Managing Distributed Data Framework

Table of Contents

License . . . 7
1. Securing . . . 8
1.1. Security Hardening . . . 8
1.2. Auditing . . . 9
1.2.1. Enabling Fallback Audit Logging . . . 9
2. Installing . . . 10
2.1. Installation Prerequisites . . . 10
2.1.1. Hardware Requirements . . . 11
2.1.2. Java Requirements . . . 11
2.2. Installing With the DDF Distribution Zip . . . 14
2.2.1. Configuring Operating Permissions and Allocations . . . 17
2.2.1.1. Setting Directory Permissions . . . 17
2.2.1.2. Configuring Memory Allocation for the DDF Java Virtual Machine . . . 18
2.2.1.3. Enabling JMX . . . 19
2.2.2. Managing Keystores and Certificates . . . 19
2.2.2.1. Managing Keystores . . . 20
2.2.2.1.1. Adding an Existing Server Keystore . . . 21
2.2.2.1.2. Adding an Existing Server Truststore . . . 21
2.2.2.1.3. Creating a New Keystore / Truststore with an Existing Certificate and Private Key . . . 21
2.2.2.1.4. Updating Key Store / Trust Store via the Admin Console . . . 22
2.3. Initial Startup . . . 24
2.3.1. Verifying Startup . . . 24
2.3.2. DDF Directory Contents after Installation and Initial Startup . . . 25
2.3.3. Completing Installation . . . 26
2.3.3.1. Completing Installation from the Admin Console . . . 26
2.3.3.2. Completing Installation from the Command Console . . . 29
2.3.4. Internet Explorer 11 Enhanced Security Configuration . . . 29
2.3.5. Firewall Port Configuration . . . 30
2.4. High Availability Initial Setup . . . 30
2.4.1. High Availability Initial Setup Exceptions . . . 31
2.4.1.1. Failover Proxy Integration . . . 31

2.4.1.2. Identical Directory Structures . . . 32
2.4.1.3. Highly Available Security Auditing . . . 32
2.4.1.4. Shared Storage Provider . . . 32
2.4.1.5. High Availability Certificates . . . 32
2.4.1.6. High Availability Installation Profile . . . 33
3. Configuring . . . 33
3.1. Admin Console Tutorial . . . 34
3.2. Console Command Reference . . . 35
3.2.1. Feature Commands . . . 35
3.2.1.1. Uninstalling Features from the Command Console . . . 36
3.3. Configuration Files . . . 37
3.3.1. Configuring Global Settings with custom.system.properties . . . 37
3.3.2. Configuring with .config Files . . . 49
3.4. Configuring User Access . . . 49
3.4.1. Configuring Guest Access . . . 50
3.4.1.1. Denying Guest User Access . . . 50
3.4.1.2. Allowing Guest User Access . . . 50
3.4.1.2.1. Configuring Guest Interceptor if Allowing Guest Users . . . 50
3.4.1.2.2. Configuring Guest Claim Attributes . . . 51
3.4.2. Configuring REST Services for Users . . . 51
3.4.2.1. Configuring OpenID Connect (OIDC) and OAuth 2.0 . . . 51
3.4.2.1.1. OIDC . . . 52
3.4.2.1.2. OAuth 2.0 . . . 52
3.4.2.2. Connecting to an External SAML Identity Provider . . . 53
3.4.2.3. Configuring Without SAML . . . 53
3.4.2.4. Configuring Multi Factor Authentication . . . 54
3.4.3. Connecting to an LDAP Server . . . 54
3.4.4. Updating System Users . . . 55
3.4.5. Restricting Access to Admin Console . . . 56
3.4.5.1. Restricting Feature, App, Service, and Configuration Access . . . 56
3.4.6. Restricting Access to Features, Apps, Services, and Configurations . . . 57
3.4.7. Removing Default Users . . . 58
3.4.8. Disallowing Login Without Certificates . . . 59
3.4.9. Managing Certificate Revocation . . . 59
3.4.9.1. Managing a Certificate Revocation List (CRL) . . . 60
3.4.9.1.1. Creating a CRL . . . 60
Revoke a Certificate and Create a New CRL that Contains the Revoked Certificate . . . 60
Viewing a CRL . . . 60

