Managing DDF

Table of Contents

1. License 1
2. Overview 1
   2.1. Installing DDF 1
   2.2. Configuring DDF 8
   2.3. Securing DDF 40
   2.4. Running DDF 108
   2.5. Troubleshooting DDF 139
3. Managing DDF Admin 142
   3.1. Installing DDF Admin 142
   3.2. Console Commands 144
   3.3. Administrative User Interface 144
   3.4. Admin Console Access Control 147
4. Managing DDF Catalog 149
   4.1. Installing DDF Catalog 149
   4.2. Configuring DDF Catalog 149
5. Managing DDF GeoWebCache 158
   5.1. Prerequisites for DDF GeoWebCache 158
   5.2. Installing DDF GeoWebCache 158
   5.3. Configuring DDF GeoWebCache 158
6. Managing DDF Platform 161
   6.1. Using DDF Platform Application 161
   6.2. Installing and Uninstalling Platform 161
7. Managing DDF Registry 163
   7.1. DDF Registry Prerequisites 163
   7.2. Installing DDF Registry 163
   7.3. Configuring DDF Registry 163
   7.4. Using DDF Registry 166
8. Managing DDF Security 171
   8.1. Installing DDF Security 171
   8.2. Configuring DDF Security 171
9. Managing DDF Solr 186
   9.1. DDF Catalog Solr External Provider 186
   9.2. DDF Catalog Solr Embedded Provider 187
   9.3. Standalone Solr Server 190
10. Managing DDF Spatial 194
   10.1. Prerequisites 194
   10.2. Installing 194
   10.3. Optional Features 196
   10.4. Console Commands 196
11. Managing DDF Search UI 198
   11.1. Prerequisites 198
   11.2. Installing DDF Search UI 198

11.3. Configuring DDF Search UI 198
11.4. Troubleshooting DDF Search UI 201

1. License

2. Overview

2.1. Installing DDF

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

2.1.1. Prerequisites

Directory Permissions

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

Directory Permissions on Windows

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

- Install/Upgrade to Java 8 J2SE 8 SDK
- Supported platforms are *NIX - Unix/Linux/OSX, Solaris, and Windows.
- JDK8 must be installed.
- The `JAVA_HOME` environment variable must be set to the location where the JDK is installed.

*Setting `JAVA_HOME` on *NIX*

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

Setting `JAVA_HOME` on Windows

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

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
```

Directory Permissions on *NIX

Protect the DDF from unauthorized access.

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

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

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

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

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

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

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

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

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

2.1.2. Deployment Guidelines

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

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

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

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

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

2.1.3. Installing With the DDF Distribution Zip

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

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
```

Execute the following command at the command line for status:

View Status

```
ddf@local>list
```

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

Additional Configuration Step

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

- Run the DDF using the appropriate script.

**NIX*

```
<INSTALL_DIRECTORY>/ddf-2.9.1/bin/ddf
```

Windows

```
<INSTALL_DIRECTORY>/ddf-2.9.1/bin/ddf.bat
```

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

View Status

```
DDF@local>list
```

The list of bundles should look similar to this:

DDF List of Apps Installed

```
.DDF@local>list
START LEVEL 100 , List Threshold: 50
   ID   State         Blueprint      Spring    Level  Name
[ 111] [Active     ] [            ] [       ] [   80] Commons IO (2.1.0)
[ 113] [Active     ] [Created     ] [       ] [   80] DDF :: Distribution :: Console
Branding Plugin (2.9.1)
```

WARNING

DDF Application Installation Dependencies

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

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

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

Verifying Installation

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

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

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

View Status

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

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

Example: Bundle Fragment in the Command Line Console

```
215 | Resolved |  80 | 2.9.1     | DDF :: Platform :: Security :: Session, Hosts: 90
```

DDF Directory Contents after Installation and Initial Startup

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

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 (optio This log rolls up to 10 times, frequency based on a confi MB)
	`data/log/ingest.log`	Log file for any ingest errors that occur within DDF.
	`data/log/security.log`	Log file that records user interactions with the system f
`deploy`	Hot-deploy directory – KARs and bundles added to this directory will be hot-deployed (Empty upon DD	
`documentation`	HTML and PDF copies of DDF documentation.	
`etc`	Directory monitored for addition/modification/deletion of `.config` configuration files or third party `.cf`	
	`etc/failed`	If there is a problem with any of the `.config` files, such a tokens, they will be moved here.
	`etc/processed`	All successfully processed `.config` files will be moved he
	`etc/templates`	Template `.config` files for use in configuring DDF source copying to the etc directory.
`lib`	The system's bootstrap libraries. Includes the `ddf-branding.jar` file which is used to brand the system c	
`licenses`	Licensing information related to the system.	
`system`	Local bundle repository. Contains all of the JARs required by DDF, including third-party JARs.	

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

2.2. Configuring DDF

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

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

- RECOMMENDED: Using the browser and the Admin Console. Configuring DDF From the Admin Console
- Using a terminal and the Command Console. Configuring DDF From the Command Console
- Editing configuration files. Configuring DDF From Configuration Files
- Importing the configurations from an existing DDF instance. Importing Configurations

2.2.1. Configuring DDF From the Admin Console

Accessing the Admin Console

- Navigate to the Admin Console. (Default: https://localhost:8993/admin)
- Enter Username and Password. (Default: `admin/admin`)

Initial Configuration

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

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

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

Viewing Currently Active Applications from Admin Console

Tile View

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

List View

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

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

Configuration

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

Details

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

Features

The features tab breaks down the individual features of the application that can be installed or uninstalled as configurable features.

Managing Applications

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

Activating / Deactivating Applications

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

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

Adding Applications

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

Removing Applications

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

Upgrading Applications

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

System Settings Tab

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

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

Managing Features Using the Admin Console

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

Add Feature Repositories

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

Configuring HTTP Port from Admin Console

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

Configuring HTTP to HTTPS Proxy From the Admin Console

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

1. Click the **DDF Platform** application tile.
2. Choose the **Features** tab.

3. Select `platform-http-proxy`.
4. Click on the **Play** button to the right of the word “Uninstalled”

Configuring the proxy:

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

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

2.2.2. Configuring DDF From the Command Console

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

In an environment hardened for security purposes, access to the DDF Admin Console might be denied. It is necessary to configure DDF (e.g., providers, Schematron rulesets, etc.) using `.config` files or the Admin Console. Configuration via the Karaf command line console is not supported and may result in configuration errors. The OSGi container detects the creation of `.config` files in the `etc/` directory. The following sections describe how to configure each DDF item using both of these mechanisms. A template file is provided for some configurable DDF items so that they can be copied/renamed then modified with the appropriate settings.

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

Templates included with DDF:

DDF Service	**Template File Name**	**Factory PID**	**Configurable Properties**
DDF Catalog Framework	`ddf.catalog.impl.service.CatalogFrameworkImpl.cfg`	`ddf.catalog.CatalogFrameworkImpl`	Standard Catalog Framework

Configuring Using a `.cfg` File Template

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

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 Admin Console.

Managing Applications From the Command Console

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

Table 1. App Commands

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

Managing Features From the Command Console

1. Determine which feature to install by viewing the available features on DDF.
   `ddf@local>feature:list`

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

```
State         Version           Name                                    Repository
Description
[installed  ] [2.9.1  ] security-handler-api                    security-services-app-
2.9.1 API for authentication handlers for web applications.
[installed  ] [2.9.1  ] security-core                           security-services-app-
2.9.1 DDF Security Core
[uninstalled] [2.9.1  ] security-expansion                      security-services-app-
2.9.1 DDF Security Expansion
[uninstalled] [2.9.1  ] security-cas-client                     security-services-app-
2.9.1 DDF Security CAS Client.
[uninstalled] [2.9.1  ] security-cas-tokenvalidator             security-services-app-
2.9.1 DDF Security CAS Validator for the STS.
[uninstalled] [2.9.1  ] security-cas-cxfservletfilter           security-services-app-
2.9.1 DDF Security CAS Servlet Filter for CXF.
[installed  ] [2.9.1  ] security-pdp-authz                      security-services-app-
2.9.1 DDF Security PDP.
[uninstalled] [2.9.1  ] security-pep-serviceauthz               security-services-app-
2.9.1 DDF Security PEP Service AuthZ
[uninstalled] [2.9.1  ] security-expansion-user-attributes      security-services-app-
2.9.1 DDF Security Expansion User Attributes Expansion
[uninstalled] [2.9.1  ] security-expansion-metacard-attributes  security-services-app-
2.9.1 DDF Security Expansion Metacard Attributes Expansion
[installed  ] [2.9.1  ] security-sts-server                     security-services-app-
2.9.1 DDF Security STS.
[installed  ] [2.9.1  ] security-sts-realm                      security-services-app-
2.9.1 DDF Security STS Realm.
[uninstalled] [2.9.1  ] security-sts-ldaplogin                  security-services-app-
2.9.1 DDF Security STS JAAS LDAP Login.
[uninstalled] [2.9.1  ] security-sts-ldapclaimshandler          security-services-app-
2.9.1 Retrieves claims attributes from an LDAP store.
```

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

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

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

Uninstall Features

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

`ddf@local>feature:list`

```
State         Version          Name                           Repository
Description
[installed  ] [2.9.1         ] ddf-core                       ddf-2.9.1
[uninstalled] [2.9.1         ] ddf-sts                        ddf-2.9.1
[installed  ] [2.9.1         ] ddf-security-common            ddf-2.9.1
[installed  ] [2.9.1         ] ddf-resource-impl              ddf-2.9.1
[installed  ] [2.9.1         ] ddf-source-dummy               ddf-2.9.1
```

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

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

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

```
State         Version          Name                           Repository  Description
[installed  ] [2.9.1         ] ddf-core                       ddf-2.9.1
[uninstalled] [2.9.1         ] ddf-sts                        ddf-2.9.1
[installed  ] [2.9.1         ] ddf-security-common            ddf-2.9.1
[installed  ] [2.9.1         ] ddf-resource-impl              ddf-2.9.1
[uninstalled] [2.9.1         ] ddf-source-dummy               ddf-2.9.1
```

Configuring HTTP Port from the Command Console

2.2.3. Configuring HTTP to HTTPS Proxy From the Command Console

NOTE	If DDF has not been installed, configure from Admin Console.

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

2.2.4. Configuring DDF From Configuration Files

2.2.5. Configuring DDF Global Settings

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

Table 2. Configurable Properties

Title	Property	Type	Description	Default Value	Required

Keystore and truststore java properties					
Keystore	javax.net.ssl.keyStore	String	Path to server keystore	etc/keystores/serverKeystore.jks	Yes
Keystore Password	javax.net.ssl.keyStorePassword	String	Password for accessing keystore	changeit	Yes
Truststore	Truststorejavax.net.ssl.trustStore	String	The trust store used for SSL/TLS connections. Path is relative to <INSTALL_HOME>.	etc/keystores/serverTruststore.jks	Yes
Truststore Password	javax.net.ssl.trustStorePassword	String	Password for server Truststore	changeit	Yes
Keystore Type	javax.net.ssl.keyStoreType	String	File extension to use with server keystore	jks	Yes
Truststore Type	javax.net.ssl.trustStoreType	String	File extension to use with server truststore	jks	Yes
Global URL Properties					
Protocol	org.codice.ddf.system.protocol	String	Default protocol that should be used to connect to this machine.	https://	Yes
Hostname	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

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
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
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
Root Context	`org.codice.ddf.system.rootContext`	String	The the base or root context that services will be made available under.	`/services`	Yes
System Information Properties					
Site Name	`org.codice.ddf.system.id`	String	The site name for DDF.	`ddf.distribution`	Yes
Site Contact	`org.codice.ddf.system.siteContact`	String	The email address of the site contact.		No
Version	`org.codice.ddf.system.version`	String	The version of DDF that is running. This value should not be changed from the factory default.	`2.9.1`	Yes
Organization	`org.codice.ddf.system.organization`	String	The organization responsible for this installation of DDF.	Codice Foundation	Yes
Thread Pool Settings					
Thread Pool Size	`org.codice.ddf.system.threadPoolSize`	Integer	Size of thread pool used for handling UI queries, federating requests, and downloading resources	`128`	Yes

HTTPS Specific Settings					

Cipher Suites	`https.cipherSuites`	String	Cipher suites to use with secure sockets	`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`
					No

Https Protocols	`https.protocols`	String	Protocols to allow for secure connections	`TLSv1.1,TLSv1.2`	No
Allow Basic Auth Over Http	`org.codice.allowBasicAuthOverHttp`	Boolean	Set to true to allow Basic Auth credentials to be sent over HTTP unsecurely. This should only be done in a test environment. These events will be audited.	`false`	Yes
Restrict the Security Token Service to allow connections only from DNs matching these patterns	`ws-security.subject.cert.constraints`	String	Set to a comma separated list of regex patterns to define which hosts are allowed to connect to the STS	`.*`	Yes
XML Settings					

Parse XML documents into DOM object trees	`javax.xml.parsers.DocumentBuilderFactory`	String	Enables Xerces-J implementation of `DocumentBuilderFactory`	`org.apache.xerces.jaxp.DocumentBuilderFactoryImpl`	Yes
File Upload Settings					
File extensions flagged as potentially dangerous to the host system or external clients	`bad.file.extensions`	String	Files uploaded with these bad file extensions will have their file names sanitized before being saved	`.exe,.jsp,.html,.js,.php,.phtml,.php3,.php4,.php5,.phps,.shtml,.jhtml,.pl,.py,.cgi,.msi,.com,.scr,.gadget,.application,.pif,.hta,.cpl,.msc,.jar,.kar,.bat,.cmd,.vb,.vbs,.vbe,.jse,.ws,.wsf,.wsc,.wsh,.ps1,.ps1xml,.ps2,.ps2xml,.psc1,.psc2,.msh,.msh1,.msh2,.mshxml,.msh1xml,.msh2xml,.scf,.lnk,.inf,.reg,.dll,.vxd,.cpl,.cfg,.config,.crt,.cert,.pem,.jks,.p12,.p7b,.key,.der,.csr,.jsb,.mhtml,.mht,.xhtml,.xht`	Yes

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

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

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

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

IMPORTANT	Since certain bundles can only be configured using the `.config` file format, this file format should be used.
WARNING	Only root can access ports < 1024 on Unix systems. For suggested ways to run DDF with ports < 1024 see [How do I use port 80 as a non-root user?].

