home>/bin/ddf.bat`
  - For *Nix, run `<ddf-home>/bin/ddf`
  - Admin Console: `https://ddf:8993/admin`
  - Search UI: https://ddf:8993/search
- Remove system/console from whitelist
  - `https://localhost:8993/admin`
    - Select DDF Security.
    - Select **Configuration** tab.

Select the **Web Context Policy Manager**.

- Remove `/system/console` from the Whitelisted Contexts

Configuring Solr Catalog Provider

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 DDF Application and Configuration Clustering

An essential part to the DDF Clustering solution is the ability to manage applications and configurations among cluster nodes. The following documentation will help in setting up a set of DDF server nodes in a cluster, and allow an administrator to manage the applications and configurations that are deployed.

Set Up DDF Cluster

Initial Setup

The goal of the DDF Cluster solution is to keep applications and configurations synchronized on every DDF server. When setting up a DDF server, it is recommended that only one server is setup per machine (virtual or non-virtual). This means that for each physical machine or virtual machine (VM) that is setup, only one instance of the DDF is deployed on it. This configuration provides the least complexity in configuration of the DDF Cluster and allows the DDF server itself to utilize the resources of the entire system. It is also recommended that all DDF servers and systems be configured the same. This means that all DDF server platforms have the same configurations and applications running initially. Also, ensure that all physical and virtual systems running the DDF servers have the same configuration (e.g., operating system, memory, CPU, etc.).

Start and Configure New DDF Server Node Clusters

Before starting your DDF cluster nodes, you must first setup your node network configurations. See the Configuring DDF Clustering For Multicast & Unicast section for more information. To view all of the available nodes, navigate in your browser to the DDF Admin Console (https://localhost:8993/admin) and click on the tab labeled “Clustered Groups”. Under the group “default”, you should see all running DDF nodes that have been clustered. It is also possible to move all instances into a named group. In order to perform this action, you must have access to one of the DDF command line shells directly or utilize Gogo located at https://localhost:8993/system/console/gogo.

```
features:install cellar
```

The first action that will be performed is the creation of a new group. To create a new cluster group named “mygroup”, execute the following command:

```
cluster:group-create mygroup
```

This command will create a new cluster group which will not contain any nodes. Execute the following command to view all groups:

```
cluster:group-list
```

You should see the following output:

```
Group Members
* [default ] [192.168.1.110:5701* ]
[mygroup ] []
```

As you can see, there are now two groups, the default group and the new group that you have just created. We now need to move the DDF node out of the “default” group and into “mygroup”. Execute the following commands:

```
cluster:group-join mygroup 192.168.1.110:5701

cluster:group-quit default 192.168.1.110:5701
```

The first command allows the DDF node to join the new group. The second command removes the DDF node from the “default” group.

```
Group Members
[default ] []
* [mygroup ] [192.168.1.110:5701* ]
```

These commands should be executed on all “production” nodes located in the default group. Note: The default group will always remain and cannot be deleted. It is possible for DDF nodes to be separated into multiple groups. It is also possible for one DDF node to exist in multiple groups. These are advanced topics and can be addressed in additional documentation links.

Add a DDF Server Node to the Cluster

There may be a need for an additional DDF server node after an existing DDF cluster has already been configured and deployed. It is important that when adding additional nodes, the new node must match the existing nodes in terms of applications and configurations. Therefore it is good practice to copy one of the existing nodes and push that copy to a new virtual machine or server machine instance. This will provide a stabler transition of the new node into the cluster. Once you have setup the new node, you can follow the instructions above to add the new node into the cluster group, if needed.

Manage Applications

Application management within the cluster has been designed to function as if you were only managing one instance of DDF. All DDF applications are handled at the feature level. Therefore, the Features section of the Admin Console can be used to manage all applications within the DDF Cluster.

The Features section of the Web Console can be accessed by navigating to https://localhost:8993/system/console/features/features in a web browser. Credentials may be required (username and password) to access the console.

From the Features console, the following functions are available:

- Provides a listing of the available features and repositories in the cluster
- Add new repositories to the cluster
- Remove existing repositories from the cluster
- Install features from repositories into the cluster
- Uninstall or remove features from the cluster

For more information on using the console, refer to the DDF User documentation.

Manage Configurations

Just as with Application Management, managing configurations within a cluster was designed to function as if you were configuring one instance of DDF. There are two areas where DDF configurations can be modified. Most modifications will occur within the Configurations section of the Web Console. The Web Console can be accessed by navigating to https://localhost:8993/system/console in a web browser. Credentials may be required (username and password) to access the console.

From the Configuration console, the following features are available:

- Provides a listing of the available configurations
- Edit configuration values in the cluster
- Unbind the configuration from the bundle
- Remove the configuration from the cluster

For more information on using the console, refer to the DDF User's Guide.

Control Feature and Configuration Synchronization

There may be instances where certain configurations are to remain local to a certain DDF node. This behavior can be controlled through the cellar groups configuration. To open this configuration, navigate to Within the configuration list, search for the configuration with name "org.apache.karaf.cellar.groups". Click on the configuration to view / edit. Within the file you will see many configurations listed with the following format:

```
[cluster group name] . [configuration type (e.g. feature, configuration, etc.)]. [list
type] = [values]
```

These configurations allow you to control the synchronization of features and configurations through blacklists and whitelists. If you do not want a specific feature or configuration to propagate throughout your cluster group, you can put it into a blacklist. By default for all cluster groups, all features are in the white list:

```
mygroup.features.whitelist.outbound = *
```

with the exception of "`cellar`":

```
mygroup.features.blacklist.outbound = cellar
```

As for configurations, by default all configurations are whitelisted with the exception of the following:

```
mygroup.config.blacklist.outbound = org.apache.felix.fileinstall*,
org.apache.karaf.cellar.groups, org.apache.karaf.cellar.node,
org.apache.karaf.management, org.apache.karaf.shell, org.ops4j.pax.logging
```

Once you have made your changes, you can save the configuration by pressing the "Save" button.

Additional Details

Configure DDF Clustering for Unicast and Multicast

By default, DDF clustering utilizes TCP-IP unicast for discoverying other DDF nodes. The hazelcast.xml file located under <DDF root>/etc/ contains the port and address configurations for network setup. The TCP-IP unicast mode has been setup to allow for manual configuration and control of initial clustering. This configuration is also beneficial for cases where a particular network cannot support multicast or multicast has been turned off for certain reasons. There is a configuration which allows auto-discovery of DDF nodes and utilizes multicast as a transport. The hazelcast.xml file is configured like the following to allow for TCP-IP unicast discovery of cluster nodes:

```
<join>
<multicast enabled="false">
<multicast-group>224.2.2.3</multicast-group>
<multicast-port>54327</multicast-port>
</multicast>
<tcp-ip enabled="true">
<interface>127.0.0.1</interface>
</tcp-ip>
<aws enabled="false">
<access-key>my-access-key</access-key>
<secret-key>my-secret-key</secret-key>
<region>us-east-1</region>
</aws>
</join>
```

As you can see, the multicast option has been set to false and the tcp-ip option is set to true. All systems that will participate in the cluster need to have their ip addresses listed within the interface section highlighted. These modifications must be made for each node. Once these modifications have been made to the hazelcast.xml file, it is recommended that the nodes be restarted. The following hazelcast.xml configuration would be used for multicast auto-discovery:

```
<join>
<multicast enabled="true">
<multicast-group>224.2.2.3</multicast-group>
<multicast-port>54327</multicast-port>
</multicast>
<tcp-ip enabled="false">
<interface>127.0.0.1</interface>
</tcp-ip>
<aws enabled="false">
<access-key>my-access-key</access-key>
<secret-key>my-secret-key</secret-key>
<region>us-east-1</region>
</aws>
</join>
```

As you can see, the multicast option has been set to true and the tcp-ip option is set to false. A multicast group and port can be specified in the file as highlighted above. These modifications must be made for each node. Once these modifications have been made to the hazelcast.xml file, it is recommended that the nodes be restarted.

Verify Synchronized DDF Nodes

In most cases, the DDF system console should provide you with a listing of all features, repositories, and configurations that are installed on the cluster. There are times when the cluster can become out

of sync. This instance may occur if a system has been offline for some time. One way to verify the synchronized lists of the cluster is to run cluster commands from the command line. In order to perform these actions, you must have access to one of the DDF command line shells directly or through the use of Gogo (http://localhost:8993/system/console/gogo). Once at the command line, execute the following command to see the list of deployed features for your cluster:

```
cluster:feature-list mygroup
```

This command will list the available features for your cluster group “mygroup”.

```
Features for cluster group mygroup
Status Version Name
[installed ] [2.2.0 ] catalog-opensearch-endpoint
......
```

To view the cluster group’s configurations, execute the following command:

```
cluster:config-list mygroup
```

This command will show all shared configurations among the cluster group “mygroup”.

```
----------------------------------------------------------------
Pid: org.ops4j.pax.url.mvn
Properties:
org.ops4j.pax.url.mvn.useFallbackRepositories = false
service.pid = org.ops4j.pax.url.mvn
org.ops4j.pax.url.mvn.disableAether = true
----------------------------------------------------------------
Pid: org.apache.karaf.webconsole
Properties: ....
```

The following command will list all repositories associated with the cluster group “mygroup”:

```
cluster:features-url-list mygroup
The following will be displayed:
mvn:org.apache.cxf.karaf/apache-cxf/2.7.2/xml/features
mvn:org.apache.activemq/activemq-karaf/5.6.0/xml/features
mvn:ddf.catalog.kml/catalog-kml-app/2.1.0/xml/features
mvn:ddf.mime.tika/mime-tika-app/1.0.0/xml/features
```

If for any reason, any of the lists above do not match the list of features, repositories, or configurations

found in the DDF system consoles, the following command can be executed:

```
cluster:sync
```

This command should allow for a DDF node to be synchronized with the rest of the cluster.