3.4.9.1.2. Enabling Certificate Revocation . . . 60
Add Revocation to a Web Context . . . 61
Adding Revocation to an Endpoint . . . 61
Verifying Revocation . . . 62
3.4.9.2. Managing an Online Certificate Status Protocol (OCSP) Server . . . 63
3.4.9.2.1. Enabling OCSP Revocation . . . 63
3.5. Configuring Data Management . . . 63
3.5.1. Configuring Solr . . . 63
3.5.1.1. Configuring Solr Catalog Provider Synonyms . . . 64
3.5.1.1.1. Defining synonym rules in the Solr Provider . . . 64
3.5.1.2. Hardening Solr . . . 64
3.5.1.2.1. Configuring Solr Encryption . . . 64
3.5.1.3. Accessing the Solr Admin UI . . . 64
3.5.2. Changing Catalog Providers . . . 64
3.5.3. Changing Hostname . . . 65
3.5.4. Configuring Errors and Warnings . . . 65
3.5.4.1. Enforcing Errors or Warnings . . . 65
3.5.4.2. Hiding Errors or Warnings from Queries . . . 66
3.5.4.3. Hiding Errors and Warnings from Users Based on Role . . . 66
3.5.5. Configuring Product Caching . . . 67
3.5.6. Content Directory Monitor . . . 67
3.5.6.1. Installing the Content Directory Monitor . . . 67
3.5.6.2. Configuring Permissions for the Content Directory Monitor . . . 67
3.5.6.3. Configuring the Content Directory Monitor . . . 68
3.5.6.4. Using the Content Directory Monitor . . . 68
3.5.7. Configuring System Usage Message . . . 71
3.5.8. Configuring Data Policy Plugins . . . 71
3.5.8.1. Configuring the Metacard Attribute Security Policy Plugin . . . 71
3.5.8.2. Configuring the Metacard Validation Marker Plugin . . . 72
3.5.8.3. Configuring the Metacard Validity Filter Plugin . . . 72
3.5.8.4. Configuring the XML Attribute Security Policy Plugin . . . 73
3.5.9. Configuring Data Access Plugins . . . 73
3.5.9.1. Configuring the Security Audit Plugin . . . 73
3.6. Configuring Security Policies . . . 74
3.6.1. Configuring the Web Context Policy Manager . . . 74
3.6.1.1. Guest Access . . . 74
3.6.1.2. Session Storage . . . 74
3.6.1.3. Authentication Types . . . 74

3.6.1.3.1. Terminating and Non-Terminating Authentication Types . . . 75
3.6.1.4. Required Attributes . . . 76
3.6.1.5. White Listed Contexts . . . 76
3.6.2. Configuring Catalog Filtering Policies . . . 76
3.6.2.1. Setting Internal Policies . . . 76
3.6.2.2. Setting XACML Policies . . . 77
3.6.2.3. Catalog Filter Policy Plugins . . . 77
3.7. Configuring User Interfaces . . . 77
3.8. Environment Hardening . . . 77
3.8.1. Known Issues with Environment Hardening . . . 77
3.9. Configuring Federation . . . 81
3.9.1. Enable SSL for Clients . . . 81
3.9.2. Configuring HTTP(S) Ports . . . 83
3.9.3. Federation Strategy . . . 83
3.9.3.1. Configuring Federation Strategy . . . 83
3.9.3.1.1. Catalog Federation Strategy . . . 84
3.9.4. Connecting to Sources . . . 84
3.9.4.1. CSW Specification Profile Federated Source . . . 86
3.9.4.2. Federated Source for Atlassian Confluence ® . . . 87
3.9.4.3. Content File System Storage Provider . . . 88
3.9.4.4. OpenSearch Source . . . 89
3.9.4.5. WFS 2.0 Source . . . 91
3.9.4.6. CSW Federation Profile Source . . . 93
3.9.4.7. Solr Catalog Provider . . . 93
3.9.4.8. WFS 1.1 Source . . . 96
3.9.4.9. GMD CSW Source . . . 97
3.9.5. Configuring Endpoints . . . 98
3.9.5.1. Configuring Catalog REST Endpoint . . . 98
3.9.5.2. Configuring CSW Endpoint . . . 98
3.9.5.3. Configuring KML Endpoint . . . 98
3.9.5.4. Configuring OpenSearch Endpoint . . . 99
3.10. Configuring for Special Deployments . . . 99
3.10.1. Multiple Installations . . . 100
3.10.1.1. Isolating SolrCloud and Zookeeper . . . 100
3.10.2. Configuring for a Fanout Proxy . . . 100
3.10.3. Configuring for a Highly Available Cluster . . . 100
3.11. Configuring UI Themes . . . 101
3.11.1. Landing Page . . . 101

3.11.1.1. Installing the Landing Page . . . 101
3.11.1.2. Configuring the Landing Page . . . 101
3.11.1.3. Customizing the Landing Page . . . 101
3.11.2. Configuring Logout Page . . . 102
3.11.3. Platform UI Themes . . . 102
3.11.3.1. Navigating to UI Theme Configuration . . . 102
3.11.3.2. Customizing the UI Theme . . . 102
3.12. Miscellaneous Configurations . . . 103
3.12.1. Configuring Thread Pools . . . 103
3.12.2. Configuring Jetty ThreadPool Settings . . . 103
3.12.3. Configuring Alerts . . . 103
3.12.3.1. Configuring Decanter Alerting . . . 103
3.12.4. Encrypting Passwords . . . 104
3.12.4.1. Encryption Command . . . 104
4. Running . . . 104
4.1. Starting . . . 105
4.1.1. Managed Services . . . 105
4.1.2. Run DDF as a Managed Service . . . 105
4.1.2.1. Running as a Service with Automatic Start on System Boot . . . 105
4.1.2.2. Karaf Documentation . . . 107
4.1.3. Run Solr as Managed Service . . . 107
4.1.4. Starting from Startup Scripts . . . 107
4.1.5. Starting as a Background Process . . . 107
4.1.6. Stopping DDF . . . 108
4.2. Maintaining . . . 109
4.2.1. Console Commands . . . 109
4.2.1.1. Console Command Help . . . 109
4.2.1.2. CQL Syntax . . . 110
4.2.1.3. Available Console Commands . . . 110
4.2.1.3.1. Catalog Commands . . . 111
4.2.1.3.2. Solr Commands . . . 114
4.2.1.3.3. Subscriptions Commands . . . 114
4.2.1.3.4. Platform Commands . . . 117
4.2.1.3.5. Persistence Store Commands . . . 118
4.2.1.4. Command Scheduler . . . 118
4.2.1.4.1. Schedule a Command . . . 118
4.2.1.4.2. Updating a Scheduled Command . . . 119
4.2.1.4.3. Output of Scheduled Commands . . . 119