Configuring DDF `.config` Files

The DDF is configured using `.config` files. Like the Karaf `.cfg` files, these configuration files must be

located in the `<DDF_HOME>/etc/` directory, have a name that matches the *configuration persistence ID* (PID) they represent, and have a `service.pid` property set to the configuration PID.

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

IMPORTANT	This new configuration file format **must** be used for any configuration that makes use of lists. Examples include Web Context Policy Manager (PID: `org.codice.ddf.security.policy.context.impl.PolicyManager`) and Security STS Guest Claims Handler (PID: `ddf.security.sts.guestclaims`).
WARNING	Only one configuration file should exist for any given PID. The result of having both a `.cfg` and a `.config` file for the same PID is undefined and could cause the application to fail.

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

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

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

Configuring Managed Service Factory Bundles

Services that are created using a Managed Service Factory can be configured using `.config` files as well. The configuration files follow a different naming convention however. The files must start with the Managed Service Factory PID, be followed by a unique identifier and have a `.config` extension. For instance, assuming that the Managed Service Factory PID is `org.codice.ddf.factory.pid` and two instances of the service need to be configured, files `org.codice.ddf.factory.pid.uniqueID1.config` and `org.codice.ddf.factory.pid.uniqueID2.config` should be created and added to `<DDF_HOME>/etc`.

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

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

File Format

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

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

Type	Format	Example
Service PID	`service.pid = "servicePid"`	`service.pid = "org.codice.ddf.security.policy.context.impl.PolicyManager"`
Factory PID	`service.factoryPid = "serviceFactoryPid"`	`service.factoryPid = "Csw_Federated_Source"`
Strings	`name = "value"`	`name = "john"`
Booleans	`name = B"true|false"`	`authorized = B"true"`
Integers	`name = I"value"`	`timeout=I"60"`
Longs	`name = L"value"`	`diameter = L"10000"`
Floats	`name = F"value"`	`cost = F"10.50"`
Doubles	`name = D"value"`	`latitude = D"45.0234"`
Lists of Strings	`name = [ "value1", "value2", ]`	`authenticationTypes = [ "/\=SAML|GUEST", "/admin\=SAML|basic", "/system\=basic", "/solr\=SAML|PKI|basic", "/sources\=SAML|basic", "/security-config\=SAML|basic" ]`
Lists of Integers	`name = I[ "value1", "value1", ]`	`sizes = I[ "10", "20", "30" ]`

NOTE

- Lists of values can be prefixed with any of the supported types (`B`, `I`, `L`, `F` or `D`)
- To prevent any configuration issues, the `=` signs used in values should be escaped using `\`
- Boolean values will default to `false` if any value other than `true` is provided

Sample configuration file

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

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

realms=["/\=karaf"]

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

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

Editing HTTP Ports for Multiple Local DDF Nodes

Edit the port numbers in the files in the DDF install folder.

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

2.2.6. Editing DDF Web Service Providers Configuration Files

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

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

Configuring Files in HOME Directory Hierarchy

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

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

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

To be:

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

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

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

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

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

2.2.7. Configuring Notifications

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

Using Notifications

DDF notifications are currently being utilized in the DDF Catalog application for resource retrieval. When a user initiates a resource retrieval via the DDF Standard Search UI, DDF opens the channel `/ddf/notification/catalog/downloads`, where notifications indicating the progress of that resource download are sent. Any client interested in receiving these progress notifications must subscribe to

that channel. When DDF starts downloading the resource to the client that requested it, a notification with a status of "Started" will be broadcast. If the resource download fails, a notification with a status of "Failed" will be broadcast. Or, if the resource download is being attempted again after a failure, "Retry" will be broadcast.

When a notification is received, DDF Standard UI displays a popup containing the contents of the notification, so a user is made aware of how their downloads are proceeding.

Behind the scenes, the DDF Standard Search UI invokes the REST endpoint to retrieve a resource. In this request, it adds the query parameter "user" with the CometD session ID or the unique User ID as the value. This allows the CometD server to know which subscriber is interested in the notification. For example, http://DDF_HOST:8993/services/catalog/sources/ddf.distribution/2f5db9e5131444279a1293c541c106cd?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/**`

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);
```

2.2.8. Configuring Solr Catalog Provider Data Directory

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

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

Edit the `system.properties` file

```
# Uncomment the following line and set it to the desired path
#solr.catalog.data.dir=/opt/solr/catalog/data
```

Changing the Data Directory after DDF has ingested data

1. Shut down DDF.

2. Create the new directory to hold the indexes.

   Make new Data Directory

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

3. Copy the indexes to the new directory.

   Copy the indexes to the new Directory.

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

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

   Set the SOLR_CATALOG_DATA_DIR

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

5. Restart DDF.

Configuring Thread Pools

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

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

2.2.9. Importing Configurations

The Configuration Export/Import capability allows administrators to export the current DDF configuration and use it as a starting point for a new installation.

IMPORTANT

- Importing configuration files is only guaranteed to work when importing files from the same DDF version. Importing from a different version is not recommended as it may cause the new DDF instance to be incorrectly configured and become unusable.
- All configurations editable in the Admin Console will be exported to `<DDF_HOME>/etc/exported/etc`.
- All other configuration files (system configurations) will be put under `etc/exported/<ID>` followed by their relative path from `DDF_HOME`. For instance, if a key store file was located under `<DDF_HOME>/keystores/keystore.jks`, it will be exported to `<DDF_HOME>/etc/exported/<ID>/keystore/keystore.jks`.
- To keep the export/import process simple and consistent, all system configuration files are required to be under the `DDF_HOME` directory.

2.2.10. Exporting Configurations from Admin Console

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

To do so, follow these instructions:

1. Select the `System` tab (next to the Applications tab)
2. Click the `Export Configuration` button
3. Fill out the form, specifying the destination for the export. A relative path will be relative to DDF home.
4. Click the `Start Export` button
5. If there are no warnings or errors, the form will automatically close upon finishing the export

2.2.11. Export Configuration Settings from Command Console

To export the current DDF configuration:

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

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

Troubleshooting Common Warnings or Failures

Insufficient Write Permissions

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

Properties Set to Absolute File Paths

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

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

2.2.12. Import Configuration Settings

To import a previously exported configuration:

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

2.2.13. Managing Features

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

NOTE

Transitive Dependencies

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

2.2.14. Configuring DDF with New Certificates

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

Table 3. Important Terms for Certificates

Term	Definition	Example Value
`<DDF_HOME>`	The path to the unzipped DDF distribution	`/opt/ddf/ddf-2.9.1`
alias	The nickname given to a certificate within a keystore to make it easily identifiable. Normally, the alias should be the FQDN.	`localhost`
certificate	A combination of an entity's identity information with the entity's public key. The entity can be a person, organization, or something else, but in this case the entity is a computer on the network. To be valid, a certificate must be digitally (cryptographically) signed by a certificate authority. By signing a certificate, the CA attests that the public key truly belongs to the entity and no one else. See also **PKIX**.	`<FQDN>.crt`
CN	Common Name - The FQDN of the DDF instance as defined within the Certificate.	`search.codice.org`

Term	Definition	Example Value
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://`
JKS	Java **keystore**. A dictionary of cryptographic objects (e.g. private keys, certificates) referenced by an **alias**. The JKS format is specific to 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.	

Term	Definition	Example Value
keytool	The Java keytool is a key and certificate management command line utility.	
openssl	The openssl program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell.	
PKCS#12	Personal Information Exchange Syntax. A standard that allows certificates, 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 known 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.	

2.2.15. Configuring DDF Web Service Providers

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

This can be changed during Initial Configuration or later by editing the `<INSTALL_HOME>/etc/system.properties` file. See Editing DDF Web Service Providers Configuration Files

Creating and Installing Keys and Certificates

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

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

The above command creates a new entry in the keystore for a server named `my.server.com`.

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

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

To create and install the certificates on Windows, use the `CertNew.cmd` file in the same directory.

```
CertNew <FQDN>
```

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

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

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

Restart and Test

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

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

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