Check for Active Nodes

Checking whether a node is active or not can be done utilizing the node ping command. In order to use this command you must have access to one of the the DDF command line shells. A list of nodes can be shown by executing the following command:

```
cluster:node-list
```

The command should show the following output:

```
ID Host Name Port
* [192.168.1.110:5701 ] [192.168.1.110 ] [ 5701]
```

The output will show the ID, host name, and port of each active DDF node in the cluster. The asterisk shows which node you are currently accessing the shell on. Now that you have a listing of node IDs, you can use these to ping other nodes. Execute the following command:

```
cluster:node-ping
```

The following result will print out until you press ctrl-c:

```
PING 192.168.1.110:5701
from 1: req=192.168.1.110:5701 time=9 ms
from 2: req=192.168.1.110:5701 time=4 ms
from 3: req=192.168.1.110:5701 time=2 ms
from 4: req=192.168.1.110:5701 time=3 ms
from 5: req=192.168.1.110:5701 time=4 ms
from 6: req=192.168.1.110:5701 time=2 ms
from 7: req=192.168.1.110:5701 time=2 ms
^C
```

The output will provide you with a typical ping result showing connectivity and response times.

Configuring Catalog Provider

This scenario describes how to reconfigure DDF to use a different catalog provider. This scenario assumes DDF is already running.

NOTE

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     [ddf.catalog.source.CatalogProvider]                    ddf.providers.provider-dummy
(175)
          osgi.service.blueprint.compname DummyProvider
```

Configuring Notifications

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

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:8993/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?transform=resource&user=1w1qlo79j6tscii19jszwp9s2i55 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. . Set a properties map containing entries for each of the parameters listed above in the Usage section.

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

+ . Post the notification to the OSGi event defined in the previous step.

Example for Publishing Notification

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

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

eventAdmin.postEvent(event);
```

Configuring Thread Pools

The `system.properties` file found under `$DDF_HOME/etc` contains properties that will be made available through system properties at the beginning of Karaf's boot process. The `org.codice.ddf.system.threadPoolSize` property can be used to specify the size of thread pools used by: * Federating requests between DDF systems * Downloading resources * Handling asynchronous queries, such as queries from the UI

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

Configuring Global System Properties

The `system.properties` file found under `$DDF_HOME/etc` contains properties that will be made available through the system properties on startup. After changing any of these properties you will need to restart the system. If you change the 'hostname' property you will also need to configure the certs as

described in Configuring DDF with New Certificates. The system by default uses both http and https so both httpsPort and httpPort need to be specified. The protocol and port properties are the defaults the system should use in places where either http or https could be valid.

```
org.codice.ddf.system.protocol=https://        #one of http:// or https://
org.codice.ddf.system.hostname=localhost       #should be the fully qualified domain name
org.codice.ddf.system.httpsPort=8993           #secure port
org.codice.ddf.system.httpPort=8181            #public port
org.codice.ddf.system.port=8993                #default port corresponding to the
protocol selected. Should match the httpPort or httpsPort
org.codice.ddf.system.rootContext=/services    #the root context that services will be
made available under

org.codice.ddf.system.siteName=ddf.distribution        #the name of this instance
org.codice.ddf.system.siteContact=                        #contact for this instance
org.codice.ddf.system.version=2.8.2          #this instances version
org.codice.ddf.system.organization=Codice Foundation   #who owns/runs this instance
```

The above properties (along with any other system properties) are available to be used as variable parameters in input url fields within the admin UI. For example if you wanted to enter the url for the csw service you could write

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

instead of

```
https://localhost:8993/services/csw
```

The variable version is longer but will not need to be changed if the system host, port or root context changes.

Managing Web Service Security

Configuring WSS

- Add system console to whitelisted contexts
  - https://localhost:8993/admin
    - Navigate to the DDF Security

Navigate to Web Context Policy Manager

- Add /system/console to the Whitelisted Contexts

NOTE

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

Enable Backup Plugin: true

Assumes a hostname of 'ddf'

IMPORTANT

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

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

A detailed view is available of the properties to update.

- Configure Catalog External Solr Catalog Provider
  - Change the HTTP URL to: `https://ddf:8993/solr`
- Configure Persistent Store
  - Solr URL: `https://ddf:8993/solr`
- Configure Catalog Federation Strategy
  - Change Solr URL to: `https://ddf:8993/solr`
- Configure Security STS Client
  - Change the STS WSDL Address to: `https://ddf:8993/services/SecurityTokenService?wsdl`
- Configure Security STS Server
  - SAML Assertion Lifetime: `86400`
  - Change Token Issuer to: `ddf`
  - Change Signature Username to: `ddf`
  - Change Encryption Username to: `ddf`
- Configure Security STS LDAP Login
  - `features:install security-sts-ldaplogin`

- LDAP URL: `ldaps://ddf:1636`
- SSL Keystore Alias: `ddf`
- Configure Platform Global Configuration
- Protocol: `https`
- Host: `ddf`
- Port: `8993`

- Configure the Web Context Policy Manager
  - In White Listed Contexts, add /sso
- Install and Configure the Embedded LDAP

IMPORTANT

The Embedded LDAP is not recommended for production use. It is only recommended to be used for development purposes or extremely small server loads.

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

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

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

- The default password in `config.ldif` for `serverKeystore.jks` is `changeit`. This needs to be modified to

password.

  - `ds-cfg-key-store-file: ../../keystores/serverKeystore.jks`
  - `ds-cfg-key-store-type: JKS`
  - `ds-cfg-key-store-pin: password`
  - `cn: JKS`

- The default password in `config.ldif` for `serverTruststore.jks` is `changeit`. This needs to be modified to password.
  - `ds-cfg-trust-store-file: ../../keystores/serverTruststore.jks`
  - `ds-cfg-trust-store-pin: password`
  - `cn: JKS`
- If using the default keystores and certificates, start the `opendj-embedded app:start opendj-embedded`
- Shutdown DDF
  - `<ddf-home>/bin/shutdown -f`
- Add the newly created keystore and truststore
  - put the newly created `serverKeystore.jks` in `<ddf-home>/etc/keystores`
  - put the newly created `serverTruststore.jks` in `<ddf-home>/etc/keystores`
- Configure system properties
  - In `<ddf-home>/etc/system.properties`, modify the keystore and truststore paths and passwords as appropriate.

system.properties

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

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

- Configure users properties
  - In `<ddf-home>/etc/users.properties`, change `localhost=localhost,admin` to `ddf=ddf,admin`
- Start ddf
  - For Windows, run `<ddf-home>/bin/ddf.bat`
  - For *Nix, run `<ddf-home>/bin/ddf`
  - Admin Console: `https://ddf:8993/admin`
  - Search UI: https://ddf:8993/search
- Remove system/console from whitelist
  - `https://localhost:8993/admin`
    - Navigate to DDF Security
    - Navigate to the Web Context Policy Manager
    - Remove `/system/console` from the Whitelisted Contexts

Auditing

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

CAS (SSO) Authentication

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

Username and Password

Sample – Successful login

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

Sample – Failed login

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

PKI Certificate

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

Sample – Successful login

```
2013-04-24 15:13:14,388 INFO
[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthentication
Handler] - <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.X509CredentialsAuthentication
Handler 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.X509CredentialsAuthentication
Handler@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.X509CredentialsAuthentication
Handler] - <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.X509CredentialsAuthentication
Handler] - <.* 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.X509CredentialsAuthentication
Handler] - <CN=Hogwarts Certifying Authority.+ matches EMAILADDRESS=goss-
support@owfgoss.org, CN=localhost, OU=Ozone, O=Ozone, L=Columbia, ST=Maryland, C=US ==
false>
2013-04-25 14:15:47,478 DEBUG
[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthentication
Handler] - <Found valid client certificate>
2013-04-25 14:15:47,478 INFO
[org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthentication
Handler] - <Failed to authenticate
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc>
2013-04-25 14:15:47,478 INFO [org.jasig.cas.authentication.AuthenticationManagerImpl] -
<org.jasig.cas.adaptors.x509.authentication.handler.support.X509CredentialsAuthentication
Handler failed to authenticate
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc>
2013-04-25 14:15:47,478 INFO
[com.github.inspektr.audit.support.Slf4jLoggingAuditTrailManager] - <Audit trail record
BEGIN
=============================================================
WHO:
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc
WHAT: supplied credentials:
org.jasig.cas.adaptors.x509.authentication.principal.X509CertificateCredentials@1795f1cc
ACTION: AUTHENTICATION_FAILED
APPLICATION: CAS
WHEN: Thu Apr 25 14:15:47 MST 2013
CLIENT IP ADDRESS: 127.0.0.1
SERVER IP ADDRESS: 127.0.0.1
=============================================================
>
```

STS Authentication

Username and Password

Sample – Successful login

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

Sample – Failed login

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

PKI Certificate

Sample – Successful login

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

Sample – Failed login

```
[WARN ] 2014-07-17 15:05:46,061 | qtp1401560510-75 | securityLogger  |  Security Token
Service REQUEST
STATUS: FAILURE
OPERATION: Issue
URL: N.A.
TOKENTYPE: N.A.
APPLIESTO: <null>
EXCEPTION: org.apache.cxf.ws.security.sts.provider.STSException: 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-
wssecurity-utility-
1.0.xsd">U1QtMTctQmw0aGRrS05jaTV3cE82Zm11VE0tY2FzfGh0dHBzOi8vdG9rZW5pc3N1ZXI6ODk5My9zZXJ2
aWNlcy9TZWN1cml0eVRva2VuU2VydmljZQ==</BinarySecurityToken>
 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-
wssecurity-utility-
1.0.xsd">U1QtMjctOU43RUlkNHkzVFoxQmZCb0RIdkItY2Fz</BinarySecurityToken>
10:54:22,119 | INFO  | qtp995500086-141 | securityLogger                    |
rity.common.audit.SecurityLogger  143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Validating ticket [ST-27-9N7EId4y3TZ1BfBoDHvB-cas] for service
[https://server:8993/services/SecurityTokenService]. Request IP: 127.0.0.1, Port: 64548
10:54:22,169 | INFO  | qtp995500086-141 | securityLogger                    |
rity.common.audit.SecurityLogger  143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Unable to validate CAS token. Request IP: 127.0.0.1, Port: 64548
10:54:22,244 | INFO  | qtp995500086-618 | securityLogger                    |
rity.common.audit.SecurityLogger  143 | 245 - security-core-commons - 2.2.0.ALPHA5-
SNAPSHOT | Error requesting the security token from STS at:
https://server:8993/services/SecurityTokenService.
```

AuditPlugin

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

Configuring Audit Plugin

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

1. Select DDF Catalog
2. Select *Features* tab
3. Install both `catalog-security-logging` and `catalog-security-audit-plugin` features.....
   .....