4.3. Monitoring . . . 120
4.3.1. Metrics Reporting . . . 120
4.3.2. Managing Logging . . . 120
4.3.2.1. Configuring Logging . . . 120
4.3.2.2. DDF log file . . . 120
4.3.2.3. Controlling log level . . . 120
4.3.2.4. Controlling the size of the log file . . . 120
4.3.2.5. Number of backup log files to keep . . . 120
4.3.2.6. Enabling logging of inbound and outbound SOAP messages for the DDF SOAP endpoints . . . 120
4.3.2.7. Logging External Resources . . . 121
4.3.2.8. Enabling HTTP Access Logging . . . 121
4.3.2.9. Using the LogViewer . . . 122
4.4. Troubleshooting . . . 122
4.4.1. Deleted Records Are Being Displayed In The Search UI's Search Results . . . 125
5. Data Management . . . 125
5.1. Ingesting Data . . . 125
5.1.1. Ingest Command . . . 126
5.1.2. Content Directory Monitor Ingest . . . 126
5.1.3. External Methods of Ingesting Data . . . 127
5.1.4. Creating And Managing System Search Forms Through Karaf . . . 128
5.1.5. Other Methods of Ingesting Data . . . 129
5.2. Backing Up the Catalog . . . 129
5.3. Validating Data . . . 129
5.3.1. Validator Plugins on Ingest . . . 130
5.3.1.1. Validators run on ingest . . . 130
5.3.2. Configuring Schematron Services . . . 131
5.3.3. Injecting Attributes . . . 131
5.3.4. Overriding Attributes . . . 131
5.4. Removing Expired Records from the Catalog . . . 132
5.5. Automatically Added Metacard Attributes . . . 132
5.5.1. Attributes Added on Ingest . . . 132
5.5.1.1. Attributes Added by Input Transformers . . . 133
5.5.1.2. Attributes Added by Attribute Injection . . . 134
5.5.1.3. Attributes Added by Default Attribute Types . . . 134
5.5.1.4. Attributes Added by Attribute Overrides (Ingest) . . . 134
5.5.1.5. Attributes Added by Pre-Authorization Plugins . . . 134
5.5.1.6. Attributes Added by Pre-Ingest Plugins . . . 135

5.5.2. Attributes Added on Query . . . 135
5.5.2.1. Attributes Added by Attribute Overrides (Query) . . . 135
Index . . . 136

License

Copyright (c) Codice Foundation.
This work is licensed under a Creative Commons Attribution 4.0 International License.

This document last updated: 2026-04-24.

Administrators install, maintain, and support existing applications. Use this section to prepare, install, configure, run, and monitor a DDF.

1. Securing

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

IMPORTANT

Securing DDF Components

DDF is enabled with an Insecure Defaults Service which warns 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.

Security concerns are highlighted in the configuration sections to follow.

1.1. Security Hardening

Security Hardening

To harden DDF, extra security precautions are required.

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

Refer to the Hardening Checklist for a compilation of these mitigations.

NOTE

The security precautions are best performed as configuration is taking place, so hardening steps are integrated into configuration steps.

This is to avoid setting an insecure configuration and having to revisit during hardening. Most configurations have a security component to them, and important considerations for hardening are labeled as such during configuration as well as provided in a checklist format.

Some of the items on the checklist are performed during installation and others during configuration. Steps required for hardening are marked as **Required for Hardening** and are collected here for convenience. Refer to the checklist during system setup.

1.2. Auditing

- **Required Step for Security Hardening**

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

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

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

NOTE

Audit Logging Best Practices

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

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

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

1.2.1. Enabling Fallback Audit Logging

- **Required Step for Security Hardening**

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

- edit `<DDF_HOME>/etc/org.ops4j.pax.logging.cfg`
  - uncomment the line (remove the `#` from the beginning of the line) for `log4j2` (`org.ops4j.pax.logging.log4j2.config.file = ${karaf.etc}/log4j2.xml`)
  - delete all subsequent lines

If you want to change the location of your systems security backup log from the default location: `<DDF_HOME>/data/log/securityBackup.log`, follow the next two steps:

- edit `<DDF_HOME>/security/configurations.policy`
  - find "Security-Hardening: Backup Log File Permissions"

    - below `grant codeBase "file:/pax-logging-log4j2"` add the path to the directory containing the new log file created in the next step.
- edit `<DDF_HOME>/etc/log4j2.xml`
    - find the entry for the `securityBackup` appender. (see example)
    - change value of `filename` and prefix of `filePattern` to the name/path of the desired failover security logs.