```
127.0.0.1 <FQDN>
```

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

```
Enable Backup Plugin: true
```

IMPORTANT

The Embedded LDAP has hard-coded values for the keystore path, truststore path, keystore password, and truststore password (https://github.com/codice/opendj-osgi/blob/d5021cbac4db831467ceb109ffd7ffd2c734dcd4/embedded/opendj-embedded-server/src/main/resources/config/config.ldif). So if using a non-default keystore and non-default truststore, the Embedded LDAP will not work. You will see errors in `<INSTALL_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.
  - `ds-cfg-key-store-file: ../../keystores/serverKeystore.jks`
  - `ds-cfg-key-store-type: JKS`
  - `ds-cfg-key-store-pin: password`
  - `cn: JKS`
- The default password in `config.ldif` for `serverTruststore.jks` is `changeit`. This needs to be modified.
  - `ds-cfg-trust-store-file: ../../keystores/serverTruststore.jks`
  - `ds-cfg-trust-store-pin: password`
  - `cn: JKS`

2.2.16. Configuring DDF to Use an LDAP Server

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

Use the encryption service, described in Encryption Service, on the command line to set passwords for your LDAP server. Then change the LDAP Bind User Password in the configurations to use the encrypted password.

Table 4. STS Ldap Login Configuration

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

Configuring STS Claims Handlers

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

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

Table 5. Security STS LDAP and Roles Claims Handler

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

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

2.2.17. Standalone Security Token Service (STS) Installation

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

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

2.2.18. Configuring DDF Logging Service

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

2.2.19. Managing Asynchronous Capabilities (Search & Retrieval)

Installing the Asynchronous Capabilities Endpoint (CometD)

The CometD endpoint enables asychronous search capabilties.

It is installed by default with the Search UI application.

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

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

- The bundle is the DDF SearchUI Endpoint

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

Configuring the Product Cache

All caching properties are part of the Catalog Framework Configuration

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

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

Invalidating the Product Cache

1. The product cache directory can be administratively invalidated by turning off the product caching using the cacheEnabled property.

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

Format:

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

Example:

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

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

2.3. Securing 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.

2.3.1. Securing the Web Context Policy Manager

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

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

See Web Context Policy Manager Configurations for detailed descriptions of all fields.

Context Realms

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

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

Authentication Types

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

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

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

Table 6. Default Types of Authentication

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

2.3.2. Required Attributes

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

White Listed Contexts

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

2.3.3. Limiting Access to the STS

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

- Open `<INSTALL_HOME>/etc/system.properties`
- Edit the line `ws-security.subject.cert.constraints = .*`
  - Remove the `.*` and replace with the desired host (<MY_HOST>):
    - `ws-security.subject.cert.constraints = <MY_HOST>.*`

2.3.4. IdP/SP

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

IdP (Identity Provider) and SP (Service Provider)

IdP and SP are used for SSO authentication purposes.

Installing the IdP

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

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

Security IdP Service Provider (SP)

The IdP client that interacts with the specified Identity Provider.

IdP SSO Configuration

1. Navigate to Admin Console DDF Security Configuration IdP Client
2. Populate IdP Metadata field through one of the following:
    a. An HTTPS URL (https://)
    b. A file URL (file:)
    c. An XML block to refer to desired metadata
        i. (e.g., https://localhost:8993/services/idp/login/metadata)

IdP Client (SP) example.xml

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID=
"https://localhost:8993/services/idp/login">
  <md:IDPSSODescriptor WantAuthnRequestsSigned="true" protocolSupportEnumeration=
"urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
            MIIDEzCCAnygAwIBAgIJAIzc4FYrIp9mMA0GCSqGSIb3DQEBBQUAMHcxCzAJBgNVBAYTAlVTMQswC
QYDVQQIDAJBWjEMMAoGA1UECgwDRERGMQwwCgYDVQQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJD
AiBgkqhkiG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGxlLm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MTh
aMIGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWFyMQwwCgYDVQQKDANEREYxDDAK
BgNVBAsMA0RldjESMBAGA1UEAwwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhbXBsZS5vc
mcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZphHQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx3l3A99
s8WtA4mRAeb6n0vTR9yNBOekW4nYOiEOq//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pklr0TDI
4MRzPPkkTp4wlORQ/a6CfVsNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBhvhCAQ0EGhYYRk9S
IFRFU1RJTkcgUFVSUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrsDAfBgNVHSMEGDAWgBThV
MeX3wrCv6lfeF47CyvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQ
ADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUxO7n1SmOESp5sE3jGTnh0Gt
YV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0hlSGV0=
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:KeyDescriptor use="encryption">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
            MIIDEzCCAnygAwIBAgIJAIzc4FYrIp9mMA0GCSqGSIb3DQEBBQUAMHcxCzAJBgNVBAYTAlVTMQswC
QYDVQQIDAJBWjEMMAoGA1UECgwDRERGMQwwCgYDVQQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJD
AiBgkqhkiG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGxlLm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MTh
aMIGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWFyMQwwCgYDVQQKDANEREYxDDAK
BgNVBAsMA0RldjESMBAGA1UEAwwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhbXBsZS5vc
mcwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZphHQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx3l3A99
s8WtA4mRAeb6n0vTR9yNBOekW4nYOiEOq//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pklr0TDI
4MRzPPkkTp4wlORQ/a6CfVsNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBhvhCAQ0EGhYYRk9S
IFRFU1RJTkcgUFVSUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrsDAfBgNVHSMEGDAWgBThV
MeX3wrCv6lfeF47CyvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQ
ADgYEAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUxO7n1SmOESp5sE3jGTnh0Gt
YV0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0hlSGV0=
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://localhost:8993/logout"/>
```

```
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:8993/logout"/>
    <md:NameIDFormat>
      urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
    </md:NameIDFormat>
    <md:NameIDFormat>
      urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
    </md:NameIDFormat>
    <md:NameIDFormat>
      urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
    </md:NameIDFormat>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://localhost:8993/services/idp/login"/>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:8993/services/idp/login"/>
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

2.3.5. Security IdP Server

An internal Identity Provider solution.

Configuring IdP Server

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

IdP Server example.xml

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID=
"https://localhost:8993/services/saml">
  <md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:X509Data>
        <ds:X509Certificate>
          MIIDEzCCAnygAwIBAgIJAIzc4FYrIp9mMA0GCSqGSIb3DQEBBQUAMHcxCzAJBgNVBAYTAlVTMQswCQY
DVQQIDAJBWjEMMAoGA1UECgwDRERGMQwwCgYDVQQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAi
BgkqhkiG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGxlLm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaM
IGDMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWFyMQwwCgYDVQQKDANEREYxDDAKBg
NVBAsMA0RldjESMBAGA1UEAwwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhbXBsZS5vcmc
wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZphHQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx3l3A99s8
WtA4mRAeb6n0vTR9yNBOekW4nYOiEOq//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pklr0TDI4M
RzPPkkTp4wlORQ/a6CfVsNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBhvhCAQ0EGhYYRk9SIF
RFU1RJTkcgUFVSUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrsDAfBgNVHSMEGDAWgBThVMe
X3wrCv6lfeF47CyvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQAD
gYEAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUxO7n1SmOESp5sE3jGTnh0GtYV
0D219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0hlSGV0=
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </md:KeyDescriptor>
  <md:KeyDescriptor use="encryption">
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:X509Data>
        <ds:X509Certificate>
        MIIDEzCCAnygAwIBAgIJAIzc4FYrIp9mMA0GCSqGSIb3DQEBBQUAMHcxCzAJBgNVBAYTAlVTMQswCQYDV
QQIDAJBWjEMMAoGA1UECgwDRERGMQwwCgYDVQQLDANEZXYxGTAXBgNVBAMMEERERiBEZW1vIFJvb3QgQ0ExJDAiBg
kqhkiG9w0BCQEWFWRkZnJvb3RjYUBleGFtcGxlLm9yZzAeFw0xNDEyMTAyMTU4MThaFw0xNTEyMTAyMTU4MThaMIG
DMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQVoxETAPBgNVBAcMCEdvb2R5ZWFyMQwwCgYDVQQKDANEREYxDDAKBgNV
BAsMA0RldjESMBAGA1UEAwwJbG9jYWxob3N0MSQwIgYJKoZIhvcNAQkBFhVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwg
Z8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMeCyNZbCTZphHQfB5g8FrgBq1RYzV7ikVw/pVGkz8gx3l3A99s8Wt
A4mRAeb6n0vTR9yNBOekW4nYOiEOq//YTi/frI1kz0QbEH1s2cI5nFButabD3PYGxUSuapbc+AS7+Pklr0TDI4MRz
PPkkTp4wlORQ/a6CfVsNr/mVgL2CfAgMBAAGjgZkwgZYwCQYDVR0TBAIwADAnBglghkgBhvhCAQ0EGhYYRk9SIFRF
U1RJTkcgUFVSUE9TRSBPTkxZMB0GA1UdDgQWBBSA95QIMyBAHRsd0R4s7C3BreFrsDAfBgNVHSMEGDAWgBThVMeX3
wrCv6lfeF47CyvkSBe9xjAgBgNVHREEGTAXgRVsb2NhbGhvc3RAZXhhbXBsZS5vcmcwDQYJKoZIhvcNAQEFBQADgY
EAtRUp7fAxU/E6JD2Kj/+CTWqu8Elx13S0TxoIqv3gMoBW0ehyzEKjJi0bb1gUxO7n1SmOESp5sE3jGTnh0GtYV0D
219z/09n90cd/imAEhknJlayyd0SjpnaL9JUd8uYxJexy8TJ2sMhsGAZ6EMTZCfT9m07XduxjsmDz0hlSGV0=
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </md:KeyDescriptor>
  <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://localhost:8993/logout"/>
  <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
```

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

Related Configuration

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

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

Limitations

The internal Identity Provider solution should be used in favor of any external solutions until the IdP Service Provider fully satisfies the SAML 2.0 Web SSO profile.

CAS Authentication

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

Username and Password

Sample – Successful login

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

Sample – Failed login

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

PKI Certificate

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

Sample – Successful login

```
2013-04-24 15:13:14,388 INFO
[org.jasig.cas.adaptors.x509.authentication.handler.support.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.
```

2.3.6. CAS SSO Configuration

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

General Server Setup and Configuration

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

Install using DDF CAS WAR

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

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

1. Download and unzip Tomcat Distribution. The installation location is referred to as `<TOMCAT_INSTALLATION_DIR>`.

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

2. Clone https://github.com/codice/cas-distribution to a convenient location. This folder will be referred to as `cas-distribution`.

3. Set up Keystores and enable SSL. There are sample configurations located within the `security-cas-server-webapp` project.

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

   Linux

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

   Windows

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

   b. Copy server.xml (`cas-distribution/src/main/resources/tomcat/conf`) to `<TOMCAT_INSTALLATION_DIR>/conf`

   Linux

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

   Windows

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

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

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

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

4. Start Tomcat.

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

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

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

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

uncomment the following in server.xml:

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

then comment out:

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

5. Deploy the DDF CAS WAR to Tomcat.

   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.

2.3.7. Configuring CAS for LDAP

Install and Configure LDAP

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

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

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

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

Add spring-ldap-1.2.1_1.jar to CAS

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

Modify developerConfigContext.xml

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

   deployerConfigContext.xml

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

       <!-- other property definitions -->

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

               <!-- other bean definitions -->

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

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

3. In <OZONE-WIDGET-FRAMEWORK>/apache-tomcat-<VERSION>/webapps/cas/WEB-INF/deployerConfigContext.xml, add the contextSource bean stanza to the beans stanza:

deployerConfigContext.xml

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

Configure Ozone

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

1. Modify OWFsecurityContext.xml

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

    OWFsecurityContext.xml

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

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

    OWFsecurityContext.xml

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

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

    OWFsecurityContext.xml

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

2. Modify LdapBeans.xml

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

LdapBeans.xml

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

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

LdapBeans.xml

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

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

LdapBeans.xml

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

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

LdapBeans.xml

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

3. Modify OWFCASBeans.xml

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

   OWFCasBeans.xml

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

2.3.8. Configuring CAS for X509 User Certificates

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

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

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

Configure Web Flow

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
`X509CertificateCredentialsToDistinguishedNamePrincipalResolver`	Retrieve the complete distinguished name and use that as the identifier.
`X509CertificateCredentialsToIdentifierPrincipalResolver`	Transform some subset of the identifier into the ID for the principal.
`X509CertificateCredentialsToSerialNumberPrincipalResolver`	Use the unique serial number of the certificate.
`X509CertificateCredentialsToSerialNumberAndIssuerDNPrincipalResolver`	Create a most-likely globally unique reference to this certificate as a DN-like entry, using the CA name and the unique serial number of the certificate for that CA.

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

Default Certificates

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

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

External Links

For more information on CAS configuration options and what each setting means, refer to their documentation page.

2.3.9. Certificate Management

DDF uses certificates in two ways:

i. Transmitting and receive encrypted messages.

ii. Performing authentication of an incoming user request.

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

Default Certificates

DDF comes with a default keystore that contains certificates.

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

Host
`Keystore`
`Truststore`
`Configuration Location`
`Usage`
`localhost`
`serverKeystore.jks`

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

Configuring a Java Keystore for Secure Communications

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

Creating a Client Keystore

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

i. Obtain a personal ECA cert (client certificate)

1. To do this, open Internet Explorer > Tools > Options.
2. Select the Content tab.
3. Click Certificates
4. Select the Personal tab.
5. Select the certificate needed to export. There should be the one without a "Friendly Name" and it is not the "Encryption Cert").
6. Click Export.
7. Follow Certificate Export Wizard.
8. When a prompt requests to export the private key, select Yes.

ii. Add a cert to a new java keystore, replacing cert with the name of the PKCS12 Keystore needed to

convert, and replace MY_KEYSTORE.jks with the desired name of the Java Keystore:

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

1. The command prompts for two passwords:
    1. Input keystore passphrase is the passphrase that is used to protect cert.p12
    2. Output keystore passphrase is the passphrase that is set for the new java keystore serverKeystore.jks
2. It is recommended that the private key password be the same as the keystore password due to limitations in java.
    1. Run the following command to determine the alias name of the added current entry. It is listed after "Alias Name:"

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

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

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

a. When prompted for a password, use the same password used when the keystore was created.

b. Delete the original alias

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

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

Creating a Truststore

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

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

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

1. Enter in a keystore password when prompted.

Adding a certificate to an existing Keystore

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

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

1. Enter in the keystore password if prompted.

Updating Truststore and Keystore information

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

Locations

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

Certificate Configuration Management

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

Certificate Revocation Configuration

Enabling Revocation

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

i. Place the CRL in <DDF.home>/etc/keystores.

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

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

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

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

Add Revocation to a Web Context

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

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

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

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

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

Adding Revocation to a new Endpoint

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

This guide assumes that the endpoint being created uses CXF and is being started via Blueprint from inside the OSGi container. If other tools are being used the configuration may differ.