Web Context Policy Manager

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

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

Configuring Web Context Policy Manager

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

For example, installing the `security-sts-ldaplogin` feature adds an ldap realm. Contexts can then be pointed to ldap realm for authentication and STS will be instructed to authenticate them against ldap. As you add REST endpoints, you may need to add different types of authentication through the Web Context Policy Manager.

Table 1. Default Types of Authentication

saml	**Activates single-sign on (SSO) across all REST endpoints that use.**
basic	Activates basic authentication.
PKI	Activates public key infrastructure authentication
anon(anonymous)	provides anonymous access

CAS SSO Configuration

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

General Server Setup and Configuration

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

Install using DDF CAS WAR

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

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

1. Download and unzip Tomcat Distribution [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

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

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

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

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

4. Start Tomcat.

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

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

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

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

uncomment the following in server.xml:

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

then comment out:

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

5. Deploy the DDF CAS WAR to Tomcat.

   a. Obtain the CAS WAR by building it from cas-distribution.

   b. Copy it into the webapps folder on Tomcat:

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

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

log and look for lines similar to the following:

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

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

Configure an Existing CAS Installation

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

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

Configure CAS for DDF

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

CAS Client

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

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

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

CAS Token Validator

The 'CAS Server URL' configuration should point to the hostname of the CAS server and should match

the SSL certificate that it is using.

Additional Configuration

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

Example Workflow

The following is a sample workflow hat 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.

Configuring CAS for LDAP

Install and Configure LDAP

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

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

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

Copy `thirdparty/cas-server-support-ldap-3.3.1/target/cas-server-support-x509-3.3.1_1.jar` to `{ozone-widget-framework}/apache-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-tomcat-{version}/webapps/cas/WEB-INF/deployerConfigContext.xml, add the contextSource bean stanza to the beans stanza:

deployerConfigContext.xml

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

Configure Ozone

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

1. Modify OWFsecurityContext.xml

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

   b. In {ozone-widget-framework}/apache-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.DefaultLdapAuthoritiesPopulator">
    <constructor-arg ref="contextSource"/>
    <!-- search base for determining what roles a user has -->
    <constructor-arg value="ou=roles"/>
</bean>
```

   c. In {ozone-widget-framework}/apache-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.FilterBasedLdapUserSearch">
    <!-- search base for finding User records -->
    <constructor-arg value="ou=users" />
    <constructor-arg value="(uid={0})" /> <!-- filter applied to entities under the
search base in order to find a given user.
                                              this default searches for an entity
with a matching uid -->
    <constructor-arg ref="contextSource" />
</bean>
```

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

3. Modify OWFCASBeans.xml

   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.CasAuthenticationProvider">
    <property name="userDetailsService" ref="ldapUserService" />
    <property name="serviceProperties" ref="serviceProperties" />
    <property name="ticketValidator" ref="ticketValidator" />
    <property name="key" value="an_id_for_this_auth_provider_only" />
</bean>
```

Configuring CAS for X509 User Certificates

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

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

Copy `thirdparty/cas-server-support-x509-3.3.1/target/cas-server-support-x509-3.3.1.jar` to `apache-tomecat-{version}/webapps/cas/WEB-INF/lib/cas-server-support-x509-3.3.1.jar`.

Configure Web Flow

1. In apache-tomcat-{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:

a. Define the x509Check bean.

x509Check

```
<bean
   id="x509Check"
   p:centralAuthenticationService-ref="centralAuthenticationService"
   class="org.jasig.cas.adaptors.x509.web.flow.X509CertificateCredentialsNonInteract
iveAction" >
   <property name="centralAuthenticationService" ref="centralAuthenticationService
"/>
</bean>
```

Configure the Authentication Handler

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

1. In the `list` stanza of the property stanza with name `authenticationHandlers` of the bean stanza with id `authenticationManager`, add the `X509CredentialAuthenticationHander` bean definition.

   X509CredentialAuthenticationHander

```
<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.X509Credenti
alsAuthenticationHandler">
         <property name="trustedIssuerDnPattern" value=".*" />
         <!--
           <property name="maxPathLength" value="3" />
           <property name="checkKeyUsage" value="true" />
           <property name="requireKeyUsage" value="true" />
         -->
       </bean>
    </list>
  </property>
</bean>
```

Configure the Credentials to the Principal Resolver

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

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

X509CertificateCredentialsToIdentifierPrincipalResolver

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

      <!-- Other bean definitions -->

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

</bean>
```

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

Resolver Class	**Identifier Output**
X509CertificateCredentialsToDistinguishedName PrincipalResolver	Retrieve the complete distinguished name and use that as the identifier.
X509CertificateCredentialsToIdentifierPrincipal Resolver	Transform some subset of the identifier into the ID for the principal.
X509CertificateCredentialsToSerialNumberPrinc ipalResolver	Use the unique serial number of the certificate.
X509CertificateCredentialsToSerialNumberAndIs suerDNPrincipalResolver	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

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	File: etc/org.ops4j.pax.web.cfg File: etc/ws-security/server/encryption.properties File: etc/ws-security/server/signature.properties File: etc/ws-security/issuer/encryption.properties File: etc/ws-security/issuer/signature.properties File: etc/system.properties	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.

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 if 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`

Configuration Management

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

Certificate Revocation Configuration

Enable Revocation

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

1. Place the CRL in `<ddf.home>/etc/keystores`.
2. 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/crlTokenissuerValid.pem`

Uncommenting this property will also enable CRL revocation for any context policy implementing PKI authentication. For example, adding an authentication policy in the Web Context Policy Manager of "/search=PKI|ANON" will disable basic authentication, and require a certificate for the search UI. If a certificate is not in the CRL, it will be allowed through, otherwise it will get a 401 error. Not providing a cert will pass it to the anonymous handler and the user will be granted anonymous access.

Disable Revocation

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

1. Comment out the following line in `<ddf.home>/etc/ws-security/server/encryption.properties`.
   `#org.apache.ws.security.crypto.merlin.x509crl.file=etc/keystores/crlTokenissuerValid.pem`

The PKIHandler will not check the CRL if this property is not defined.

Add Revocation to a Web Context

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

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

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

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

Add Revocation to a New Endpoint

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

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:838)[161:org.apac
he.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.validate.SignatureTrustValidator.verifyTrustInCert(SignatureTrustV
alidator.java:213)[161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.validate.SignatureTrustValidator.validate(SignatureTrustValidator.
java:72)[161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.validate.SamlAssertionValidator.verifySignedAssertion(SamlAssertio
nValidator.java:121)[161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.validate.SamlAssertionValidator.validate(SamlAssertionValidator.ja
va:100)[161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.processor.SAMLTokenProcessor.handleSAMLToken(SAMLTokenProcessor.ja
va:188)[161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.processor.SAMLTokenProcessor.handleToken(SAMLTokenProcessor.java:7
8)[161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.ws.security.WSSecurityEngine.processSecurityHeader(WSSecurityEngine.java:396)[
161:org.apache.ws.security.wss4j:1.6.9]
    at
org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor.handleMessage(WSS4JInInterceptor.java
:274)[164:org.apache.cxf.cxf-rt-ws-security:2.7.3]
    at
org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor.handleMessage(WSS4JInInterceptor.java
:93)[164:org.apache.cxf.cxf-rt-ws-security:2.7.3]
    at
org.apache.cxf.phase.PhaseInterceptorChain.doIntercept(PhaseInterceptorChain.java:271)[12
3:org.apache.cxf.cxf-api:2.7.3]
    at
org.apache.cxf.transport.ChainInitiationObserver.onMessage(ChainInitiationObserver.java:1
21)[123:org.apache.cxf.cxf-api:2.7.3]
    at
org.apache.cxf.transport.http.AbstractHTTPDestination.invoke(AbstractHTTPDestination.java
:239)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
org.apache.cxf.transport.servlet.ServletController.invokeDestination(ServletController.ja
va:218)[130:org.apache.cxf.cxf-rt-transports-http:2.7.3]
    at
```

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

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

```
sun.security.provider.certpath.PKIXMasterCertPathValidator.validate(PKIXMasterCertPathVal
idator.java:139)[:1.6.0_33]
    at
sun.security.provider.certpath.PKIXCertPathValidator.doValidate(PKIXCertPathValidator.jav
a:330)[:1.6.0_33]
    at
sun.security.provider.certpath.PKIXCertPathValidator.engineValidate(PKIXCertPathValidator
.java:178)[:1.6.0_33]
    at
java.security.cert.CertPathValidator.validate(CertPathValidator.java:250)[:1.6.0_33]
    at
org.apache.ws.security.components.crypto.Merlin.verifyTrust(Merlin.java:814)[161:org.apac
he.ws.security.wss4j:1.6.9]
    ... 45 more
```

Encryption Service

Encryption Command

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

Below is an example of the security:encrypt command used to encrypt the plain text "myPasswordToEncrypt". The output, `bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=`, is the encrypted value.