`securityBackup` *Appender Before*

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

`securityBackup` *Appender After*

```
<RollingFile name="securityBackup" append="true" ignoreExceptions="false"
                     fileName="<NEW_LOG_FILE>"
                     filePattern="<NEW_LOG_FILE>-%d{yyyy-MM-dd-HH}-%i.log.gz">
```

WARNING If the system is unable to write to the `security.log` file on system startup, fallback logging is unavailable. Verify that the `security.log` file is properly configured and contains logs before configuring a fall back.

2. Installing

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

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

NOTE Hardening guidance assumes a Standard installation.

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

2.1. Installation Prerequisites

WARNING For security reasons, DDF cannot be started from a user's home directory. If attempted, the system automatically shuts down.

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

2.1.1. Hardware Requirements

Table 1. Using the Standard installation of the DDF application:

Minimum and Recommended Requirements for DDF Systems		
Criteria	**Minimum**	**Recommended**
CPU	Dual Core 1.6 GHz	Quad Core 2.6 GHz
RAM	8 GB*	32 GB
Disk Space	40 GB	80 GB
Video Card	—	WebGL capable GPU
Additional Software	JRE 8 x64	JDK 8 x64

*The amount of RAM can be increased to support memory-intensive applications. See Memory Considerations

Operating Systems

DDF has been tested on the following operating systems and with the following browsers. Other operating systems or browsers may be used but have not been officially tested.

Table 2. Tested Operating Systems and Browsers

Operating Systems	Browsers
Windows Server 2012 R2 Windows Server 2008 R2 Service Pack 1 Windows 10 Linux CentOS 7 Debian 9	Internet Explorer 11 Microsoft Edge Firefox Chrome

2.1.2. Java Requirements

For a runtime system:

- JRE 8 x64 or OpenJDK 8 JRE must be installed.
- The `JRE_HOME` environment variable must be set to the locations where the JRE is installed

For a development system:

- JDK8 must be installed.
- The `JAVA_HOME` environment variable must be set to the location where the JDK is installed.
  1. Install/Upgrade to Java 8 x64 J2SE 8 SDK
     a. The recommended version is 8u60 or later.
     b. Java version must contain only number values.
  2. Install/Upgrade to JDK8.
  3. Set the `JAVA_HOME` environment variable to the location where the JDK is installed.

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

NOTE **NIX Unset JAVA_HOME if Previously Set*

Unset `JAVA_HOME` if it is already linked to a previous version of the JRE:

```
unset JAVA_HOME
```

If JDK was installed:

Setting JAVA_HOME variable

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

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

   *Find Java Path in *NIX*

   ```
   which java
   ```

   Find Java Path in Windows

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

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

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

   *Setting `JAVA_HOME` on *NIX*

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

Setting `JAVA_HOME` on Windows

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

Adding `JAVA_HOME` to `PATH` Environment Variable on Windows

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

5. Restart or open up a new Terminal (shell) or Command Prompt to verify `JAVA_HOME` was set correctly. It is not necessary to restart the system for the changes to take effect.

   **NIX*

   ```
   echo $JAVA_HOME
   ```

   Windows

   ```
   echo %JAVA_HOME%
   ```

If JRE was installed:

Setting JRE_HOME variable

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

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

   *Find Java Path in *NIX*

   ```
   which java
   ```

   Find Java Path in Windows

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

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

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

*Setting `JRE_HOME` on *NIX*

```
JRE_HOME=<PATH_TO_JAVA><JAVA_VERSION>
export JRE_HOME
```

Setting `JRE_HOME` on Windows

```
set JRE_HOME=<PATH_TO_JAVA><JAVA_VERSION>
setx JRE_HOME "<PATH_TO_JAVA><JAVA_VERSION>"
```

Adding `JRE_HOME` to `PATH` Environment Variable on Windows

```
setx PATH "%PATH%;%JRE_HOME%\bin"
```

5. Restart or open up a new Terminal (shell) or Command Prompt to verify `JRE_HOME` was set correctly. It is not necessary to restart the system for the changes to take effect.

**NIX*

```
echo $JRE_HOME
```

Windows

```
echo %JRE_HOME%
```

NOTE

File Descriptor Limit on Linux

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

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

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

**Nix Restart Command*

```
init 6
```

2.2. Installing With the DDF Distribution Zip

WARNING *Check System Time*

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

To install the DDF distribution zip, perform the following:

1. Download the DDF zip file.
2. After the prerequisites have been met, change the current directory to the desired install directory, creating a new directory if desired. This is referred to as `<DDF_HOME>`.

WARNING	*Windows Pathname Warning* Do not use spaces in directory or file names of the `<DDF_HOME>` path. For example, do not install in the default `Program Files` directory.