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

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

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

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

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

Verifying Revocation is taking place

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

```
11:48:00,016 | WARN  | tp2085517656-302 | WSS4JInInterceptor                |
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
```

Certificate File Management

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

The following tools are used:

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

General Certificates

Create a CA Key and Certificate

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

1. Create a key pair.
   `$> openssl genrsa -aes128 -out root-ca.key 1024`
2. Use the key to sign the CA certificate.
   `$> openssl req -new -x509 -days 3650 -key root-ca.key -out root-ca.crt`

Using the CA to Sign Certificates

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

1. Generate a private key and a Certificate Signing Request (CSR).
   `$> openssl req -newkey rsa:1024 -keyout tokenissuer.key -out tokenissuer.req`

2. Sign the certificate by the CA.
   `$> openssl ca -out tokenissuer.crt -infiles tokenissuer.req`

Java Keystore (JKS)

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

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

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

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

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

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

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

Import into a Java Keystore (JKS)

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

1. Put the private key and the certificate into one file.
   `$> cat tokenissuer.crt >> tokenissuer.key`

2. Put the private key and the certificate in a PKCS12 keystore.
   `$> openssl pkcs12 -export -in tokenissuer.key -out tokenissuer.p12`

3. Import the PKCS12 keystore into a JKS.
   `$> keytool -importkeystore -srckeystore tokenissuer.p12 -destkeystore stsKeystore.jks -srcstoretype pkcs12 -alias 1`

4. Change the alias.
   `$> keytool -changealias -alias 1 -destalias tokenissuer`

Certificate Revocation List (CRL)

Creating a Certificate Revocation List (CRL)

1. Using the CA create in the above steps, create a CRL in which the `tokenissuer  s certificate is valid.`
   `$> openssl ca -gencrl -out crl-tokenissuer-valid.pem`

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

```
$> openssl ca -revoke tokenissuer.crt

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

Viewing a CRL

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

Enabling SSL for Sources

Updating Key Store / Trust Store for all sources

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

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

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

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

Updating Key Store / Trust Store via the Admin Console

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

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

NOTE *Reference*

- Oracle Keytool Documentation

Enabling SSL/TLS for Services

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

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

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

Table 7. Pax Web Configuration Settings

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

Example `.cfg` file:

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

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

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


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

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

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

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

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

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

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

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

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

WARNING

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

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

NOTE

Reference

Configuring a Java Keystore for Secure Communications

Additional Pax-Web SSL configuration info: SSL Configuration

2.3.10. Updating System Users

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

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

```
{
    "admin" : {
        "test" : "testValue",
        "test1" : [ "testing1", "testing2", "testing3" ]
    },
    "localhost" : {

    },
    ".*host.*" : {
        "reg" : "ex"
    }
}
```

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

WARNING	It is possible for a regex in `users.attributes` to match users as well as a system, so verify that the regex pattern's scope will not be too great when using this feature.
WARNING	If your data will contain security markings, and these markings are being parsed out into the METACARD.security attribute via a PolicyPlugin, then the FQDN user **MUST** be updated with attributes that would grant the privileges to ingest that data. Failure to update the FQDN user with sufficient attributes will result in an error being returned for any ingest request.

2.3.11. Encryption Service

Encryption Command

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

Below is an example of the security:encrypted command used to encrypt the plain text `myPasswordToEncrypt`. The output is the encrypted value.

```
ddf@local>security:encrypt myPasswordToEncrypt

ddf@local>bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

2.3.12. Catalog Operation Authorization

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

Configuring Operation Authorization

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

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

2.3.13. Catalog Filtering

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

How Filtering Works

Each metacard result can contain security attributes that are pulled from the metadata record after being processed by a `PolicyPlugin` that populates this attribute. The security attribute is a Map containing a set of keys that map to lists of values. The metacard is then processed by a filter 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 the metacard eventually relies on the installed PDP (`feature:install security-pdp-authz`). The PDP that is being used returns a decision, and the metacard will either be filtered or allowed to pass through.

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

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

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

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

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

In the above example, the user's claims are represented very simply and are similar to how they would actually appear in a SAML 2 assertion. Each of these user (or subject) claims will be converted to a `KeyValuePermission` object. These permission objects will be implied against the permission object generated from the 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.

Filter Policies

The procedure for setting up a policy differs depending on whether that policy is to be used internally or by the external XACML processing engine. Setting up an internal policy is as follows:

1. Open the Admin Console at https://localhost:8993/admin
2. Click the DDF Security application tile
3. Click the **Configuration** tab
4. Click on the **Security AuthZ Realm** configuration.
5. Add any attribute mappings necessary to map between subject attributes and the attributes to be asserted.
    a. For example, the above example would require two Match All mappings of `subjectAttribute1=assertedAttribute1` and `subjectAttribute2=assertedAttribute2`
    b. Match One mappings would contain subjectAttribute3=assertedAttribute3` and `subjectAttribute4=assertedAttribute4`.

With the `security-pdp-authz` feature configured in this way, the above Metacard would be displayed to the user. Note that this particular configuration would not require any XACML rules to be present. All

of the attributes can be matched internally and there is no reason to call out to the external XACML processing engine. For more complex decisions, it might be necessary to write a XACML policy to handle certain attributes.

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

Catalog Filter Policy Plugins

Several PolicyPlugins for catalog filtering exist currently: **Metacard Attribute Security Policy Plugin** and **XML Attribute Security Policy Plugin**. These PolicyPlugin implementations allow an administrator to easily add filtering capabilities to some standard Metacard types for all DDF Catalog operations. These plugins will place policy information on the Metacard itself that allows the FilterPlugin to restrict unauthorized users from viewing content they are not allowed to view.

The **XML Attribute Security Policy Plugin** will parse XML metadata contained within a metacard for security attributes on any number of XML elements in the metadata. The configuration for the plugin contains one field for setting the XML elements that will be parsed for security attributes and the other two configurations contain the XML attributes that will be pulled off of those elements. The **Security Attributes (union)** field will compute the union of values for each attribute defined. While the **Security Attributes (intersection)** field will computer the intersection of values for each attribute defined.

The **Metacard Attribute Security Policy Plugin** will pull attributes directly off of a metacard and combine these attributes into a security field for the metacard. This plugin assumes that the pertinent information has already been parsed out of the metadata and placed directly on the metacard itself. The plugin supports mapping attributes from their names on the metacard to a different security policy name. Attributes can be mapped by union or intersection. E.g. A union mapping of `src1=dest` and `src2=dest` will result in `dest` containing the union of all attributes from `src1` and `src2`; an intersection mapping of `src1=dest` and `src2=dest` will result in `dest` containing the intersection of common attributes from `src1` and `src2`.

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

Creating a XACML Policy

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

through a SOAP endpoint.

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

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

XACML Policy Attributes

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

XACML Policy Subject

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

XACML Policy Resource

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

Example Requests and Responses

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

Policy

This is the sample policy that was used for the following sample request and responses. The policy was made to handle the following actions: `filter`, `query`, and `LocalSiteName`. The filter action is used to compare subject's `SUBJECT_ACCESS` attributes to a 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>
```

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>
```

Using a XACML Policy

To use a XACML policy, do the following:

- Copy the XACML policy into the `<INSTALL_DIRECTORY>/etc/pdp/policies` directory.

2.3.14. Expansion Service

The expansion service defines rulesets to map metacard attributes and user attributes to more complete sets of values. For example if a user has an attribute "alphabet" that contained the value "full", the expansion service can be configured to expand the "full" value out to ["a","b","c",...].

Configuring Expansion Service

To use the expansion service, modify the following two files within the `<INSTALLATION_HOME>/etc`

directory:

- `<INSTALLATION_HOME>/etc/pdp/ddf-metacard-attribute-ruleset.cfg`
- `<INSTALLATION_HOME>/etc/pdp/ddf-user-attribute-ruleset.cfg`

Within these files, the following configuration details will be defined.

Expansion Service Instances and Configuration

It is expected that multiple instances of the expansion service will be running at the same time. Each instance of the service defines a unique property that is useful for retrieving specific instances of the expansion service. The following table lists the two pre-defined instances used by DDF for expanding user attributes and 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
```

Table 8. Expansion Commands

Title	Namespace	Description
DDF::Security::Expansion::Commands	security	The expansion commands provide detailed information about the expansion rules in place and the ability to see the results of expanding specific values against the active rule set.

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

Expansion Command Examples and Explanation

`security:expansions`

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

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

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

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

`security:expand`

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

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

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

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

2.3.15. Configure WSS Using Standalone Servers

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

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

Server	`Keystore File`	Comments
DDF	`serverKeystore.jks`	Keystore used for SSL/TLS connections.

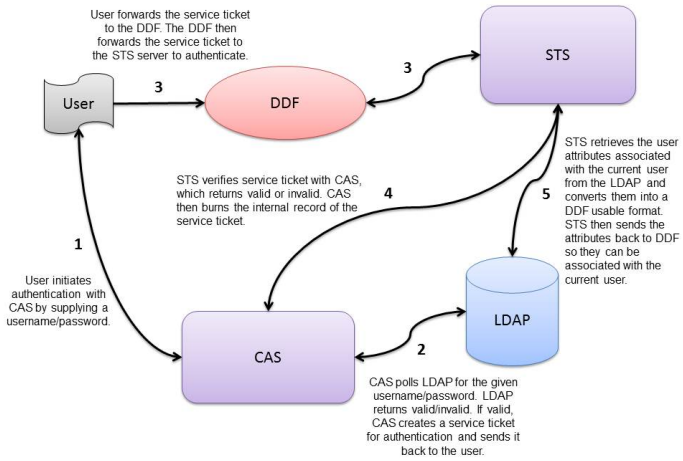


Figure 1. Login Authentication Scheme

2.3.16. Authentication Components

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

LDAP

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

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

STS

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

1. Verify that the `serverKeystores.jks` file in `/etc/keystores` trusts the hostnames used in your environment (the hostnames of LDAP, and any DDF users that make use of this STS server).
2. Start the distribution.
3. Run the installer at https://localhost:8993/admin (if needed)
4. After the installer completes, click the DDF Security application tile

5. Click the **Features** tab
6. Install `security-sts-ldaplogin` and `security-sts-ldapclaimshandler` features by clicking the **Play** button next to each
7. Open the Admin Console as an administrator (https://localhost:8993/admin).
8. Click the **DDF Security** application tile
9. Click the **Configuration** tab
10. Open the **Security STS LDAP Login** configuration.
11. Verify that the **LDAP URL**, **LDAP Bind User DN**, and **LDAP Bind User Password** fields match your LDAP server's information. The default DDF LDAP settings will match up with the default settings of the OpenDJ embedded LDAP server. If not using the embedded LDAP server, change these values to map to the location and settings of the LDAP server being used.
12. Select the **Save changes** button if changes were made.
13. Open the **Security STS LDAP and Roles Claims Handler** configuration.
14. Populate the same URL, user, and password fields with your LDAP server information.
15. Select the **Save Changes** button.

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

2.3.17. Configuring DDF Authentication Scheme

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

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

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

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

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

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

Enable SSL for Clients

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

Property	Sample Value	Description
`javax.net.ssl.trustStore`	`etc/keystores/serverTruststore.jks`	The java keystore that contains the trusted public certificates for Certificate Authorities (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 for signing, encryption, and SSL/TLS.
`javax.net.ssl.keyStorePassword`	`changeit`	The password for the keystore listed above
`javax.net.ssl.keyStoreType`	`jks`	The type of keystore
`https.cipherSuites`	`TLS_DHE_RSA_WITH_AES_128_CBC_SHA`, `TLS_DHE_RSA_WITH_AES_128_CBC_SHA`, `TLS_DHE_DSS_WITH_AES_128_CBC_SHA`, `TLS_RSA_WITH_AES_128_CBC_SHA`	The cipher suites that are supported when making outbound HTTPS connections

Property	Sample Value	Description
`https.protocols`	`TLSv1.1`,`TLSv1.2`	The protocols that are supported when making outbound HTTPS connections

NOTE `<INSTALLATION_HOME>`
DDF is installed in the `<INSTALLATION_HOME>` directory.

2.3.18. Auditing

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

Configuring Audit Plugin

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

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

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

2.3.19. Assuring Authenticity of Bundles and Applications

DDF Artifacts in the JAR file format (such as bundles or DDF applications packaged as KAR files) can be signed and verified using the tools included as part of the Java Runtime Environment.

Prerequisites

To work with Java signatures, a keystore/truststore is required. For the purposes of this example we'll sign and validate using a self signed certificate which can be generated with the keytool utility. In production a certificate issued from a trusted Certificate Authority should be used.

Additional documentation on keytool can be found at Keytool home.

Using keytool to generate a self-signed certificate keystore

```
~ $ keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass
password -validity 360 -keysize 2048
What is your first and last name?
  [Unknown]:  Nick Fury
What is the name of your organizational unit?
  [Unknown]:  Marvel
What is the name of your organization?
  [Unknown]:  SHIELD
What is the name of your City or Locality?
  [Unknown]:  New York
What is the name of your State or Province?
  [Unknown]:  NY
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Nick Fury, OU=SHIELD, O=Marvel, L="New  York", ST=NY, C=US correct?
  [no]:  yes
Enter key password for <selfsigned>
    (RETURN if same as keystore password):
Re-enter new password:
```

Signing a JAR/KAR

Once a keystore is available, the JAR can be signed using the `jarsigner` tool.

Additional documentation on jarsigner can be found at Jarsigner.

Using jarsigner to sign a KAR

```
~ $ jarsigner -keystore keystore.jks -keypass shield -storepass password  catalog-app-
2.5.1.kar selfsigned
```

Verifying a JAR/KAR

The jarsigner utility is also used to verify a signature in a JAR-formatted file.

Using jarsigner to verify a file

```
~ $ jarsigner -verify -verbose -keystore keystore.jks catalog-app-2.5.1.kar
        9447 Mon Oct 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.