```
ddf@local>security:encrypt myPasswordToEncrypt

bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

Redaction and Filtering

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

How Redaction and Filtering Works

Each metacard 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 metacard is then processed by a filter/redaction plugin that creates a KeyValueCollectionPermission from the metacard's security attribute. This permission is then checked against the user subject to determine if the subject has the correct claims to view that metacard. The decision to filter/redact* the metacard 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 metacard will either be filtered/redacted or allowed to pass through.

NOTE **The default setting is to redact records.**

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

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

```
<metacard>
    <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>
</metacard>
```

```
<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 metacard record. In this particular case, the metacard 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 https://localhost:8993/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 metacard 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.

NOTE	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 Metacard 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 Metacard

To enable redaction/filtering on a new type of record, implement a PostQueryPlugin that is able to read the string metadata contained within the metacard record. The plugin must set the security attribute to a map of list of values extracted from the metacard. Note that in DDF, there is no default plugin that populates the security attribute on the metacard. 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"?>
<metacard xmlns="urn:catalog:metacard" xmlns:ns2="http://www.opengis.net/gml" xmlns:ns3=
"http://www.w3.org/1999/xlink" xmlns:ns4="http://www.w3.org/2001/SMIL20/" xmlns:ns5=
"http://www.w3.org/2001/SMIL20/Language" ns2:id="99f494b22f4341e9a3ba3ef6a5fe8734">
    <type>ddf.metacard</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>
</metacard>
```

Security Token Service

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 ingests 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.

- 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 moreclaims 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.ClaimsHander 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

NOTE This XML file is found inside of the STS bundle and is named ws-trust-1.4-service.wsdl.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:tns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
xmlns:wstrust="http://docs.oasis-open.org/ws-sx/ws-trust/200512/" xmlns:wsdl=
"http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsap10="http://www.w3.org/2006/05/addressing/wsdl" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp=
"http://www.w3.org/ns/ws-policy" xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsam=
"http://www.w3.org/2007/05/addressing/metadata" targetNamespace="http://docs.oasis-
open.org/ws-sx/ws-trust/200512/">
    <wsdl:types>
        <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/CancelFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
        <wsdl:operation name="Issue">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTRC/IssueFinal" message="tns:RequestSecurityTokenResponseCollectionMsg"/>
        </wsdl:operation>
        <wsdl:operation name="Renew">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Renew" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/RenewFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
        <wsdl:operation name="Validate">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Validate" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/ValidateFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
        <wsdl:operation name="KeyExchangeToken">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
```

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

```
        </wsdl:operation>
        <wsdl:operation name="Renew">
            <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Renew"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="KeyExchangeToken">
            <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/KeyExchangeToken"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="RequestCollection">
            <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/RequestCollection"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsp:Policy wsu:Id="STS_policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <wsap10:UsingAddressing/>
                <wsp:ExactlyOne>
                    <sp:TransportBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                        <wsp:Policy>
                            <sp:TransportToken>
                                <wsp:Policy>
                                    <sp:HttpsToken>
                                        <wsp:Policy/>
                                    </sp:HttpsToken>
                                </wsp:Policy>
                            </sp:TransportToken>
                            <sp:AlgorithmSuite>
```

```
                                <wsp:Policy>
                                    <sp:Basic128/>
                                </wsp:Policy>
                            </sp:AlgorithmSuite>
                            <sp:Layout>
                                <wsp:Policy>
                                    <sp:Lax/>
                                </wsp:Policy>
                            </sp:Layout>
                            <sp:IncludeTimestamp/>
                        </wsp:Policy>
                    </sp:TransportBinding>
                </wsp:ExactlyOne>
                <sp:Wss11 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                    <wsp:Policy>
                        <sp:MustSupportRefKeyIdentifier/>
                        <sp:MustSupportRefIssuerSerial/>
                        <sp:MustSupportRefThumbprint/>
                        <sp:MustSupportRefEncryptedKey/>
                    </wsp:Policy>
                </sp:Wss11>
                <sp:Trust13 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                    <wsp:Policy>
                        <sp:MustSupportIssuedTokens/>
                        <sp:RequireClientEntropy/>
                        <sp:RequireServerEntropy/>
                    </wsp:Policy>
                </sp:Trust13>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsp:Policy wsu:Id="Input_policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <sp:SignedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                    <sp:Body/>
                    <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
"/>
                    <sp:Header Name="From" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="FaultTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="ReplyTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="MessageID" Namespace=
```

```
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="RelatesTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="Action" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                </sp:SignedParts>
                <sp:EncryptedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                    <sp:Body/>
                </sp:EncryptedParts>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsp:Policy wsu:Id="Output_policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <sp:SignedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                    <sp:Body/>
                    <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing
"/>
                    <sp:Header Name="From" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="FaultTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="ReplyTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="MessageID" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="RelatesTo" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                    <sp:Header Name="Action" Namespace=
"http://www.w3.org/2005/08/addressing"/>
                </sp:SignedParts>
                <sp:EncryptedParts xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                    <sp:Body/>
                </sp:EncryptedParts>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>
    <wsdl:service name="SecurityTokenService">
        <wsdl:port name="STS_Port" binding="tns:STS_Binding">
            <soap:address location="https://localhost:8993/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.

BinarySecurityToken (CAS) SAML Security Token Request/Response

BinarySecurityToken (CAS) Sample Request/Response

Request

Sample Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RST/Issue</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:60652909-faca-
4e4a-a4a7-8a5ce243a7cb</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>https://server:8993/services/SecurityTokenService</To>
        <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
            <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
        </ReplyTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-1">
                <wsu:Created>2013-04-29T18:35:10.688Z</wsu:Created>
                <wsu:Expires>2013-04-29T18:40:10.688Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512">
            <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
                    <wsa:Address>https://server:8993/services/SecurityTokenService</wsa:A
ddress>
                </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"
>U1QtMTQtYUtmcDYxcFRtS0FxZG1pVDMzOWMtY2FzfGh0dHBzOi8vdG9rZW5pc3N1ZXI6ODk5My9zZXJ2aWNlcy9T
ZWN1cml0eVRva2VuU2VydmljZQ==</BinarySecurityToken>
            </wst:OnBehalfOf>
            <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</wst:TokenType>
            <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/PublicKey</wst:KeyType>
            <wst:UseKey>
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data>
                        <ds:X509Certificate>
MIIC5DCCAk2gAwIBAgIJAKj7ROPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUzEQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZHllYXIxGDAWBgNVBAoMD0xvY2toZWVkIE1h
cnRpbjENMAsGA1UECwwESTRDRTEPMA0GA1UEAwwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGNl
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTAlVT
MRAwDgYDVQQIDAdBcml6b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2hlZWQg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbGllbnQxHDAaBgkqhkiG9w0BCQEWDWk0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVCnEmHndkf9bWUlAOfNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnhlNW9+lxBDAfBgNVHSMEGDAWgBQx11VLtYXL
vFGpFdHnhlNW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4GBAHYs2OI0K6yVXzyS
sKcv2fmfw6XCICGTnyA7BOdAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcDTR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/O5tywXRT1+freI3bwAN0
L6tQ
</ds:X509Certificate>
                    </ds:X509Data>
                </ds:KeyInfo>
            </wst:UseKey>
            <wst:Renewing/>
        </wst:RequestSecurityToken>
    </soap:Body>
</soap:Envelope>
```

Response

Sample Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:7a6fde04-9013-
41ef-b08b-0689ffa9c93e</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>http://www.w3.org/2005/08/addressing/anonymous</To>
        <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:60652909-faca-
4e4a-a4a7-8a5ce243a7cb</RelatesTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-2">
                <wsu:Created>2013-04-29T18:35:11.459Z</wsu:Created>
                <wsu:Expires>2013-04-29T18:40:11.459Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/ws-
sx/ws-trust/200512" xmlns:ns2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns4="http://www.w3.org/2005/08/addressing"
xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200802">
            <RequestSecurityTokenResponse>
                <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</TokenType>
                <RequestedSecurityToken>
                    <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ID="_BDC44EB8593F47D1B213672605113671" IssueInstant="2013-04-29T18:35:11.370Z"
Version="2.0" xsi:type="saml2:AssertionType">
                        <saml2:Issuer>tokenissuer</saml2:Issuer>
                        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                            <ds:SignedInfo>
                                <ds:CanonicalizationMethod Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                <ds:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                                <ds:Reference URI="#_BDC44EB8593F47D1B213672605113671">
                                    <ds:Transforms>
                                        <ds:Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                        <ds:Transform Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#">
```

```
                                        <ec:InclusiveNamespaces xmlns:ec=
"http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs"/>
                                    </ds:Transform>
                                </ds:Transforms>
                                <ds:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
                                <ds:DigestValue>6wnWbft6Pz5XOF5Q9AG59gcGwLY=</ds:Dige
stValue>
                            </ds:Reference>
                        </ds:SignedInfo>
                        <ds:SignatureValue>h+NvkgXGdQtca3/eKebhAKgG38tHp3i2n5uLLy8xXX
Ig02qyKgEP0FCowp2LiYlsQU9YjKfSwCUbH3WR6jhbAv9zj29CE+ePfEny7MeXvgNl3wId+vcHqti/DGGhhgtO2Mb
x/tyX1BhHQUwKRlcHajxHeecwmvV7D85NMdV48tI=</ds:SignatureValue>
                        <ds:KeyInfo>
                            <ds:X509Data>
                                <ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9
w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxMVoXDTIz
MDQwNzE4MzcxMVowgaYxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdBcml6b25hMREwDwYDVQQHEwhH
b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtxtN9
uB44diiIqq3JOzDGfDhGLu6mjpuHO1hrKItv42hBOhhmH7lS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lfi7IvsVVsNrtAEQljOMmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqpV8Kb0COOGwIDAQAB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2Vy
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgNVHSMEgZ8wgZyAFBcn
en6/j05DzaVwORwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk7OcwO7gwwDQYJKoZIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZlO6VrAxRQ+eFeN6NyuyhgYy5K6l/sIWdaGou5iJOQx
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>MIIC5DCCAk2gAwIBAgIJAKj7R
OPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUzEQ
```

```
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZHllYXIxGDAWBgNVBAoMD0xvY2toZWVkIE1h
cnRpbjENMAsGA1UECwwESTRDRTEPMA0GA1UEAwwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGNl
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTAlVT
MRAwDgYDVQQIDAdBcml6b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2hlZWQg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbGllbnQxHDAaBgkqhkiG9w0BCQEWDWk0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVCnEmHndkf9bWUlAOfNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnhlNW9+lxBDAfBgNVHSMEGDAWgBQx11VLtYXL
vFGpFdHnhlNW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4GBAHYs2OI0K6yVXzyS
sKcv2fmfw6XCICGTnyA7BOdAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcDTR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/O5tywXRT1+freI3bwAN0
L6tQ</ds:X509Certificate>
                                    </ds:X509Data>
                                </ds:KeyInfo>
                            </saml2:SubjectConfirmationData>
                        </saml2:SubjectConfirmation>
                    </saml2:Subject>
                    <saml2:Conditions NotBefore="2013-04-29T18:35:11.407Z"
NotOnOrAfter="2013-04-29T19:05:11.407Z">
                        <saml2:AudienceRestriction>
                            <saml2:Audience>https://server:8993/services/SecurityToke
nService</saml2:Audience>
                        </saml2:AudienceRestriction>
                    </saml2:Conditions>
                    <saml2:AuthnStatement AuthnInstant="2013-04-29T18:35:11.392Z">
                        <saml2:AuthnContext>
                            <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:a
c:classes:unspecified</saml2:AuthnContextClassRef>
                        </saml2:AuthnContext>
                    </saml2:AuthnStatement>
                    <saml2:AttributeStatement>
                        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                            <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
                        </saml2:Attribute>
                        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                            <saml2:AttributeValue xsi:type="xs:string"
>srogers@example.com</saml2:AttributeValue>
                        </saml2:Attribute>
                        <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                            <saml2:AttributeValue xsi:type="xs:string">
```

```
srogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
                            </saml2:Attribute>
                        </saml2:AttributeStatement>
                    </saml2:Assertion>
                </RequestedSecurityToken>
                <RequestedAttachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedAttachedReference>
                <RequestedUnattachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedUnattachedReference>
                <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                    <wsa:EndpointReference xmlns:wsa=
"http://www.w3.org/2005/08/addressing">
                        <wsa:Address>https://server:8993/services/SecurityTokenService</w
sa:Address>
                    </wsa:EndpointReference>
```

```
                </wsp:AppliesTo>
                <Lifetime>
                    <ns2:Created>2013-04-29T18:35:11.444Z</ns2:Created>
                    <ns2:Expires>2013-04-29T19:05:11.444Z</ns2:Expires>
                </Lifetime>
            </RequestSecurityTokenResponse>
        </RequestSecurityTokenResponseCollection>
    </soap:Body>
</soap:Envelope>
```