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

```
mkdir new_installation
```

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

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

b. Create a new group or use an existing group (such as DDF_GROUP) (Windows users skip this step)

   *Example Add New Group on *NIX*

   ```
   groupadd DDF_GROUP
   ```

   *Example Switch User on *NIX*

   ```
   chown DDF_USER:DDF_GROUP new_installation

   su - DDF_USER
   ```

3. Change the current directory to the location of the zip file (ddf-2.30.1.zip).

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

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

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

```
cd D:\
```

4. Copy ddf-2.30.1.zip to <DDF_HOME>.

   **NIX*

   ```
   cp ddf-2.30.1.zip <DDF_HOME>
   ```

   Windows

   ```
   copy ddf-2.30.1.zip <DDF_HOME>
   ```

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

   **NIX or Windows*

   ```
   cd <DDF_HOME>
   ```

6. The DDF zip is now located within the `<DDF_HOME>`. Unzip ddf-2.30.1.zip.

   **NIX*

   ```
   unzip ddf-2.30.1.zip
   ```

WARNING

Windows Zip Utility Warning

The Windows Zip implementation, which is invoked when a user double-clicks on a zip file in the Windows Explorer, creates a corrupted installation. This is a consequence of its inability to process long file paths. Instead, use the java jar command line utility to unzip the distribution (see example below) or use a third party utility such as 7-Zip.

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

```
"<PATH_TO_JAVA>\jdk<JAVA_VERSION>\bin\jar.exe" xf ddf-2.30.1.zip
```

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

2.2.1. Configuring Operating Permissions and Allocations

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

Within the `<DDF_HOME>`, a directory is created named ddf-2.30.1. This directory is referred to in the documentation as `<DDF_HOME>`.

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

2.2.1.1. Setting Directory Permissions

- **Required Step for Security Hardening**

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

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

Setting Directory Permissions on Windows

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

1. Right-click on the `<DDF_HOME>` directory.
2. Select **Properties** → **Security** → **Advanced**.
3. Under **Owner**, select **Change**.
4. Enter `Creator Owner` into the **Enter the Object Name...** field.
5. Select **Check Names**.
6. Select **Apply**.
   a. If prompted **Do you wish to continue**, select **Yes**.
7. Remove all Permission Entries for any groups or users with access to <DDF_HOME> other

than **System**, **Administrators**, and **Creator Owner**.

a. Note: If prompted with a message such as: **You can't remove X because this object is inheriting permissions from its parent.** when removing entries from the Permission entries table:

   i. Select **Disable Inheritance**.

   ii. Select **Convert Inherited Permissions into explicit permissions on this object**.

   iii. Try removing the entry again.

8. Select the option for **Replace all child object permission entries with inheritable permission entries from this object**.
9. Close the **Advanced Security Settings** window.

Setting Directory Permissions on *NIX

Set directory permissions to protect the DDF from unauthorized access.

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

2.2.1.2. Configuring Memory Allocation for the DDF Java Virtual Machine

The amount of memory allocated to the Java Virtual Machine host DDF by the operating system can be increased by updating the `setenv` script:

*Setenv Scripts (*NIX)*

```
<DDF_HOME>/bin/setenv
Update the JAVA_OPTS -Xmx value
<DDF_HOME>/bin/setenv-wrapper.conf
Update the wrapper.java.additional -Xmx value
```

Setenv Scripts (Windows)

```
<DDF_HOME>/bin/setenv.bat
Update the JAVA_OPTS -Xmx value
<DDF_HOME>/bin/setenv-windows-wrapper.conf
Update the wrapper.java.additional -Xmx value
```

2.2.1.3. Enabling JMX

By default, DDF prevents connections to JMX because the system is more secure when JMX is not enabled. However, many monitoring tools require a JMX connection to the Java Virtual Machine. To enable JMX, update the setenv script:

*Setenv Scripts (*NIX)*

```
<DDF_HOME>/bin/setenv
Remove -XX:+DisableAttachMechanism from JAVA_OPTS
<DDF_HOME>/bin/setenv-wrapper.conf
Comment out the -XX:+DisableAttachMechanism line and re-number remainder lines
appropriately
```

Setenv Scripts (Windows)

```
<DDF_HOME>/bin/setenv.bat
Remove -XX:+DisableAttachMechanism from JAVA_OPTS
<DDF_HOME>/bin/setenv-windows-wrapper.conf
Comment out the -XX:+DisableAttachMechanism line and re-number remainder lines
appropriately
```

2.2.2. Managing Keystores and Certificates

- **Required Step for Security Hardening**

DDF uses certificates in two ways:

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

To ensure proper configuration of keystore, truststore, and certificates, follow the options below according to situation.