2.4. Running DDF

Follow the below steps to start and stop DDF.

2.4.1. Starting DDF

***NIX**

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

```
INSTALLATION_HOME/bin/ddf
```

Windows

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

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

2.4.2. Stop DDF

There are two options:

- Call shutdown from the console:

Shut down with a prompt

```
ddf@local>shutdown
```

Force Shutdown without prompt

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

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

**NIX*

```
DDF_INSTALL/bin/stop
```

Windows

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

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 shortcut from the command line console.

2.4.3. Automatic Start on System Boot

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

1. Create the Karaf wrapper.

   Within the DDF console

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

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

   Change:

INSTALLATION_HOME/bin/ddf-service

```
#RUN_AS_USER=
```

to:

INSTALLATION_HOME/bin/ddf-service

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

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

   a. Add the setting for PermGen space under the JVM Parameters section.

   b. Update heap space to 2048.

   INSTALLATION_HOME/etc/ddf-wrapper.conf

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

   #Update the following:
   wrapper.java.maxmemory=2048
   ```

4. Set the `DDF_HOME` property.

   INSTALLATION_HOME/etc/ddf-wrapper.conf

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

5. Install the wrapper startup/shutdown scripts.

   Windows

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

   ```
   INSTALLATION_HOME/bin/ddf-service.bat install
   ```

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

Redhat

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

Ubuntu

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

Solaris

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

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

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

Solaris DDF_INSTALL/bin/ddf-service

```
pidtest=`$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.

2.4.4. Managing Applications from Admin Console

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

Activating / Deactivating Applications

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

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

Removing Applications

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

Upgrading Applications

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

System Settings Tab

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

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

2.4.5. Federation

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

2.4.6. Console Commands

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

Access the System Console

The Command Line Console is the console that is available to the user when the distribution is started manually. It may also be accessed by using the `bin/client.bat` or `bin/client.sh` scripts. For more information on how to use the `client` scripts or how to remote into the the shell console, see Using Remote Instances.

Example Commands

View Bundle Status

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

View Installed Features

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

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

2.4.7. 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` 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      catalog:validate
```

Table 9. 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.

Command	Description
`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 `CatalogProvider`s 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.
`validate`	Validates an XML file against all installed validators and prints out human readable errors and warnings.

Available System Console Commands

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

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

System Console Command Help

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

Example Help

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

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

`catalog:dump` Options

The `catalog:dump` command was extended in DDF version 2.5.0 to provide selective export of metacards based on date ranges. The `--created-after` and `--created-before` options allow filtering on the date and time that the 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
```

Application Commands

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

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

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

Listing Available System Console Commands

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

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

Command Descriptions

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

Command	Syntax	Description
`status`	`app:status appName`	Shows status of an application. Gives information on the current state, features within the application, what required features are not started and what required bundles are not started.
`tree`	`app:tree`	Creates a hierarchy tree of all of the applications.

Command Usage Examples

Listing all applications

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

[...]
```

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

Getting status for a specific application

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

Current State is: ACTIVE

Features Located within this Application:
    catalog-security-filter
    catalog-transformer-resource
    catalog-rest-endpoint
    abdera
    catalog-transformer-xml
    catalog-transformer-thumbnail
    catalog-transformer-metadata
    catalog-transformer-xsltengine
    catalog-core-fanoutframework
    catalog-transformer-tika
    catalog-core-api
    catalog-opensearch-source
    catalog-plugin-federationreplication
    catalog-opensearch-endpoint
    catalog-schematron-plugin
    catalog-transformer-geoformatter
    catalog-transformer-atom
    catalog-core-sourcemetricsplugin
    catalog-core-metricsplugin
    catalog-app
    catalog-transformer-json
    catalog-core-standardframework
    catalog-core

Required Features Not Started
    NONE

Required Bundles Not Started
    NONE
```

Application in Failure State

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

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

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

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

Current State is: FAILED

Features Located within this Application:
    catalog-security-filter
    catalog-transformer-resource
    catalog-rest-endpoint
    abdera
    catalog-transformer-xml
    catalog-transformer-thumbnail
    catalog-transformer-metadata
    catalog-transformer-xsltengine
    catalog-core-fanoutframework
    catalog-transformer-tika
    catalog-core-api
    catalog-opensearch-source
    catalog-plugin-federationreplication
    catalog-opensearch-endpoint
    catalog-schematron-plugin
    catalog-transformer-geoformatter
    catalog-transformer-atom
    catalog-core-sourcemetricsplugin
    catalog-core-metricsplugin
    catalog-app
    catalog-transformer-json
    catalog-core-standardframework
    catalog-core

Required Features Not Started
    NONE

Required Bundles Not Started
    [261]   catalog-opensearch-endpoint
```

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

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

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

2.4.8. Command Scheduler

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

Using the Command Scheduler

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

Schedule a Command

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

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

Updating a Scheduled Command

1. Navigate to the **Admin Console**.
2. Click on the **DDF Platform** application.
3. Click on the **Configuration** tab.
4. Under the **Platform Command Scheduler** configuration are all the scheduled commands. Scheduled commands have the following syntax `ddf.platform.scheduler.Command.{GUID}` such as `ddf.platform.scheduler.Command.4d60c917-003a-42e8-9367-1da0f822ca6e`.
5. Find the desired configuration to modify and update either the **Command** text field or the **Interval In Seconds** field or both.
6. Click **Save changes**. Once the Save button has been clicked, the command will be executed immediately. Its next scheduled execution happens after the time specified in Interval In Seconds and repeats indefinitely until the container is shutdown or the Scheduled Command is deleted.

Command Output

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

Sample Command Output in the Log

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

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

2.4.9. 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.
```

2.4.10. Platform Commands

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

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

Commands

Command Descriptions

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

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

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.

2.4.11. 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.

2.4.12. CQL Syntax

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

Examples

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

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

2.4.13. Ingesting Data

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

File types supported

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

2.4.14. 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 Catalog application contains a Directory Monitor feature that allows files placed in a single directory to be monitored and ingested automatically. For more information about configuring a directory to be monitored, consult Directory Monitor.

Using Directory Monitor

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

Medium

External Methods

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

Windows Example

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

+ .*NIX Example

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

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

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

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

+ .Sample Response

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

+ . Verify the entry was successfully ingested by entering in a browser the URL returned in the POST response's HTTP header. For instance in our example, it was `/services/catalog/44dc84da101c4f9d9f751e38d9c4d97b`. This should display the catalog entry in XML within the browser. . Verify the catalog entry exists by executing a query via the OpenSearch endpoint. . Enter the following URL in a browser /services/catalog/query?q=ddf. A single result, in Atom format, should be returned.

Verifying Ingest

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

    a. Send a GET command to read the content from the content repository using the Content REST endpoint. This can be done using `cURL` command below. Note that the GUID will be different for each ingest. The GUID can be determined by going to the `<DISTRIBUTION_INSTALL_DIR>/content/store` directory and copying the sub-directory in this folder (there should only be one).

Windows Example

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

**NIX Example*

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

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

Advanced (more records, automated ingest)

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

2.4.15. Removing Expired Records from Catalog

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

Universal Expired Records Removal

Manual Removal

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

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

When prompted, type yes to remove all expired records.

TIP

For help on the removeall command, execute

`help catalog:removeall`

Automated Removal

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

`catalog:removeall --force --expired`

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

Explanation of Command to Remove Expired Records

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

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

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

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

Non-Universal or Catalog Specific Removal

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

Automatic Catalog Backup

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

Admin Console DDF Catalog Configuration Backup Post-Ingest Plugin.

2.4.16. Metrics Reporting

Metrics are available in several formats and levels of detail.

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

1. Select **DDF-Platform**
2. Select **Metrics** tab
3. For individual metrics, choose the format desired from the desired timeframe column
    a. PNG
    b. CSV

c. XLS

4. For a detailed report of all metrics, at the bottom of the page are selectors to choose time frame and summary level. A report is generated in *xls* format.

2.4.17. Monitoring DDF

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

Managing Logging

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

Configuring Logging

Edit the configuration file `[ddf_install_dir]/etc/org.ops4j.pax.logging.cfg]`

DDF log file

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

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

Controlling log level

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

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

Controlling the size of the log file

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

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

Number of backup log files to keep

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

```
log4j.appender.out.maxBackupIndex=10
```

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

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

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

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

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

External Resources

Other appenders can be selected and configured.

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

2.4.18. Enabling HTTP Access Logging

Configuring

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

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

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

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

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

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

External Resources

Advanced Jetty Configuration Jetty Request Log Tutorial

Using the LogViewer

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

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

The LogViewer displays the most recent 500 log messages by default, but will grow to a maximum of 5000 messages.

Log events can be filtered by:

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

WARNING	It is not recommended to use the LogViewer if the system logger is set to a low reporting level such as `TRACE`. The volume of messages logged will exceed the polling rate, and incoming logs may be missed. The actual logs being polled by the LogViewer can still be accessed at `<INSTALL_HOME>/data/log`

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

2.5. Troubleshooting DDF

Table 10. General Troubleshooting

Issue	Solution
Ingest more than 200,000 data files stored NFS shares may cause Java Heap Space error (Linux-only issue).	This is an NFS bug where it creates duplicate entries for some files when doing a file list. Depend on the OS, some Linux machines can handle the bug better and able get a list of files but get an incorrect number of files. Others would have a Java Heap Space error because there are too many file to list.
Ingest millions of complicated data into Solr can cause Java heap space error.	Complicated data has spatial types and large text.
Ingest serialized data file with scientific notation in WKT string causes RuntimeException.	WKT string with scientific notation such as POINT (-34.8932113039107 -4.77974239601E-5) won't ingest. This occurs with serialized data format only.
Exception Starting DDF (Windows) An exception is sometimes thrown starting DDF on a Windows machine (x86). If using an unsupported terminal, `java.lang.NoClassDefFoundError: Could not initialize class org.fusesource.jansi.internal.Kernel32` is thrown.	Install missing Windows libraries. Some Windows platforms are missing libraries that are required by DDF. These libraries are provided by the Microsoft Visual C++ 2008 Redistributable Package x64.

CXF BusException
 	
The following exception is thrown: `org.apache.cxf.BusException: No conduit initiator`
	Restart DDF.
	. Shut down DDF:
	
	`ddf@local>shutdown`
	. Start up DDF:
	`./ddf`

Issue	Solution

Distribution Will Not Start
 	
DDF will not start when calling the start script defined during installation.
	Complete the following procedure.
	
	. Verify that Java is correctly installed.
	+
	`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`

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

3. Managing DDF Admin

Version: 2.9.1

The DDF Admin Application contains components that are responsible for the installation and configuration DDF applications.

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

The Admin application contains an application service that handles all operations that are performed on applications. This includes adding, removing, starting, stopping, and showing status.

3.1. Installing DDF Admin

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

3.1.1. Configurable Properties DDF Admin

The following configuration properties can be changed under the Configuration tab under Admin UI

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

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

3.1.2. Defining Platform Settings

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

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

3.2. Console Commands

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

3.2.1. Application Service Interfaces

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

3.2.2. Installing and Uninstalling

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

3.2.3. Configuring

None.

3.3. Administrative User Interface

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

3.3.1. Modules

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

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

3.3.2. Included Modules

- Installer Module
- Configuration Module

Installer Module

The application installer module enables a user to install and remove applications. Each application includes a features file that provides a description of the application and a list of the dependencies

required to successfully run that application. The installer reads the features file and presents the applications in a manner that allows the administrator to visualize these dependencies. As applications are selected or deselected, the corresponding dependent applications are selected or deselected as necessary.

Set Up the Installer Module

1. Install the module if it is not already pre-installed.
   `feature:install admin-modules-installer`
2. Open a web browser and navigate to the Installation page.
   `http://DDF_HOST:DDF_PORT/admin`
   - Adding the `?dev=true` query string will auto generate the certificates
     `http://DDF_HOST:DDF_PORT/admin/index.html?dev=true`
3. Log in with the default username of "admin" (no quotes) and the default password of "admin" (no quotes).
4. Select the Setup tab if not already selected.

Example Screenshots

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

Welcome Page

Anonymous Claims page

Installation Profile Page

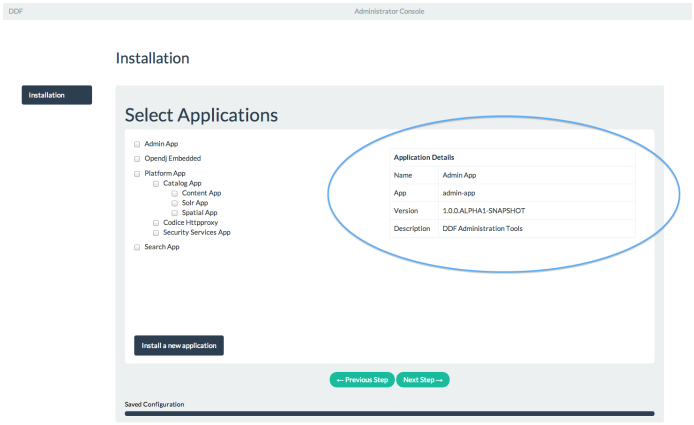
IMPORTANT	Do NOT deselect/uninstall the Platform App or the Admin App. Doing so will disable the use of this installer and the ability to install/uninstall other applications.