UsernameToken Bearer SAML Security Token Request/Response

To obtain a SAML assertion to use in secure communication to DDF, a RequestSecurityToken (RST) request has to be made to the STS.

A Bearer SAML assertion is automatically trusted by the endpoint. The client doesn't have to prove it can own that SAML assertion. It is the simplest way to request a SAML assertion, but many endpoints won't accept a KeyType of Bearer.

Request

Explanation

- WS-Addressing header with Action, To, and Message ID
- Valid, non-expired timestamp
- Username Token containing a username and password that the STS will authenticate
- Issued over HTTPS
- KeyType of http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer
- Claims (optional): Some endpoints may require that the SAML assertion include attributes of the user, such as an authenticated user's role, name identifier, email address, etc. If the SAML assertion needs those attributes, the RequestSecurityToken must specify which ones to include.

Sample Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-1">
                <wsu:Created>2013-04-29T17:47:37.817Z</wsu:Created>
                <wsu:Expires>2013-04-29T17:57:37.817Z</wsu:Expires>
            </wsu:Timestamp>
            <wsse:UsernameToken wsu:Id="UsernameToken-1">
                <wsse:Username>srogers</wsse:Username>
                <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">password1</wsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
        <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue</wsa:Action>
        <wsa:MessageID>uuid:a1bba87b-0f00-46cc-975f-001391658cbe</wsa:MessageID>
        <wsa:To>https://server:8993/services/SecurityTokenService</wsa:To>
    </soap:Header>
    <soap:Body>
        <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512">
            <wst:SecondaryParameters>
                <t:TokenType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</t:TokenType>
                <t:KeyType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer</t:KeyType>
                <t:Claims xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512" Dialect=
"http://schemas.xmlsoap.org/ws/2005/05/identity">
                    <!--Add any additional claims you want to grab for the service-->
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/uid"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
                </t:Claims>
            </wst:SecondaryParameters>
```

```
            <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
                    <wsa:Address>https://server:8993/services/QueryService</wsa:Address>
                </wsa:EndpointReference>
            </wsp:AppliesTo>
            <wst:Renewing/>
        </wst:RequestSecurityToken>
    </soap:Body>
</soap:Envelope>
```

Response

This is the response from the STS containing the SAML assertion to be used in subsequent requests to QCRUD endpoints:

The `saml2:Assertion` block contains the entire SAML assertion.

The `Signature` block contains a signature from the STS's private key. The endpoint receiving the SAML assertion will verify that it trusts the signer and ensure that the message wasn't tampered with.

The `AttributeStatement` block contains all the Claims requested.

The `Lifetime` block indicates the valid time interval in which the SAML assertion can be used.

Sample Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:eee4c6ef-ac10-
4cbc-a53c-13d960e3b6e8</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>http://www.w3.org/2005/08/addressing/anonymous</To>
        <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:a1bba87b-0f00-46cc-
975f-001391658cbe</RelatesTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-2">
                <wsu:Created>2013-04-29T17:49:12.624Z</wsu:Created>
                <wsu:Expires>2013-04-29T17:54:12.624Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/ws-
sx/ws-trust/200512" xmlns:ns2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns4="http://www.w3.org/2005/08/addressing"
xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200802">
            <RequestSecurityTokenResponse>
                <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</TokenType>
                <RequestedSecurityToken>
                    <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ID="_7437C1A55F19AFF22113672577526132" IssueInstant="2013-04-29T17:49:12.613Z"
Version="2.0" xsi:type="saml2:AssertionType">
                        <saml2:Issuer>tokenissuer</saml2:Issuer>
                        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                            <ds:SignedInfo>
                                <ds:CanonicalizationMethod Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                <ds:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                                <ds:Reference URI="#_7437C1A55F19AFF22113672577526132">
                                    <ds:Transforms>
                                        <ds:Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                        <ds:Transform Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#">
```

```
                                        <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:Dige
stValue>
                            </ds:Reference>
                        </ds:SignedInfo>
                        <ds:SignatureValue>X5Kzd54PrKIlGVV2XxzCmWFRzHRoybF7hU6zxbEhSL
MR0AWS9R7Me3epq91XqeOwvIDDbwmE/oJNC7vI0fIw/rqXkx4aZsY5a5nbAs7f+aXF9TGdk82x2eNhNGYpViq0YZJ
fsJ5WSyMtG8w5nRekmDMy9oTLsHG+Y/OhJDEwq58=</ds:SignatureValue>
                        <ds:KeyInfo>
                            <ds:X509Data>
                                <ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9
w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxMVoXDTIz
MDQwNzE4MzcxMVowgaYxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdBcml6b25hMREwDwYDVQQHEwhH
b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtxtN9
uB44diiIqq3JOzDGfDhGLu6mjpuHO1hrKItv42hBOhhmH7lS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lfi7IvsVVsNrtAEQljOMmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqpV8Kb0COOGwIDAQAB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2Vy
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgNVHSMEgZ8wgZyAFBcn
en6/j05DzaVwORwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk7OcwO7gwwDQYJKoZIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZlO6VrAxRQ+eFeN6NyuyhgYy5K6l/sIWdaGou5iJOQx
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: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:a
c:classes:unspecified</saml2:AuthnContextClassRef>
                            </saml2:AuthnContext>
                        </saml2:AuthnStatement>
                        <saml2:AttributeStatement>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string"
>srogers@example.com</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
                            </saml2:Attribute>
                        </saml2:AttributeStatement>
                    </saml2:Assertion>
                </RequestedSecurityToken>
                <RequestedAttachedReference>
```

```
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_7437C1A55F19AFF22113672577526132</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedAttachedReference>
                <RequestedUnattachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_7437C1A55F19AFF22113672577526132</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedUnattachedReference>
                <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                    <wsa:EndpointReference xmlns:wsa=
"http://www.w3.org/2005/08/addressing">
                        <wsa:Address>https://server:8993/services/QueryService</wsa:Addre
ss>
                    </wsa:EndpointReference>
                </wsp:AppliesTo>
                <Lifetime>
                    <ns2:Created>2013-04-29T17:49:12.620Z</ns2:Created>
                    <ns2:Expires>2013-04-29T18:19:12.620Z</ns2:Expires>
                </Lifetime>
            </RequestSecurityTokenResponse>
        </RequestSecurityTokenResponseCollection>
    </soap:Body>
</soap:Envelope>
```

X.509 PublicKey SAML Security Token Request/Response

In order to obtain a SAML assertion to use in secure communication to DDF, a RequestSecurityToken (RST) request has to be made to the STS.

An endpoint's policy will specify the type of security token needed. Most of the endpoints that have been used with DDF require a SAML v2.0 assertion with a required KeyType of http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey. This means that the SAML assertion provided by the client to a DDF endpoint must contain a SubjectConfirmation block with a type of "holder-of-key" containing the client's public key. This is used to prove that the client can possess the SAML assertion returned by the STS.