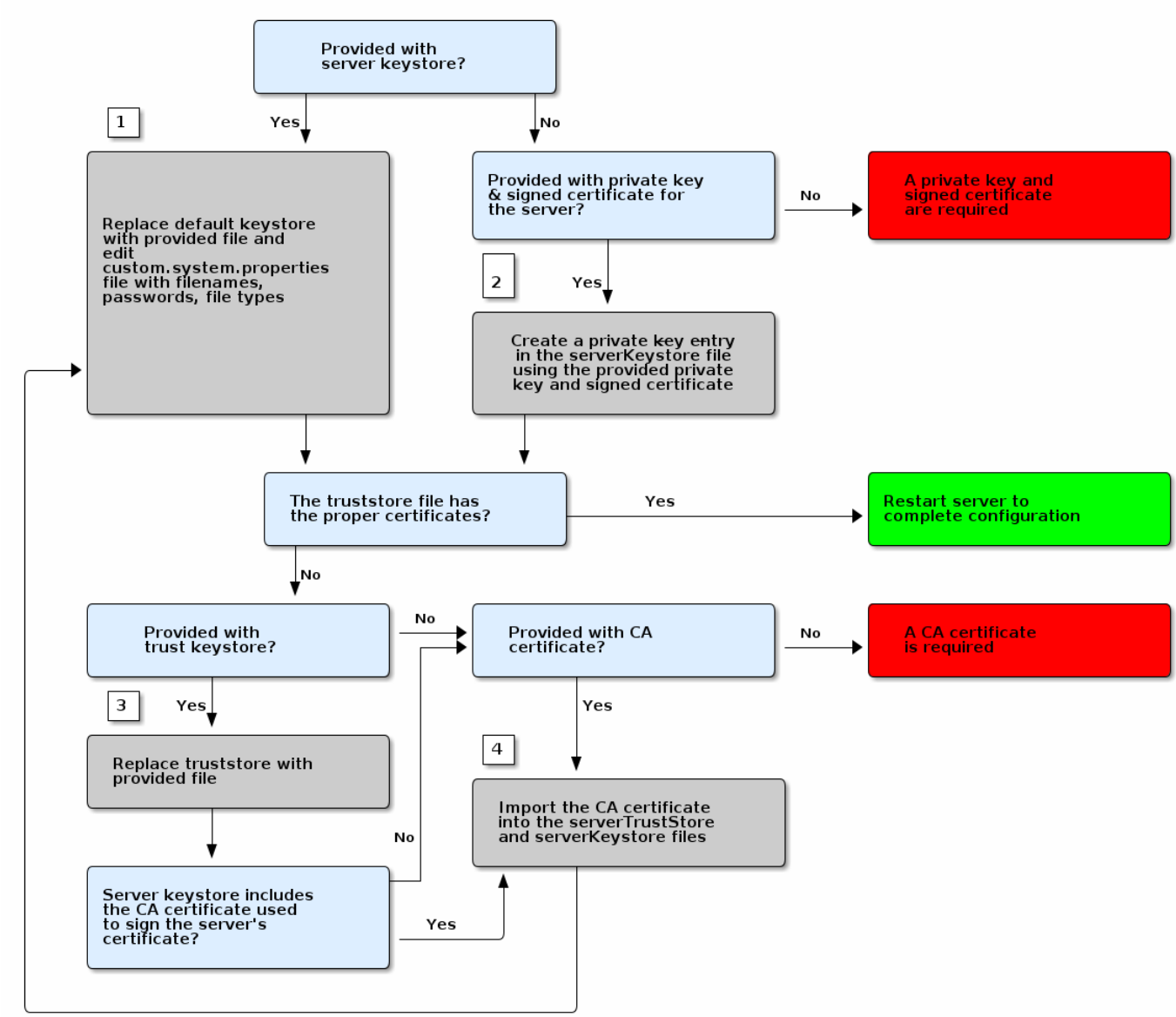

Figure 1. Configuring Certificates Workflow

Jump to the steps referenced in the diagram:

Certificate Workflow Steps

1. Adding an Existing Keystore
2. Creating a New Keystore/Truststore with an Existing Certificate and Private Key
3. Adding an Existing Truststore
4. Creating a Server Keystore
    a. Creating a Server Truststore

2.2.2.1. Managing Keystores

Certificates, and sometimes their associated private keys, are stored in keystore files. DDF includes two

default keystore files, the server key store and the server trust store. The server keystore holds DDF's certificate and private key. It also holds the certificates of other nodes whose signature DDF accept.The truststore holds the certificates of nodes or other entities that DDF needs to trust.

NOTE Individual certificates of other nodes should be added to the keystore instead of CA certificates. If a CA's certificate is added, DDF automatically trusts any certificate that is signed by that CA.

2.2.2.1.1. Adding an Existing Server Keystore

If provided an existing keystore for use with DDF, follow these steps to replace the default keystore.

1. Remove the default keystore at `etc/keystores/serverKeystore.jks`.
2. Add the desired keystore file to the `etc/keystores` directory.
3. Edit `custom.system.properties` file to set filenames and passwords.
   a. If using a type of keystore other than `jks` (such as `pkcs12`), change the `javax.net.ssl.keyStoreType` property as well.
4. If the truststore has the correct certificates, restart server to complete configuration.
   a. If provided with an existing server truststore, continue to Adding an Existing Server Truststore.
   b. Otherwise, create a server truststore.

2.2.2.1.2. Adding an Existing Server Truststore

1. Remove the default truststore at `etc/keystores/serverTruststore.jks`.
2. Add the desired truststore file to the `etc/keystores` directory.
3. Edit `custom.system.properties` file to set filenames and passwords.
   a. If using a type of truststore other than `jks` (such as `pkcs12`), change the `javax.net.ssl.trustStoreType` property as well.

If the provided server keystore does not include the CA certificate that was used to sign the server's certificate, add the CA certificate into the `serverKeystore` file.

NOTE *Trust Chain*

All CAs in the trust chain, including all intermeditate certificates, must be included in the trust store.