- Installation Profile Page
  - When a profile is selected, it will auto select applications on the Select Application Page and install them automatically.
  - If choose to customize a profile, you will be given the options to manually selected the applications on the Select Application Page.
- In the Select applications to install page, hover over each application to view additional details about the application.

New applications can be added and existing applications can be upgraded using the Applications Module.

- When an application is selected, dependent applications will automatically be selected.
- When an application is unselected, dependent applications will automatically be unselected.

3.3.3. Custom Installation



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

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.

General Configuration Page

General Configuration Page (Certificates)

NOTE

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

Final Page

Shutdown Page

-

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

3.3.4. Configuration Module

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

Set Up the Module

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

`http://DDF_HOST:DDF_PORT/admin`

1. Select the Configurations tab if not already selected.

Configurations Tab

3.4. Admin Console Access Control

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

3.4.1. Restricting DDF Access

1. See the documentation for your specific security infrastructure to configure users, roles, and groups.
2. Select the Web Context Policy Manager.
   a. A dialogue will pop up that allows you to edit DDF access restrictions.
   b. Once you have configured your realms in your security infrastructure, you can associate them with DDF contexts.
   c. If your infrastructure supports multiple authentication methods, they may be specified on a per-context basis.
   d. Role requirements may be enforced by configuring the required attributes for a given context.
   e. The whitelist allows child contexts to be excluded from the authentication constraints of their parents.

3.4.2. Restricting DDF Feature, App, Service, and Configuration Access

1. See the documentation for your specific security infrastructure to configure users, roles, and groups.

2. Select the Admin Configuration Policy under the DDF Admin Admin App

3. To add a feature or app permission:

   a. Add a new field to "Feature and App Permissions" in the format of: `feature name/app name = "attribute name=attribute value","attribute name2=attribute value2",`

   b. For example, to restrict access of any user without an admin role to the catalog-app: `catalog-app = "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=admin",`

4. To add a configuration permission:

   a. Add a new field to "Configuration Permissions" in the format of: `configuration id = "attribute name=attribute value","attribute name2=attribute value2",`

   b. For example, to restrict access of any user without an admin role to the Web Context Policy Manager: `org.codice.ddf.security.policy.context.impl.PolicyManager="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=admin"`

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

3.4.3. LDAP Admin Role Configuration

The admin role will default to `system-admin`. This can be configured to work with an external LDAP with a few minor changes.

3.4.4. Update the admin role in `INSTALL_HOME/etc/users.properties`

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

Example `user.properties` *entries:*

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

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

3.4.5. Update the web context policy to point to the new admin role

1. Open DDF Security in the Admin Console

2. Select the Configuration tab and open Web Context Policy Manager

3. Update the entries under 'Required Attributes' to set the new admin role

Web Context Policy Manager

4. Managing DDF Catalog

Version: 2.9.1

4.1. Installing DDF Catalog

4.1.1. Prerequisites

Before the DDF Catalog application can be installed:

- the DDF Platform application must be installed

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

4.2. Configuring DDF Catalog

Table 11. Available Configurations

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

Table 12. Backup Post-Ingest Plugin

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

Table 13. Catalog OpenSearch Federated Source

Name	Property	Type	Description	Default Value	Required
Source Name	`shortname`	String		DDF-OS	Yes
OpenSearch service URL	`endpointUrl`	String	The OpenSearch endpoint URL or DDF's OpenSearch endpoint (https://localhost:8993/services/catalog/query)	${org.codice.ddf.system.protocol}${org.codice.ddf.system.hostname}:${org.codice.ddf.system.port}${org.codice.ddf.system.rootContext}/catalog/query	Yes
Username	`username`	String	Username to use with HTTP Basic Authentication. This auth info will overwrite any federated auth info. Only set this if the OpenSearch endpoint requires basic authentication.		No

Name	Property	Type	Description	Default Value	Required
Password	`password`	Password	Password to use with HTTP Basic Authentication. This auth info will overwrite any federated auth info. Only set this if the OpenSearch endpoint requires basic authentication.		No
OpenSearch query parameters	`parameters`	String	Query parameters to use with the OpenSearch connection.	q,src,mr,start,count,mt,dn,lat,lon,radius,bbox,polygon,dtstart,dtend,dateName,filter,sort	Yes
Always perform local query	`localQueryOnly`	Boolean	When federating with other DDFs, keep this checked. If checked, this source performs a local query on the remote site (by setting src=local in endpoint URL), as opposed to an enterprise search.	true	Yes
Convert to BBox	`shouldConvertToBBox`	Boolean	Converts Polygon and Point-Radius searches to a Bounding Box for compatability with older interfaces. Generated bounding box is a very rough representation of the input geometry.	true	Yes

Table 14. Catalog Policy Plugin

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

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

Table 15. Catalog Standard Framework

Name	Property	Type	Description	Default Value	Required
Enable Fanout Proxy	`fanoutEnabled`	When enabled the Framework acts as a proxy, federating requests to all available sources. All requests are executed as federated queries and resource retrievals, allowing the framework to be the sole component exposing the functionality of all of its Federated Sources.	Boolean	true	No

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

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

Table 16. Metacard Attribute Security Policy Plugin

Name	Property	Type	Description	Default Value	Required
Metacard Attributes:	`metacardAttributes`	String	Metacard attributes that will be collected and mapped to security information. Example: `security.classification=classification`.		No

Table 17. Schematron Validation Services

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

Table 18. XML Attribute Security Policy Plugin

Name	Property	Type	Description	Default Value	Required
XML Elements:	`xmlElements`	String	XML elements within the metadata that will be searched for security attributes. If these elements contain matching attributes, the values of the attributes will be combined.		true
Security Attributes (union):	`securityAttributeUnions`	String	Security Attributes. These attributes, if they exist on any of the XML elements listed above, will have their values extracted and the union of all of the values will be saved to the metacard. For example: if element1 and element2 both contain the attribute 'attr' and that attribute has values X,Y and X,Z, respectively, then the final result will be the union of those values: X,Y,Z. The X,Y,Z value will be the value that is placed within the security attribute on the metacard.		false

Name	Property	Type	Description	Default Value	Required
Security Attributes (intersection):	`securityAttributeIntersections`	String and the intersection of all of the values will be saved to the metacard. For example: if element1 and element2 both contain the attribute 'attr' and that attribute has values X,Y and X,Z, respectively, then the final result will be the intersection of those values: X. The X value will be the value that is placed within the security attribute on the metacard.	Security Attributes. These attributes, if they exist on any of the XML elements listed above, will have their values extracted		false

Table 19. Xml Query Transformer

Name	Property	Type	Description	Default Value	Required
Parallel Marhsalling Threshold	`threshold`	Integer	Response size threshold above which marshalling is run in parallel	50	true

Table 20. Catalog Duplicate Validator

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

5. Managing DDF GeoWebCache

Version: 2.9.1

DDF GeoWebCache enables a server providing a tile cache and tile service aggregation. See (GeoWebCache) for more information. This application also provides an administrative plugin for the management of GeoWebCached layers. GeoWebCache also provides a user interface for previewing, truncating, or seeding layers at https://localhost:8993/geowebcache/.

5.1. Prerequisites for DDF GeoWebCache

none.

5.2. Installing DDF GeoWebCache

The DDF GeoWebCache application is **not** installed by default. To install:

- Navigate to the Admin Console.
- Select **Manage**.
- Select the application **Start** button. The application will move to **Active Applications** on startup.
- Select **Back** in Admin Console.

5.3. Configuring DDF GeoWebCache

DDF GeoWebCache can be configured to cache layers locally, using the following procedures.

5.3.1. Adding GeoWebCache Layers

- Navigate to the Admin Console.
- Select the DDF GeoWebCache Application.
- Select the **GeoWebCache Layers** tab.
- Click the **Add** button.
- Enter the data in the fields provided.
- If necessary, click the **Add** button to add additional MIME types.
- If necessary, click the **Add** button to add additional WMS Layer Names.

Table 21. Add Layer

Name	Property	Type	Description	Default Value	Required
Name		String			no
Mime Formats		String			yes
URL		URI			yes
WMS Layer Name		String			yes

5.3.2. Editing GeoWebCache Layers

- Navigate to the Admin Console.
- Select the DDF GeoWebCache application.
- Navigate to the **GeoWebCache Layers** tab.
- Click the **Name** field of the layer to edit.

5.3.3. Removing GeoWebCache Layers

- Click the **Delete** icon at the end of the row of the layer to be deleted.

5.3.4. Configuring GWC Disk Quota

Storage usage for a GeoWebCache server is managed by a `diskquota.xml` file with configuration details to prevent image-intensive data from filling the available storage.

To view the disk quota XML representative: https://localhost:8993/geowebcache/rest/diskquota.xml"

To update the disk quota, a client can post a new XML configuration: `curl -v -k -XPUT -H "Content-type: text/xml" -d @diskquota.xml "${secure_url}/geowebcache/rest/diskquota.xml"`

Example `diskquota.xml`

```
<gwcQuotaConfiguration>
  <enabled>true</enabled>
  <diskBlockSize>2048</diskBlockSize>
  <cacheCleanUpFrequency>5</cacheCleanUpFrequency>
  <cacheCleanUpUnits>SECONDS</cacheCleanUpUnits>
  <maxConcurrentCleanUps>5</maxConcurrentCleanUps>
  <globalExpirationPolicyName>LFU</globalExpirationPolicyName>
  <globalQuota>
    <value>100</value>
    <units>GiB</units>
  </globalQuota>
  <layerQuotas/>
</gwcQuotaConfiguration>
```

See Disk Quotas for more information on configuration options for disk quota.

5.3.5. Configuring the Standard Search UI for GeoWebCache

Add a new Imagery Provider in the Admin Console:

- Navigate to https://<localhost>:8993/admin.
- Select **Configuration** tab.
- Select **Standard Search UI** configuration.
- Click the **Add** button next to **Imagery Providers**
- Enter configuration for Imagery Provider in new textbox:
  - `{"type" "WMS" "url" "https://<HOSTNAME>:<PORT>/geowebcache/service/wms" "layers" ["states"] "parameters" {"FORMAT" "image/png"} "alpha" 0.5}`
- Set the Map Projection to `EPSG:900913` or `EPSG:4326`. (GeoWebCache supports either of these projections.)

NOTE

Currently, GeoWebCache only supports WMS 1.1.1 and below. If the version number is not specified in the imagery provider, DDF will default to version `1.3.0`, and OpenLayers will not project the image tiles properly. Thus, the version `1.1.1` must be specified when using `EPSG:4326` projections.

```
{"type" "WMS" "url" "https://<HOSTNAME:<PORT>/geowebcache/service/wms" "layers" ["states"] "parameters" {"FORMAT" "image/png" "VERSION" "1.1.1"} "alpha" 0.5}
```

6. Managing DDF Platform

Version: 2.9.1

The DDF Platform application is considered to be a core application of the distribution. The Platform application provides the fundamental building blocks that the distribution needs to run. These building blocks include subsets of:

- Karaf,
- CXF, and
- Camel.

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

6.1. Using DDF Platform Application

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

6.2. Installing and Uninstalling Platform

6.2.1. Prerequisites

None.

6.2.2. Installing DDF Platform

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

6.2.3. Configuring DDF Platform

Configuration	ID	Description
Landing Page	`org.codice.ddf.distribution.landing-page.properties`	Customize elements seen on landing page.

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

7. Managing DDF Registry

Version: 2.9.1

DDF Registry contains the base registry components, plugins, sources and interfaces needed for DDF to function as a registry connecting multiple nodes.

7.1. DDF Registry Prerequisites

To use the DDF Registry, the following apps/features must be installed:

- DDF Spatial

7.2. Installing DDF Registry

- Navigate to the Admin Console.
- Click the **Manage** applications button.
- Click the **Install** icon.
- DDF Registry will move to **Active Applications** upon startup.