Request

Explanation The STS that comes with DDF requires the following to be in the RequestSecurityToken request in order to issue a valid SAML assertion. See the request block below for an example of how these components should be populated. * WS-Addressing header containing Action, To, and MessageID blocks * Valid, non-expired timestamp * Issued over HTTPS * TokenType of http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0 * KeyType of http://docs.oasis-open.org/ws-sx/ws-trust/200512/PublicKey * X509 Certificate as the Proof of Possession or POP. This needs to be the certificate of the client that will be both requesting the SAML assertion and using the SAML assertion to issue a query * Claims (optional): Some endpoints may require that the SAML assertion include attributes of the user, such as an authenticated user's role, name identifier, email address, etc. If the SAML assertion needs those attributes, the RequestSecurityToken must specify which ones to include. ** UsernameToken: If Claims are required, the RequestSecurityToken security header must contain a UsernameToken element with a username and password.

Sample Request

```
<soapenv:Envelope xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
      <wsa:MessageID>uuid:527243af-94bd-4b5c-a1d8-024fd7e694c5</wsa:MessageID>
      <wsa:To>https://server:8993/services/SecurityTokenService</wsa:To>
      <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
         <wsu:Timestamp wsu:Id="TS-17">
            <wsu:Created>2014-02-19T17:30:40.771Z</wsu:Created>
            <wsu:Expires>2014-02-19T19:10:40.771Z</wsu:Expires>
         </wsu:Timestamp>

         <!-- OPTIONAL: Only required if the endpoint that the SAML assertion will be
sent to requires claims. -->
         <wsse:UsernameToken wsu:Id="UsernameToken-16">
            <wsse:Username>pparker</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">password1</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary">LCTD+5Y7hlWIP6SpsEg9XA==</wsse:Nonce>
            <wsu:Created>2014-02-19T17:30:37.355Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body>
      <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512">
         <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</wst:TokenType>
         <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/PublicKey</wst:KeyType>

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

```
            <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
         </wst:Claims>
         <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
            <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
            <wsa:Address>https://server:8993/services/QueryService</wsa:Address>
            </wsa:EndpointReference>
         </wsp:AppliesTo>
         <wst:UseKey>
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
               <ds:X509Data>
                  <ds:X509Certificate>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAk
GA1UEBhMCVVMxGDAWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjU0OVoXDTE2MDUwNzAwMjU0OVowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxDzANBgNVBAMTBmNsaWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAOq6L1/jjZ5cyhjhHEbOHr5WQpboKACYbrsn8lg85LGNoAfcwImr9KBmOxGb
ZCxHYIhkW7pJ+kppyH8DDMviIvvdkvrAIU0l8OBRn2wReCBGQ01Imdc3+WzFF2svW75d6wii2ZVd
eMvUO15p/pAD/sdIfXmAfyu8+tqtiO8KVZGkTnlg3AMzfeSrkci5UHMVWj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpdZN2pC85tj9xIoWGh5dFWG1fPcYRgzGPxsybiGOylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAhiWRGyPlY08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNXy43NZLBBlnDjDplNZJoMB0GA1UdDgQWBBRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2NybC5nZHMu
bml0LmRpc2EubWlsL2NybC9ET0RKSVRDQ0FfMjcuY3JsMIGUoIGRoIGOhoGLbGRhcDovL2NybC5n
ZHMubml0LmRpc2EubWlsL2NuJTNkRE9EJTIwSklUQyUyMENBLTI3JTJjb3UlM2RQS0klMmNvdSUz
ZERvRCUyY28lM2RVLlMuJTIwR292ZXJubWVudCUyY2MlM2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN0O2JpbmFyeTAjBgNVHSAEHDAaMAsGCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzAChjFodHRwOi8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVRDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzcC5uc24wLnJjdnMubml0LmRp
c2EubWlsMA0GCSqGSIb3DQEBBQUAA4IBAQCGUJPGh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSgzxnT4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIHr
hwtE3xo6qQRRWlO3/clC4RnTev1crFVJQVBF3yfpRu8udJ2SOGdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCxg1LjC4dmbjKVr9s4QK+/JLNjxD8IkJiZE</ds:X509Certific
ate>
               </ds:X509Data>
            </ds:KeyInfo>
         </wst:UseKey>
      </wst:RequestSecurityToken>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

Explanation This is the response from the STS containing the SAML assertion to be used in subsequent requests to QCRUD endpoints.

The `saml2:Assertion` block contains the entire SAML assertion.

The `Signature` block contains a signature from the STS's private key. The endpoint receiving the SAML assertion will verify that it trusts the signer and ensure that the message wasn't tampered with.

The `SubjectConfirmation` block contains the client's public key, so the server can verify that the client has permission to hold this SAML assertion. The `AttributeStatement` block contains all of the claims requested.

Sample Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/ws-
sx/ws-trust/200512/RSTRC/IssueFinal</Action>
      <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:b46c35ad-3120-
4233-ae07-b9e10c7911f3</MessageID>
      <To xmlns="http://www.w3.org/2005/08/addressing"
>http://www.w3.org/2005/08/addressing/anonymous</To>
      <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:527243af-94bd-4b5c-
a1d8-024fd7e694c5</RelatesTo>
      <wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
         <wsu:Timestamp wsu:Id="TS-90DBA0754E55B4FE7013928310431357">
            <wsu:Created>2014-02-19T17:30:43.135Z</wsu:Created>
            <wsu:Expires>2014-02-19T17:35:43.135Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </soap:Header>
   <soap:Body>
      <ns2:RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/ws-
sx/ws-trust/200802" xmlns:ns2="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" xmlns:ns5="http://www.w3.org/2005/08/addressing">
         <ns2:RequestSecurityTokenResponse>
            <ns2:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</ns2:TokenType>
            <ns2:RequestedSecurityToken>
               <saml2:Assertion ID="_90DBA0754E55B4FE7013928310431176" IssueInstant=
"2014-02-19T17:30:43.117Z" Version="2.0" xsi:type="saml2:AssertionType" xmlns:saml2=
"urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                  <saml2:Issuer>tokenissuer</saml2:Issuer>
                  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                     <ds:SignedInfo>
                        <ds:CanonicalizationMethod Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        <ds:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                        <ds:Reference URI="#_90DBA0754E55B4FE7013928310431176">
                           <ds:Transforms>
                              <ds:Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
```

```
                                 <ec:InclusiveNamespaces PrefixList="xs" xmlns:ec=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
                              </ds:Transform>
                           </ds:Transforms>
                           <ds:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
                           <ds:DigestValue>/bEGqsRGHVJbx298WPmGd8I53zs=</ds:DigestValue>
                        </ds:Reference>
                     </ds:SignedInfo>
                     <ds:SignatureValue>
mYR7w1/dnuh8Z7t9xjCb4XkYQLshj+UuYlGOuTwDYsUPcS2qI0nAgMD1VsDP7y1fDJxeqsq7HYhFKsnqRfebMM4WL
H1D/lJ4rD4UO+i9l3tuiHml7SN24WM1/bOqfDUCoDqmwG8afUJ3r4vmTNPftwOss8BZ/8ODgZzm08ndlkxDfvcN7O
rExbV/3/45JwF/MMPZoqvi2MJGfX56E9fErJNuzezpWnRqPOlWPxyffKMAlVaB9zF6gvVnUqcW2k/Z8X9lN7O5jou
BI281ZnIfsIPuBJERFtYNVDHsIXM1pJnrY6FlKIaOsi55LQu3Ruir/n82pU7BT5aWtxwrn7akBg==
</ds:SignatureValue>
                     <ds:KeyInfo>
                        <ds:X509Data>
                           <ds:X509Certificate>MIIFHTCCBAWgAwIBAgICJe8wDQYJKoZIhvcNAQEFBQ
AwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjYzN1oXDTE2MDUwNzAwMjYzN1owbjELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxFDASBgNVBAMTC3Rva2VuaXNzdWVyMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAx01/U4M1wG+wL1JxX2RL1glj101FkJXMk3KFt3zD//N8x/Dcwwvs
ngCQjXrV6YhbB2V7scHwnThPv3RSwYYiO62z+g6ptfBbKGGBLSZOzLe3fyJR4RxblFKsELFgPHfX
vgUHS/keG5uSRk9S/Okqps/yxKB7+ZlxeFxsIz5QywXvBpMiXtc2zF+M7BsbSIdSx5LcPcDFBwjF
c66rE3/y/25VMht9EZX1QoKr7f8rWD4xgd5J6DYMFWEcmiCz4BDJH9sftw+n1P+CYgrhwslWGqxt
cDME9t6SWR3GLT4Sdtr8ziIM5uUteEhPIV3rVC3/u23JbYEeS8mpnp0bxt5eHQIDAQABo4IB1TCC
AdEwHwYDVR0jBBgwFoAUIxQ0IE1fLjc1ksEGWcOMOmU1kmgwHQYDVR0OBBYEFGBjdkdey+bMHMhC
Z7gwiQ/mJf5VMA4GA1UdDwEB/wQEAwIFoDCB2gYDVR0fBIHSMIHPMDagNKAyhjBodHRwOi8vY3Js
Lmdkcy5uaXQuZGlzYS5taWwvY3JsL0RPREpJVENDQV8yNy5jcmwwgZSggZGggY6GgYtsZGFwOi8v
Y3JsLmdkcy5uaXQuZGlzYS5taWwvY24lM2RET0QlMjBKSVRDJTIwQ0EtMjclMmNvdSUzZFBLSSUy
Y291JTNkRG9EJTJjbyUzZFUuUy4lMjBHb3Zlcm5tZW50JTJjYyUzZFVTP2NlcnRpZmljYXRlcmV2
b2NhdGlvbmxpc3Q7YmluYXJ5MCMGA1UdIAQcMBowCwYJYIZIAWUCAQsFMAsGCWCGSAFlAgELEjB9
BggrBgEFBQcBAQRxMG8wPQYIKwYBBQUHMAKGMWh0dHA6Ly9jcmwuZ2RzLm5pdC5kaXNhLm1pbC9z
aWduL0RPREpJVENDQV8yNy5jZXIwLgYIKwYBBQUHMAGGImh0dHA6Ly9vY3NwLm5zbjAucmN2cy5u
aXQuZGlzYS5taWwwDQYJKoZIhvcNAQEFBQADggEBAIHZQTINU3bMpJ/PkwTYLWPmwCqAYgEUzSYx
bNcVY5MWD8b4XCdw5nM3GnFlOqr4IrHeyyOzsEbIebTe3bv0l1pHx0Uyj059nAhx/AP8DjVtuRU1
/Mp4b6uJ/4yaoMjIGceqBzHqhHIJinG0Y2azua7eM9hVbWZsa912ihbiupCq22mYuHFP7NUNzBvV
j03YUcsy/sES5sRx9Rops/CBN+LUUYOdJOxYWxo8oAbtF8ABE5ATLAwqz4ttsToKPUYh1sxdx5Ef
APeZ+wYDmMu4OfLckwnCKZgkEtJOxXpdIJHY+VmyZtQSB0LkR5toeH/ANV4259Ia5ZT8h2/vIJBg
6B4=</ds:X509Certificate>
                        </ds:X509Data>
                     </ds:KeyInfo>
                  </ds:Signature>
                  <saml2:Subject>
                     <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="http://cxf.apache.org/sts">pparker</saml2:NameID>
```