2.2.2.1.3. Creating a New Keystore / Truststore with an Existing Certificate and Private Key

If provided an existing certificate, create a new keystore and truststore with it. Use a tool such as the standard Java keytool certificate management utility.

NOTE DDF requires that the keystore contains both the private key and the CA.

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
```

IMPORTANT	The localhost certificate must be removed if using a system certificate.

2.2.2.1.4. Updating Key Store / Trust Store via the Admin Console

Certificates (and certificates with keys) can be managed in the Admin Console.

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

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

This view shows the alias (name) of every certificate in the trust store and the key store. It also displays if the entry includes a private key ("Is Key") and the encryption scheme (typically "RSA" or "EC").

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

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

IMPORTANT

Keystore Password

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

The location, file name, passwords, type of the server, and trust key stores can be set in the `custom.system.properties` file:

1. Setting the Keystore and Truststore Java Properties

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

NOTE

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

Changing Default Passwords

This step is not required for a hardened system.

- The default password in `custom.system.properties` 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 `custom.system.properties` 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`

2.3. Initial Startup

Run the DDF using the appropriate script.

**NIX*

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

Windows

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

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

TIP To run DDF as a service, see Starting as a Service.

2.3.1. Verifying Startup

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

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

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

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

View Status

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

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

Example Bundle Fragment in the Command Line Console

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

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

2.3.2. DDF Directory Contents after Installation and Initial Startup

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

Table 3. DDF Directory Contents

Directory Name	Description
`bin`	Scripts to start, stop, and connect to DDF.
`data`	The working directory of the system – installed bundles and their data
`data/log/ddf.log`	Log file for DDF, logging all errors, warnings, and (optionally) debug statements. This log rolls up to 10 times, frequency based on a configurable setting (default=1 MB)
`data/log/ingest_error.log`	Log file for any ingest errors that occur within DDF.
`data/log/security.log`	Log file that records user interactions with the system for auditing purposes.
`deploy`	Hot-deploy directory – KARs and bundles added to this directory are hot-deployed (Empty upon DDF installation)
`documentation`	HTML and PDF copies of DDF documentation.
`etc`	Directory monitored for addition/modification/deletion of `.config` configuration files or third party `.cfg` configuration files.
`etc/templates`	Template `.config` files for use in configuring DDF sources, settings, etc., by copying to the etc directory.

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

2.3.3. Completing Installation

Upon startup, complete installation from either the Admin Console or the Command Console.

2.3.3.1. Completing Installation from the Admin Console

Upon startup, the installation can be completed by navigating to the Admin Console at https://{FQDN}:{PORT}/admin.

WARNING

Internet Explorer 10 TLS Warning

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

Enabling TLS1.2 in IE10

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

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

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

- Click **Start** to begin.

Setup Types

DDF is pre-configured with several installation profiles.

- Standard Installation: **Recommended**. Includes these applications by default:
  - Admin
  - Catalog
  - Platform
  - Security
  - Solr Catalog

    - Spatial
- Minimum Installation: Includes these applications for a minimum install:
    - Admin
    - Platform
    - Security
- Development: Includes all demo, beta, and experimental applications.

Configure Guest Claim Attributes Page

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

To change this later, see Configuring Guest Claim Attributes.

System Configuration Settings

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

Configure Single Sign On (SSO)

Configure Single Sign On method for SAML or OIDC.

SAML SSO

Enter the URLs for the IdP metatdata and set other options.

IdP Server

Configure $DDF's internal Identity Provider Server.

1. ***SP Metadata***: The metatada of the desired Service Provider to connect to the internal IdP. Can be configured as an HTTPS URL (`https://`), file URL (`file:`), or an XML block (`<md:EntityDescriptor>…</md:EntityDescriptor>`).
2. ***Require Signed AuthnRequests***: Toggle whether or not to require signed `AuthnRequests`. When off, unsigned `AuthnRequests` are rejected.
3. ***Limit RelayStates to 80 Bytes***: Toggle whether or not to restrict the `RelayState` length to 80 bytes. The SAML specification requires a maximum length of 80 bytes. When on, messages with `RelayStates` over 80 bytes are rejected. Only disable if this length is not enforced by the IdP being connected.
4. ***Cookie Expiration Time (minutes)***: Sets the cookie expiration time for Single Sign On, which determines how long the internal IdP caches SAML assertions for later use. This value should match the lifetime of SAML assertions.

IdP Client

Configure handling of incoming requests where SAML authentication is enabled.

1. ***IdP Metadata***: The metadata of the desired IdP to authenticate with. Can be configured as an HTTPS URL (`https://`), file URL (`file:`), or an XML block (`<md:EntityDescriptor>…</md:EntityDescriptor>`).
2. ***Perform User-Agent Check***: If selected, this allows clients that do not support ECP and are not browsers to fall back to PKI, BASIC, and potentially GUEST authentication, if enabled.