7.3. Configuring DDF Registry

Table 22. Registry Configuration Tab

Name	Configuration ID	Description
Registry Federation Admin Service	`Registry_Federation_Admin_Service`	Registry Federation Admin Service Settings
Registry Policy Plugin	`org.codice.ddf.registry.policy.RegistryPolicyPlugin`	Registry Policy Plugin Settings
Registry Source Configuration Handler	`Registry_Configuration_Event_Handler`	Configurable values for the registry source configuration handler
Registry Store	`Csw_Registry_Store`	Registry CSW Store Settings

Table 23. Registry Federation Admin Service

Name	Property	Type	Description	Default Value	Required
Registry Sub Poller Interval	`registrySubPollerInterval`	String	The polling interval for the registry subscriptions. In seconds.	30	yes

Table 24. Registry Policy Plugin

Name	Property	Type	Description	Default Value	Required
Registry CUD Attributes	`writeAccessPolicyStrings`	String	Roles/attributes required for CUD (create/update/delete) operations on registry entries. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	yes
Registry Read Attributes	`readAccessPolicyStrings`	String	Roles/attributes required for reading registry entries. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	yes
Registry Admin Attributes	`registryBypassPolicyStrings`	String	Roles/attributes required for an admin to bypass all filtering/access controls. Example: {role=role1;type=type1}	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin	yes
Disable Registry Write Access	`registryDisabled`	Boolean	Disables all write access to registry entries in the catalog. Only users with Registry Admin Attributes will be able to write registry entries		no
Entries are White List	`whiteList`	Boolean	A flag indicating whether or not the Registry Entry Ids represent a 'white list' (allowed - checked) or a 'black list' (blocked - unchecked) ids		no
Registry Entries Ids	`registryEntryIds`	String	List of registry entry ids to be used in the white/black list.		no

Table 25. Registry Source Configuration Handler

Name	Property	Type	Description	Default Value	Required
Url Binding Name	`urlBindingName`	String	The url name for communicating with the specific instance.	urlBindingName	yes
BindingType to Factory PID	`bindingTypeFactoryPid`	String	Key/Value mappings of binding type to factory PID	CSW_2.0.2=Csw_Federated_Source,WFS_1.0.0=Wfs_v1_0_0_Federated_Source,OpenSearch_1.0.0=OpenSearchSource	yes
Remove Configurations on Metacard Delete	`cleanUpOnDelete`	Boolean	Flag used to determine if configurations should be deleted when the metacard is deleted.	false	yes
Activate Configurations	`activateConfigurations`	Boolean	Flag used to determine if a configuration should be activated on creation	false	yes
Preserve Active Configuration	`preserveActiveConfigurations`	Boolean	Flag used to determine if configurations should be preserved. If true will only allow auto activation on creation. If false auto activation will happen on updates as well. Only applicable if activateConfigurations is true.	true	yes
Source Activation Priority Order	`sourceActivationPriorityOrder`	String	This is the priority list used to determine which source should be activated on creation	CSW_2.0.2,WFS_1.0.0,OpenSearch_1.0.0	yes

Table 26. Registry Store

Name	Property	Type	Description	Default Value	Required
Registry ID	`id`	String	The unique name of the store		yes

Name	Property	Type	Description	Default Value	Required
CSW URL	`cswUrl`	String	URL to the endpoint implementing CSW spec capable of returning ebrim formatted records		yes
Username	`username`	String	Username for CSW Service (optional)		no
Password	`password`	Password for CSW Service (optional)	Password		no
Allow Push	`pushAllowed`	Boolean	Enable push (write) to this registry store	true	yes
Allow Pull	`pullAllowed`	Boolean	Enable pull (read) from this registry store	true	yes

7.4. Using DDF Registry

The **Local Registry Nodes** and **Remote Registries** tabs appear in both the DDF Registry application and the DDF Catalog application.

7.4.1. Configuring Identity Node

- Navigate to **DDF Registry** (or **DDF Catalog**) application.
- Navigate to **Local Registry Nodes** tab.
- Click the name of the identity node.
- Complete all *required* and any desired *optional* fields.
  - Add any desired service bindings under the Services tab.
- Click **Save**.
  - If service bindings were added, confirm setup by locating the identity node in the **Sources** tab of the **DDF Catalog** app.

Table 27. General Information Tab

Field	Description	Type	Required
Node Name	This node's name as it should appear to external systems	string	yes

Field	Description	Type	Required
Node Description	Short description for this node	string	yes
Node Version	This node's Version	string	yes
Security Attributes	Security attributes associated with this node.	String	
Last Updated	Date this entry's data was last updated	Date	
Live Date	Date indicating when this node went live or operational	Date	
Custom Fields	click **Add** button to add custom fields	Configurable	no
Associations	click **Add** button to add custom fields	Configurable	no

Table 28. Services

Field	Description	Type	Required
Service Name	This service name	string	
Service Description	Short description for this service	string	
Service Version	This service version	string	
Service Type	Identifies the type of service this is by a URN.	string	
Bindings (Click **Add** to add a service binding)			
Binding Name	This binding name	String	yes
Binding Description	Short description for this binding	String	
Binding Version	This binding version		
Access URL	The url used to access this binding		
Service Binding Type	The binding type for the service		
URL Property Key	Property that the accessURI value should be put into for source creation		
Custom Fields	click **Add** button to add custom fields	Configurable	no
Associations	click **Add** button to add custom fields	Configurable	no

Table 29. Organizations Tab (click **Add** *to add an organization)*

Field	Description	Type	Required
Organization Name	This organization's name	string	yes
Address	This organization's primary address	Expand to enter address information	yes
TelephoneNumber	Primary contact number for this organization		no
Email	Primary contact email for this organization		no
Custom Fields	click **Add** button to add custom fields	Configurable	no
Associations	click **Add** button to add custom fields	Configurable	no

Table 30. Contacts (click **Add** *button to add contact info)*

Field	Description	Type	Required
Contact Title	Contact Title	String	yes
Contact First Name	Contact First Name	String	yes
Contact Last Name	Contact Last Name	String	yes
Address	Address for listed contact	String	minimum one
Phone number	Contact phone number		minimum one
Email	Contact email	String	minimum one
Custom Fields	click **Add** button to add custom fields	Configurable	no
Associations	click **Add** button to add custom fields	Configurable	no

Table 31. Collections (Click **Add** *to add Content Collection(s))*

Field	Description	Type	Required
Content Name	Name for this metadata content	string	yes
Content Description	Short description for this metadata content	string	no
Content Object Type	The kind of content object this will be. Default value should be used in most cases.	string	yes
Custom Fields	click **Add** button to add custom fields	Configurable	no
Associations	click **Add** button to add custom fields	Configurable	no

Adding a Service Binding

- Navigate to Admin Console.
- Select DDF Registry or DDF Catalog.
  - (**Local Registry Nodes** tab is editable from either application.)
- Click the name of the desired node.
- Enter (at minimum) a **Node Name** under **General Information** tab).
- Click the **Services** tab.
- Click **Add** to add a service.
- Expand new Service.
- Enter Service name and details.
- Click **Add** to add binding.
- Select Service Binding type.
  - Select one of the defaults or *empty* for a custom service binding.
  - If selecting *empty*, fill in all required fields.
- Click Save.

7.4.2. Publishing to Others

- Navigate to the **Remote Registries** tab in either DDF Registry or DDF Catalog application.
- Click **Add** to add a remote registry.
- Give node a unique local name.
- Enter Registry Service (CSW) Url.
- Confirm **Allow Push** is checked.
- Click **Add** to save the changes.
- Navigate to the **Sources** Tab in DDF Catalog App
- Click desired node to be published.
- Under **Operations**, click *Publish to ... * link.

7.4.3. Subscribing to Another Node

- Navigate to the **Remote Registries** tab in either DDF Registry or DDF Catalog application.
- Click **Add** to add a remote registry.
- Give new node a unique Local name.
- Add the URL to access node.
- Enter any needed credentials in the Username/password fields.
- Click **Save**/**Add**.

Editing a Subscription

- Navigate to the **Remote Registries** tab in either DDF Registry or DDF Catalog application.
- Click the name of the desired subscription.
- Make changes.
- Click **Save**.

Deleting a Subscription

- Click the **Delete** icon corresponding to the desired node to delete.

8. Managing DDF Security

Version: 2.9.1

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

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

8.1. Installing DDF Security

8.1.1. Prerequisites

Before the DDF Security application can be installed:

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

8.1.2. Installing DDF Security

DDF Security is included with a standard installation.

8.2. Configuring DDF Security

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

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

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

Table 32. Login Page

Name	Property	Type	Description	Default Value	Required
Header	header	String	Specifies the header text to be rendered on the Login page		yes

Name	Property	Type	Description	Default Value	Required
Footer	`footer`	String	Specifies the footer text to be rendered on the Login page		yes
Style	`style`	String	Specifies the style of the Header and Footer.	green	yes
Text Color	`textColor`	String	Specifies the text color of the Header and Footer.	banner-text-white	yes

Table 33. SAML NameID Policy

Name	Property	Type	Description	Default Value	Required
SAML NameID Policy	`usernameAttributeList`	String	List of attributes that are considered for replacing the username of the logged in user. If any of these attributes match any of the attributes within the SecurityAssertion, the value of the first matching attribute will be used as the username. (Does not apply when NameIDFormat is of the following: X509, persistent, kerberos or unspecified, and the username is not empty).	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier, uid	yes

Table 34. Security SOAP Guest Interceptor

Name	Property	Type	Description	Default Value	Required
Deny Anonymous Access	`anonymousAccessDenied`	Boolean	If set to true, no anonymous access will be allowed via this anonymous interceptor. If set to false, this interceptor will generate anonymous tokens for incoming requests that lack a WS-Security header.	false	no

Name	Property	Type	Description	Default Value	Required
Override Endpoint Policies	`overrideEndpointPolicies`	Boolean	If checked, forces anonymous tokens to be created and inserted into the incoming request regardless of whether the policy requires an issued token. If set to false, if the endpoint policies cannot be satisfied, no changes will be made to the incoming request. This only applies to incoming requests that lack a WS-Security header - those with a WS-Security header are passed through unchanged.	false	no

Table 35. STS Server Token Endpoint

Name	Property	Type	Description	Default Value	Required
	`endpoints`	String	The list of endpoint addresses that correspond to this service	.*	yes

Table 36. Security STS Address Provider

Name	Property	Type	Description	Default Value	Required
Use WSS STS	`useWss`	Boolean	If you have a WSS STS configured, you may prefer to use it for services that need the STS address, such as SOAP sources.	false	yes

Table 37. Security STS Client

Name	Property	Type	Description	Default Value	Required
SAML Assertion Type:	`assertionType`	String	The version of SAML to use. Most services require SAML v2.0. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0	yes
SAML Key Type:	`keyType`	String	The key type to use with SAML. Most services require Bearer. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer	yes
SAML Key Size:	`keySize`	String	The key size to use with SAML. The default key size is 256 and this is fine for most applications. Changing this value from the default could cause services to stop responding.	256	yes
Use Key:	`useKey`	Boolean	Signals whether or not the STS Client should supply a public key to embed as the proof key. Changing this value from the default could cause services to stop responding.	true	yes
STS WSDL Address:	`address`	String	STS WSDL Address	${org.codice.ddf.system.protocol}${org.codice.ddf.system.hostname}:${org.codice.ddf.system.port}${org.codice.ddf.system.rootContext}/SecurityTokenService?wsdl	yes

Name	Property	Type	Description	Default Value	Required
STS Endpoint Name:	`endpointName`	String	STS Endpoint Name.	no	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}STS_Port
STS Service Name:	`serviceName`	String	STS Service Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}SecurityTokenService	no
Signature Properties:	`signatureProperties`	String	Path to Signature crypto properties. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/signature.properties	yes
Encryption Properties:	`encryptionProperties`	String	Path to Encryption crypto properties file. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/encryption.properties	yes
STS Properties:	`tokenProperties`	String	Path to STS crypto properties file. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/signature.properties	yes

Name	Property	Type	Description	Default Value	Required
Claims:	`claims`	String	List of claims that should be requested by the STS Client.	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role	yes

Table 38. Security STS Guest Claims Handler

Name	Property	Type	Description	Default Value	Required
Attributes	`attributes`	String	The attributes to be returned for any Guest user.	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier=guest,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	yes

Table 39. Security STS Guest Validator

Name	Property	Type	Description	Default Value	Required
Supported Realms	supportedRealm	String	The realms that this validator supports.	karaf,ldap	yes

Table 40. Security STS Server

Name	Property	Type	Description	Default Value	Required
SAML Assertion Lifetime:	lifetime	Long	Set the number of seconds that an issued SAML assertion will be good for.	1800	yes
Token Issuer:	issuer	String	The name of the server issuing tokens. Generally this is the cn or hostname of this machine on the network.	${org.codice.ddf.system.hostname}	yes
Signature Username:	signatureUsername	String	Alias of the private key in the STS Server's keystore used to sign messages.	${org.codice.ddf.system.hostname}	yes
Encryption Username:	encryptionUsername	String	Alias of the private key in the STS Server's keystore used to encrypt messages.	${org.codice.ddf.system.hostname}	yes