```
                    <saml2:SubjectConfirmation Method=
"urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
                        <saml2:SubjectConfirmationData xsi:type=
"saml2:KeyInfoConfirmationDataType">
                            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                               <ds:X509Data>
                                  <ds:X509Certificate>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcN
AQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjU0OVoXDTE2MDUwNzAwMjU0OVowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxDzANBgNVBAMTBmNsaWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAOq6L1/jjZ5cyhjhHEbOHr5WQpboKACYbrsn8lg85LGNoAfcwImr9KBmOxGb
ZCxHYIhkW7pJ+kppyH8DDMviIvvdkvrAIU0l8OBRn2wReCBGQ01Imdc3+WzFF2svW75d6wii2ZVd
eMvUO15p/pAD/sdIfXmAfyu8+tqtiO8KVZGkTnlg3AMzfeSrkci5UHMVWj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpdZN2pC85tj9xIoWGh5dFWG1fPcYRgzGPxsybiGOylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAhiWRGyPlY08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNXy43NZLBBlnDjDplNZJoMB0GA1UdDgQWBBRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2NybC5nZHMu
bml0LmRpc2EubWlsL2NybC9ET0RKSVRDQ0FfMjcuY3JsMIGUoIGRoIGOhoGLbGRhcDovL2NybC5n
ZHMubml0LmRpc2EubWlsL2NuJTNkRE9EJTIwSklUQyUyMENBLTI3JTJjb3UlM2RQS0klMmNvdSUz
ZERvRCUyY28lM2RVLlMuJTIwR292ZXJubWVudCUyY2MlM2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN0O2JpbmFyeTAjBgNVHSAEHDAaMAsGCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzAChjFodHRwOi8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVRDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzcC5uc24wLnJjdnMubml0LmRp
c2EubWlsMA0GCSqGSIb3DQEBBQUAA4IBAQCGUJPGh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSgzxnT4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIHr
hwtE3xo6qQRRWlO3/clC4RnTev1crFVJQVBF3yfpRu8udJ2SOGdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCxg1LjC4dmbjKVr9s4QK+/JLNjxD8IkJiZE</ds:X509Certific
ate>
                               </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:classe
s:unspecified</saml2:AuthnContextClassRef>
                    </saml2:AuthnContext>
                </saml2:AuthnStatement>

                <!-- This block will only be included if Claims were requested in the
RST. -->
                <saml2:AttributeStatement>
```

```
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
pparker</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
pparker@example.com</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
pparker</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">Peter
Parker</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
users</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
users</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
                    </saml2:Attribute>
```

```
                  </saml2:AttributeStatement>
               </saml2:Assertion>
            </ns2:RequestedSecurityToken>
            <ns2:RequestedAttachedReference>
               <ns4:SecurityTokenReference wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0" xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
                  <ns4:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-
saml-token-profile-1.1#SAMLID">_90DBA0754E55B4FE7013928310431176</ns4:KeyIdentifier>
               </ns4:SecurityTokenReference>
            </ns2:RequestedAttachedReference>
            <ns2:RequestedUnattachedReference>
               <ns4:SecurityTokenReference wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0" xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
                  <ns4:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-
saml-token-profile-1.1#SAMLID">_90DBA0754E55B4FE7013928310431176</ns4:KeyIdentifier>
               </ns4:SecurityTokenReference>
            </ns2:RequestedUnattachedReference>
            <ns2:Lifetime>
               <ns3:Created>2014-02-19T17:30:43.119Z</ns3:Created>
               <ns3:Expires>2014-02-19T18:00:43.119Z</ns3:Expires>
            </ns2:Lifetime>
         </ns2:RequestSecurityTokenResponse>
      </ns2:RequestSecurityTokenResponseCollection>
   </soap:Body>
</soap:Envelope>
```

XACML Policy Decision Point (PDP)

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	read	security-pdp-xacmlrealm	When performing any redaction or filtering, the XACMLRealm will set the action-id to "read".
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.

NOTE 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 metacard security property. When the permissions are of a key-value type, the key being used is populated as the AttributeId value under the urn:oasis:names:tc:xacml:3.0:attribute-category:resource category.

Example Requests and Responses

The following items are a sample request, response, and the corresponding policy. 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 metacard's RESOURCE_ACCESS attributes. The query and LocalSiteName actions differ, as they are used to perform service authorization. For a query, the user must be associated with the country code ATA (Antarctica), and a LocalSiteName action can be performed by anyone.

Policy

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

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

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

Metacard Authorization

Subject Permitted

All of the resource's RESOURCE_ACCESS attributes were matched with the Subject's SUBJECT_ACCESS attributes.

Request

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

The Expansion Service and its corresponding expansion-related commands provide an easy way for developers to add expansion capabilities to DDF 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 metacard 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.metacard.attribute.mapping	This instance is configured with rules that expand the metacard's security attributes before comparing with the user's attributes.

Each instance of the expansion service can be configured using a configuration file. The configuration file can have three different types of lines: * comments - any line prefixed with the # character is ignored as a comment (for readability, blank lines are also ignored) * attribute separator - a line starting with separator= defines the attribute separator string. * rule - all other lines are assumed to be rules defined in a string format `<key>:<original value>:<new value>`

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

```
# This defines the separator that will be used when the expansion string contains
multiple
# values - each will be separated by this string. The expanded string will be split at
the
# separator string and each resulting 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

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 clientKeystore.jks	Server used for incoming connections. Client used for outgoing connections.

link:Distributed WSS.pptx[Distributed WSS]

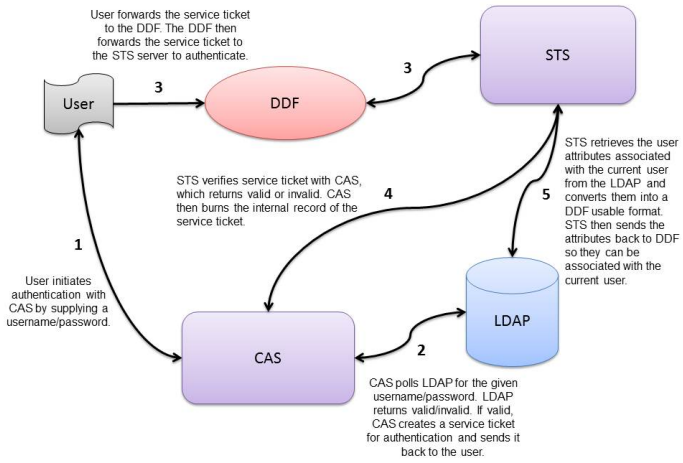


Figure 1. Login Authentication Scheme

Authentication Components

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).

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.

WARNING	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.

Configuring 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.
7. 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

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

WARNING 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: https://localhost:8993/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
```

Disable the Admin Console

==== 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

Property	Sample Value	Description
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,TLSv1.2The protocols that are supported when making outbound HTTPS connections

Configure DDF without the Web Administration Console

NOTE	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.

WARNING	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.)

   a. **Mandatory**: The dash between the PID (e.g., `OpenSearchSource_site.cfg`) and the instance name (e.g., `OpenSearchSource_site.cfg`) is required. The dash is a reserved character used by OSGi that identifies instances of a managed service factory that should be created.

   b. Not required, but a good practice is to change the instance name (e.g., `federated_source`) of the file to something identifiable (`source1- ddf`).

2. Edit the copied file to etc with the settings for the configuration. (Refer to the table above to determine the configurable properties).

   a. This file is a Java properties file, hence the syntax is `<key>` = `<value>`.

   b. Consult the inline comments in the file for guidance on what to modify.

   c. The Configurable Properties tables in the Integrator's Guide for the Included Catalog Components also describe each field and its value.

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

   a. **Mandatory**: The dash between the PID (e.g., `OpenSearchSource_site.cfg`) and the instance name

(e.g., `OpenSearchSource_site.cfg`) is required. The dash is a reserved character used by OSGi that identifies instances of a managed service factory that should be created.

```
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
https://ddf:8993/services/query?q={searchTerms}&amp;src={fs:routeTo?}&amp;mr={fs:maxRes
ults?}&amp;count={count?}&amp;mt={fs:maxTimeout?}&amp;dn={idn:userDN?}&amp;lat={geo:lat
?}&amp;lon={geo:lon?}&amp;radius={geo:radius?}&amp;bbox={geo:box?}&amp;polygon={geo:pol
ygon?}&amp;dtstart={time:start?}&amp;dtend={time:end?}&amp;dateName={cat:dateName?}&amp
;filter={fsa:filter?}&amp;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

NOTE *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.

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

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

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

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

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

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

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

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

Deployment Guidelines

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

1. Prior to allowing a hot deployment, check the available storage space on the system to ensure the deployment will not exceed the available space.

2. Set maximum storage space on the `DDF_HOME/deploy` and `DDF_HOME/system` directories to restrict the amount of space used by deployments.

3. Do not assume the deployment is from a trusted source; verify its origination.

4. Use the source code to verify a deployment is required for DDF to prevent unnecessary/vulnerable deployments.

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.

1. Prior to deploying a bundle/feature, verify the schema validation if it is a DDF endpoint.
2. Modify the `beans.xml` file
   a. 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
      ```

   b. Change directory to the `beans.xml` file.

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

   c. Open the `beans.xml` file in an editor (e.g., vi).

   d. Search for `schema-validation-enabled` and change its value to true.

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

   e. Save and close the file.

   f. Change directory to the feature directory,

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

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

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

4. Re-install the feature.

a. Open the Web browser and navigate to the DDF Web Console,

b. Select the Install button. Schema validation is now enabled for the endpoint.

Assuring Authenticity

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 06 17:05:46 MST 2014 META-INF/MANIFEST.MF
        9503 Mon Oct 06 17:05:46 MST 2014 META-INF/SELFSIGN.SF
        1303 Mon Oct 06 17:05:46 MST 2014 META-INF/SELFSIGN.RSA
           0 Wed Sep 17 17:14:06 MST 2014 META-INF/
           0 Wed Sep 17 17:14:10 MST 2014 META-INF/maven/
           0 Wed Sep 17 17:14:10 MST 2014 META-INF/maven/ddf.catalog/
           0 Wed Sep 17 17:14:10 MST 2014 META-INF/maven/ddf.catalog/catalog-app/
smk     4080 Wed Sep 17 16:54:18 MST 2014 META-INF/maven/ddf.catalog/catalog-app/pom.xml
smk      107 Wed Sep 17 17:14:06 MST 2014 META-INF/maven/ddf.catalog/catalog-
app/pom.properties
           0 Wed Sep 17 17:14:06 MST 2014 repository/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/catalog-app/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/catalog-app/2.5.1/
smk    12543 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/catalog-
app/2.5.1/catalog-app-2.5.1-features.xml
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/core/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/core/catalog-core-api/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/catalog/core/catalog-core-
api/2.5.1/
smk   188995 Wed Sep 17 16:55:28 MST 2014 repository/ddf/catalog/core/catalog-core-
api/2.5.1/catalog-core-api-2.5.1.jar
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/core/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/core/mime-core-api/
           0 Wed Sep 17 17:14:06 MST 2014 repository/ddf/mime/core/mime-core-api/2.5.0/
smk     4396 Wed Sep 10 12:38:24 MST 2014 repository/ddf/mime/core/mime-core-
api/2.5.0/mime-core-api-2.5.0.jar
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-core/
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-core/1.2/
smk   463945 Thu Feb 13 09:26:04 MST 2014 repository/org/apache/tika/tika-core/1.2/tika-
core-1.2.jar
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-bundle/
           0 Wed Sep 17 17:14:06 MST 2014 repository/org/apache/tika/tika-bundle/1.2/
smk   22360866 Thu Feb 13 09:26:54 MST 2014 repository/org/apache/tika/tika-
bundle/1.2/tika-bundle-1.2.jar
           0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/
           0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/
           0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/gt-opengis/
           0 Wed Sep 17 17:14:08 MST 2014 repository/org/codice/thirdparty/gt-
opengis/8.4_1/
```

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

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

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

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

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

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

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

```
           0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-logging/
           0 Wed Sep 17 17:14:10 MST 2014 repository/ddf/catalog/security/catalog-
security-logging/2.5.1/
smk     6768 Wed Sep 17 17:13:58 MST 2014 repository/ddf/catalog/security/catalog-
security-logging/2.5.1/catalog-security-logging-2.5.1.jar
  s = signature was verified
  m = entry is listed in manifest
  k = at least one certificate was found in keystore
  i = at least one certificate was found in identity scope
jar verified.
```

Note the last line: *jar verified*. This indicates that the signatures used to sign the JAR (or in this case, KAR) were valid according to the trust relationships specified by the keystore.

Security

Configure STS Subject DN constraints

The configuration for the STS Subject DN constraints is located in the `ws-security.subject.cert.constraints property`in the `<DDF_INSTALL_DIR>/etc/system.properties` file.

The default constraints are ".*", which allows any connection with a trusted certificate the ability to access the STS. This opens up the vulnerability for a client to "pretend" to be another client. To prevent this, modify this property to be a list of regular expressions that map to the fully qualified hostnames of systems allowed to contact the STS.

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

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:

Shut 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
```

IMPORTANT	*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.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
```

WARNING	While it is not a necessary step, information on how to convert the System V init scripts to the Solaris System Management Facility can be found at http://www.oracle.com/technetwork/articles/servers-storage-admin/scripts-to-smf-1641705.html
WARNING	*Solaris-Specific Modification* Due to a slight difference between the Linux and Solaris implementation of the ps command, the ddf-service script needs to be modified.

6. Locate the following line in DDF_INSTALL/bin/ddf-service

Solaris DDF_INSTALL/bin/ddf-service

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

7. Change the word command to comm.

Solaris DDF_Install/bin/ddf-service

```
pidtest=`$PSEXE -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.

Console Commands

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 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.

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

Catalog Commands

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

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

Commands

```
catalog:describe     catalog:dump         catalog:envlist      catalog:ingest
catalog:inspect
catalog:latest       catalog:migrate      catalog:range        catalog:remove
catalog:removeall
catalog:replicate    catalog:search       catalog:spatial
```

Command Descriptions

Command	Description
describe	Provides a basic description of the Catalog implementation.
dump	Exports metacards from the local Catalog. Does not remove them. See below for date filtering options.
envlist	[IMPORTANT] ==== Deprecated as of ddf-catalog 2.5.0. Please use platform:envlist. ==== Provides a list of environment variables.
ingest	Ingests data files into the Catalog.
inspect	Provides the various fields of a metacard for inspection.
latest	Retrieves the latest records from the Catalog based on the Metacard.MODIFIED date.
migrate	Allows two CatalogProviders to be configured and migrates the data from the primary to the secondary.
range	Searches by the given range arguments (exclusively).
remove	Deletes a record from the local Catalog.
removeall	Attempts to delete all records from the local Catalog.
replicate	Replicates data from a federated source into the local Catalog.
search	Searches records in the local Catalog.
spatial	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 metacard was created, while `--modified-after` and `--modified-before` options allow filtering on the date and time that the metacard 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 metacard 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

Issue	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

Command Scheduler is a capability exposed through the Admin Console (https://localhost:8993/admin) 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 Admin Console (https://localhost:8993/admin).
2. Select DDF Platform
3. Select **Platform Command Scheduler**.
4. Type the command or commands to be executed in the **Command** text field. Commands can be separated by a semicolon and will execute in order from left to right.
5. Type in a positive integer for the **Interval In Seconds** field.
6. Select the **Save** button. Once the **Save** button is selected, the command is executed immediately. It's next scheduled execution begins after the amount of seconds specified in the **Interval In Seconds** field and repeats indefinitely until the container is shut down or the scheduled command is deleted.

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

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

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.

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

List a Specific Subscription by ID

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

Total subscriptions found: 1

Subscription ID
my.contextual.id.v20|http://172.18.14.169:8088/mockCatalogEventConsumerBinding?WSDL
```

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

List Subscriptions Using Wildcards

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

Total subscriptions found: 3

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


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

Total subscriptions found: 1

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


ddf@local>subscriptions:list "*WSDL"

Total subscriptions found: 2

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

List Subscriptions Using an LDAP Filter

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

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

Total subscriptions found: 2

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

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

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

Total subscriptions found: 1

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

`subscriptions:delete` Command Usage Example

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

Delete a Specific Subscription Using Its Exact ID

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

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

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

Delete Subscriptions Using Wildcards

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

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

Deleted 2 subscriptions out of 2 subscriptions found.




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

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

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

Delete All Subscriptions

```
ddf@local>subscriptions:delete *

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

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

Delete Subscriptions Using an LDAP Filter

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

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

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

Platform Commands

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

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

Commands

Command Descriptions

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

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

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

Commands

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

Command Descriptions

Command	Description
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: metacard, 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

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.

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 ingesting)

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 ingesting data into the catalog. For further information, see Integrating Endpoints.

Troubleshooting

Exception Starting DDF (Windows)

Problem:

An exception is sometimes 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).

Blank Web Console

Problem:

https://localhost:8993/system/console opens as a blank page.

Solution:

Restart DDF. . Shut down DDF from the console:
`ddf@local>shutdown` . Start DDF back up: `./ddf` . Verify that all of the files were copied over correctly during the deploy bundles step.

CXF BusException

Problem:

The following exception is thrown: `org.apache.cxf.BusException: No conduit initiator`

Solution:

Restart DDF. . Shut down DDF:
`ddf@local>shutdown` . Start up DDF: `./ddf`

Distribution Will Not Start

Problem:

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

Solution:

Complete the following procedure. . Verify that Java is correctly installed.

+ `java -version` . This should return something similar to:

+ `java version "1.8.0_45" Java  SE Runtime Environment (build 1.8.0_45-b14) Java HotSpot  Server VM (build 25.45-b02, mixed mode)` . If running *nix, verify that bash is installed.

+ `echo $SHELL` . This should return:

+ `/bin/bash`

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.