Table 41. Security STS PKI Token Validator

Name	Property	Type	Description	Default Value	Required
Realms	realms	String	The realms to be validated by this validator.	karaf	yes
Do Full Path Validation	pathValidation	Boolean	Validate the full certificate path. Uncheck to only validate the subject cert. (RFC5280 6.1)	true	yes

Table 42. File Based Claims Handler

Name	Property	Type	Description	Default Value	Required
Role Claim Type:	roleClaimType	String	Role claim URI.	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role	yes

Name	Property	Type	Description	Default Value	Required
User Role File:	`propertyFileLocation`	String	Location of the file that maps roles to users.	etc/users.properties	yes
User Attribute File:	`attributeFileLocation`	String	Location of the file that maps attributes to users.	etc/users.attributes	yes

Table 43. Security STS Server

Name	Property	Type	Description	Default Value	Required
SAML Assertion Lifetime:	`lifetime`	Long	Set the number of seconds that an issued SAML assertion will be good for.	1800	yes
Token Issuer:	`issuer`	String	The name of the server issuing tokens. Generally this is the cn or hostname of this machine on the network.	${org.codice.ddf.system.hostname}	yes
Signature Username:	`signatureUsername`	String	Alias of the private key in the STS Server's keystore used to sign messages.	${org.codice.ddf.system.hostname}	yes
Encryption Username:	`encryptionUsername`	String	Alias of the private key in the STS Server's keystore used to encrypt messages.	${org.codice.ddf.system.hostname}	yes

Table 44. Security STS WSS

Name	Property	Type	Description	Default Value	Required
SAML Assertion Type:	`assertionType`	String	The version of SAML to use. Most services require SAML v2.0. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0	yes

Name	Property	Type	Description	Default Value	Required
SAML Key Type:	keyType	String	The key type to use with SAML. Most services require Bearer. Changing this value from the default could cause services to stop responding.	http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer	yes
SAML Key Size:	keySize	String	The key size to use with SAML. The default key size is 256 and this is fine for most applications. Changing this value from the default could cause services to stop responding.	256	yes
Use Key:	useKey	Boolean	Signals whether or not the STS Client should supply a public key to embed as the proof key. Changing this value from the default could cause services to stop responding.	true	yes
STS WSDL Address:	address	String	STS WSDL Address	${org.codice.ddf.system.protocol}${org.codice.ddf.system.hostname}:${org.codice.ddf.system.httpsPort}${org.codice.ddf.system.rootContext}/SecurityTokenService?wsdl	yes
STS Endpoint Name:	endpointName	String	STS Endpoint Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}STS_Port	no

Name	Property	Type	Description	Default Value	Required
STS Service Name:	`serviceName`	String	STS Service Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}SecurityTokenService	no
Signature Properties:	`signatureProperties`	String	Path to Signature crypto properties. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/signature.properties	yes
Encryption Properties:	`encryptionProperties`	String	Path to Encryption crypto properties file. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/encryption.properties	yes
STS Properties:	`tokenProperties`	String	Path to STS crypto properties file. This path can be part of the classpath, relative to ddf.home, or an absolute path on the system.	etc/ws-security/server/signature.properties	yes

Name	Property	Type	Description	Default Value	Required
Claims:	`claims`	String	Comma-delimited list of claims that should be requested by the STS.	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role	yes

Table 45. Security AuthZ Realm

Name	Property	Type	Description	Default Value	Required
Match-All Mappings	`matchAllMappings`	String	List of 'Match-All' subject attribute to Metacard attribute mapping. All values of this metacard key must be present in the corresponding subject key values. Format is subjectAttrName=metacardAttrName.		no

Name	Property	Type	Description	Default Value	Required
Match-One Mappings	`matchOneMappings`	String	List of 'Match-One' subject attribute to Metacard attribute mapping. One value of this metacard key must be present in the corresponding subject key values. Format is subjectAttrName=metacardAttrName.		no
Environment Attributes	`environmentAttributes`	String	List of environment attributes to pass to the XACML engine. Format is attributeId=attributeValue1,attributeValue2.		no

Table 46. Session

Name	Property	Type	Description	Default Value	Required
Session Timeout (in minutes)	`expirationTime`	Integer	The number of minutes after a session has been inactive that it should be invalidated.	31	yes

Table 47. Web Context Policy Manager Configuration

Name	Property	Type	Description	Default Value	Required
Context Realms	`realms`	String	List of realms supporting each context. `karaf` is provided by default. Example: `/=karaf`	/=karaf	yes
Authentication Types	`authenticationTypes`	String	List of authentication types required for each context. List of default valid authentication types are: SAML, BASIC, PKI, GUEST. Example: /context=AUTH1|AUTH2|AUTH3	/=SAML|GUEST,/admin=SAML|basic,/system=basic,/solr=SAML|PKI|basic,/sources=SAML|basic,/security-config=SAML|basic	yes

Name	Property	Type	Description	Default Value	Required
Required Attributes	`requiredAttributes`	String	List of attributes required for each Web Context. Example: /context={role=role1;type=type1}"	/=,/admin={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin},/solr={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin},/system={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin},/security-config={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=system-admin}	yes

Name	Property	Type	Description	Default Value	Required
White Listed Contexts	`whiteListContexts`	String	List of contexts that will not use security. Note that sub-contexts to ones listed here will also skip security, unless authentication types are provided for it. For example: if /foo is listed here, then /foo/bar will also not require any sort of authentication. However, if /foo is listed and /foo/bar has authentication types provided in the 'Authentication Types' field, then that more specific policy will be used.	${org.codice.ddf.system.rootContext}/SecurityTokenService,${org.codice.ddf.system.rootContext}/internal/metrics,${org.codice.ddf.system.rootContext}/saml,/proxy,${org.codice.ddf.system.rootContext}/saml,${org.codice.ddf.system.rootContext}/idp,/idp,${org.codice.ddf.system.rootContext}/platform/config/ui	yes

NOTE For more details on how sub-contexts affect authentication realms, see Configuring DDF Authentication Scheme.

8.2.1. Applications Included in DDF Security

- Security CAS
- Security Core
- Security Encryption
- Security IdP
- Security PEP
- Security PDP
- Security STS

9. Managing DDF Solr

Version: 2.9.1

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

9.1. DDF Catalog Solr External Provider

9.1.1. Using

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

9.1.2. Installing and Uninstalling

Prerequisites

Before the DDF Solr Application can be installed,

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

Installing

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

Configuring

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

Configurable Properties

Title	Property	Type	Description	Default Value	Required
HTTP URL	`url`	String	HTTP URL of the standalone, preconfigured Solr Server.	https://localhost:8993/solr	Yes

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

9.2. DDF Catalog Solr Embedded Provider

9.2.1. Using

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

9.2.2. Installing and Uninstalling

Prerequisites

Before the DDF Solr Application can be installed:

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

Installing

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

9.2.3. Configuring Embedded Solr Server and Solr Catalog Provider

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

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

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

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

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

Configurable Properties

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

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

9.2.4. Solr Configuration Files

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

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

9.2.5. Move Solr Data to a New Location

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

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

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

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

9.3. Standalone Solr Server

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

9.3.1. Using

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

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

9.3.2. Installing and Uninstalling

Prerequisites

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

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

9.3.3. Installing

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

Remove Data from Solr Core

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

How to delete Solr Core data with curl

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

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

Solr administration page

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

9.3.4. Configuring

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

9.3.5. Known Issues

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

9.3.6. Solr Standalone Server Meta Catalog Backup

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

Back Up Data from the Solr Server Standalone Metadata Catalog

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

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

command

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

location

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

numberToKeep

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

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

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

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

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

Restore Data to the Solr Server Standalone Metadata Catalog

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

1. The first step in the process is to choose which data backup will be used for restoring the catalog. A most recent backup maybe the correct choice, or the last stable backup may be a better option.

2. At this point, one more backup may be executed to save the corrupted data just in case it needs to be revisited.

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

Suggestions for Managing Backup and Recovery

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

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

10. Managing DDF Spatial

Version: 2.9.1

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

This page describes:

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

10.1. Prerequisites

Before the DDF Spatial Application can be installed:

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

10.2. Installing

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

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

10.2.1. Verifying

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

10.2.2. Uninstalling

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

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

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

Uninstall DDF application from DDF distribution

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

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

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

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

10.2.3. Reverting the Uninstall

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

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

Reverting DDF application's uninstall

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

Example

```
ddf@local>feature:repo-add mvn:ddf.catalog/catalog-app/2.9.1/xml/features
ddf@local>feature:install catalog-app
```

10.2.4. Upgrading

To upgrade an application, complete the following procedure.

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

10.3. Optional Features

10.3.1. Offline Gazetteer Service

In the Spatial Application, you have the option to install a feature called `offline-gazetteer`. This feature enables you to use an offline index of GeoNames data (as an alternative to the GeoNames Web service enabled by the `webservice-gazetteer` feature) to perform searches via the gazetteer search box in the Search UI.

To use the offline gazetteer, you will need to create an index. To do so, you'll need to use the `geonames:update` command which is explained in the next section.

10.4. Console Commands

Table 48. GeoNames Commands

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

10.4.1. Commands

```
geonames:update
```

10.4.2. Command Descriptions

Command	Description
`update`	Adds new entries to an existing local GeoNames index. Entries can be manually downloaded from http://download.geonames.org/export/dump, where the absolute path of the file would be passed as an argument to the command (ex. /Users/johndoe/Downloads/AU.zip). Currently .txt and .zip files are supported for manual entries. Entries can also be automatically downloaded from http://download.geonames.org/export/dump by passing the country code as an argument to the command (ex. AU) which will add the country to the local GeoNames index. The full list of country codes available can be found in http://download.geonames.org/export/dump/countryInfo.txt. Using the argument "all" will download all of the current country codes (this process may take some time). In addition to country codes, GeoNames also provides entries for cities based on their population sizes. The arguments "cities1000", "cities5000", and "cities15000" will add cities to the index that have at least 1000, 5000, or 15000 people respectively. The index location can be configured via the Admin Console. By default, the index location is `data/geonames-index`. If you specify a relative path, it is relative to the location of the unzipped DDF distribution. You may specify an absolute path if you want the index to be located somewhere else. The `-c` or `--create` flag can be added to create a new GeoNames index. This will overwrite any existing index at the location specified in the Admin Console. The new index will be filled with the entries in the file you pass to the command. You must create an index before you can add additional entries to it (i.e. running the command without the `-c` or `--create` flag).

11. Managing DDF Search UI

Version: 2.9.1

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

This section describes:

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

11.1. Prerequisites

Before the DDF Search UI application can be installed:

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

11.2. Installing DDF Search UI

The Search UI application is installed by default.

11.3. Configuring DDF Search UI

Configure individual features within the application with the Admin Console.

11.3.1. Configurable Properties

Configurations

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

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

Table 49. Search UI Endpoint

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

Table 50. Search UI Redirect

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

Table 51. Simple Search UI

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

Table 52. Standard Search UI

Title	Property	Type	Description	Required
Header	`header`	String	The header text to be rendered on the Search UI.	no
Footer	`footer`	String	The footer text to be rendered on the Search UI.	no
Style	`style`	String	The style name (background color) of the Header and Footer.	yes
Text Color	`textColor`	String	The text color of the Heater and Footer.	yes
Result count	`resultCount`	Integer	The max number of results to display.	yes
Imagery Providers	`imageryProviders`	String	List of imagery providers to use. Valid types are: -OSM (OpenStreetMap), -AGM (ArcGisMap), -BM (BingMap),+ -WMS (WebMapService), -WMT (WebMapTile), -TMS (TileMapService), -GE (GoogleEarth). - -Example: TYPE={url= -{"type" "WMS" "url" "http://example.com" "layers" ["layer1" "layer2"] "parameters" {"FORMAT" "image/png" "VERSION" "1.1.1"} "alpha" 0.5}}	no
Terrain Providers	`terrainProvider`	String	Terrain provider to use for height data. Valid types are: -CT (CesiumTerrain), -AGS (ArcGisImageServer), -VRW (VRTheWorld). - -Example: -{"type" "CT" "url" "http://example.com"}	no
Map Projection	`projection`	String	Projection of imagery providers	no
Connection timeout	`timeout`	Integer	The WMS connection timeout.	yes

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

11.3.2. Upgrading

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

11.4. Troubleshooting DDF Search UI

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

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

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

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

Deletion of all records in Solr query cache

```
https://localhost:8993/solr/metacard_cache/update?stream.body=<delete><query>*:*</query><
/delete>&commit=true
```

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

Deletion of record in Solr query cache by ID

```
https://localhost:8993/solr/metacard_cache/update?stream.body=<delete><query>original_id_
txt:50ffd32b21254c8a90c15fccfb98f139</query></delete>&commit=true
```

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

Deletion of records in Solr query cache using search criteria

```
https://localhost:8993/solr/metacard_cache/update?stream.body=<delete><query>title_txt:*r
emote*</query></delete>&commit=true
```