OIDC

Select the IdP type desired and set other options as needed.

OIDC Handler Configuration

Configurations relating to handling incoming requests where OIDC authentication is enabled.

1. ***IdP Type***: The type of IdP to authenticate with OIDC.
2. ***Client ID***: Unique ID for the client. This may be provided by the Identity Provider.
3. ***Realm/Tenant***: Realm to use for a multi-tenant environment. This is required for Keycloak or Azure.
4. ***Secret***: A secret shared between the IdP and its clients. This value must match the value set on the Identity Provider.
5. ***Discovery URI***: URI for fetching OP metadata (http://openid.net/specs/openid-connect-discovery-1_0.html). This may be provided by the Identity Provider.
6. ***Base URI***: URI used as a base for different IdP endpoints. This should only be populated if a Discovery URI was not found. This may be provided by the Identity Provider.

7. ***Logout URI***: URI directing to single logout service of the IdP in use.
8. ***Scope***: The OIDC scopes to send in requests to the IdP.
9. ***Use Nonce***: Whether or not to use nonce in JWT.
10. ***Response Type***: The type of OIDC flow to use.
11. ***Response Mode***: Informs the IdP of the mechanism to be used for returning parameters from the Authorization Endpoint.

Finished Page

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

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

2.3.3.2. Completing Installation from the Command Console

In order to install DDF from the Command Console, use the command `profile:install <profile-name>`. The `<profile-name>` should be the desired Setup Type in lowercase letters. To see the available profiles, use the command `profile:list`.

NOTE	This only installs the desired Setup Type. There are other components that can be set up in the Admin Console Installer that cannot be setup on the Command Console. After installing the Setup Type, these other components can be set up as described below.

The Guest Claim Attributes can be configured via the Admin Console after running the `profile:install` command. See Configuring Guest Claim Attributes.

System Settings and Contact Info, as described in System Configuration Settings, can be changed in `<DDF_HOME>/etc/custom.system.properties`. The certificates must be set up manually as described in Managing Keystores and Certificates.

NOTE	The system needs to be restarted after changing any of these settings.

2.3.4. Internet Explorer 11 Enhanced Security Configuration

Below are steps listing all of the changes that DDF requires to run on Internet Explorer 11 and several additional considerations to keep in mind.

1. In the IE11 `Settings` > `Compatibility View Settings` dialog, un-check `Display intranet sites in`

`Compatibility View`.

2. In the `Settings` > `Internet Options` > `Security` tab, `Local intranet` zone:
   a. Click the `Sites` > `Advanced` button, add the current host name to the list, such as https://windows-host-name.domain.edu, and close the dialog.
   b. Make sure the security level for the `Local intranet` zone is set to `Medium-low` in `Custom level…`.
      i. `Enable Protected Mode` is checked by default, but it may need to be disabled if the above changes do not fully resolve access issues.
3. Restart the browser.

NOTE	During installation, make sure to use the host name and not localhost when setting up the DDF's hostname, port, etc.

2.3.5. Firewall Port Configuration

Below is a table listing all of the default ports that DDF uses and a description of what they are used for. Firewalls need to be configured to open these ports in order for external systems to communicate with DDF.

Table 4. Port List

Port	Usage description
8993	https access to DDF admin and search web pages.
8101	For administering DDF instances gives ssh access to the administration console.
1099	RMI Registry Port
44444	RMI Server Port

NOTE	These are the default ports used by DDF. DDF can be configured to use different ports.

2.4. High Availability Initial Setup

This section describes how to complete the initial setup of DDF in a Highly Available Cluster.

Prerequisites

- A failover proxy that can route HTTP traffic according to the pattern described in the Introduction to High Availability. It is recommended that a hardware failover proxy be used in a production environment.
- SolrCloud: See the SolrCloud section for installation and configuration guidance to connect DDF

nodes to SolrCloud.

Once the prerequisites have been met, the below steps can be followed.

NOTE	Unless listed in the High Availability Initial Setup Exceptions section, the normal steps can be followed for installing, configuring, and hardening.

1. Install the first DDF node. See the Installation Section.
2. Configure the first DDF node. See the Configuring Section.
3. Optional: If hardening the first DDF node (excluding setting directory permissions). See the Hardening Section.
4. Export the first DDF node's configurations, install the second DDF node, and import the exported configurations on that node.
5. If hardening, set directory permissions on both DDF nodes. See Setting Directory Permissions.

2.4.1. High Availability Initial Setup Exceptions

These steps are handled differently for the initial setup of a Highly Available Cluster.

2.4.1.1. Failover Proxy Integration

In order to integrate with a failover proxy, the DDF node's system properties (in `<DDF_HOME>/etc/custom.system.properties`) must be changed to publish the correct port to external systems and users. This must be done before installing the first DDF node. See High Availability Initial Setup.

There are two internal port properties that must be changed to whatever ports the DDF uses on its system. Then there are two external port properties that must be changed to whatever ports the failover proxy is forwarding traffic through.

WARNING	Make sure that the failover proxy is already running and forwarding traffic on the chosen ports before starting the DDF. There may be unexpected behavior otherwise.

In the below example, the failover proxy with a hostname of service.org is forwarding https traffic via 8993 and http traffic via 8181. The DDF node runs on 1111 for https and 2222 for http (on the host that it's hosted on). The hostname of the DDF must match the hostname of the proxy.

```
org.codice.ddf.system.hostname=service.org
org.codice.ddf.system.httpsPort=1111
org.codice.ddf.system.httpPort=2222
org.codice.ddf.system.port=${org.codice.ddf.system.httpsPort}
```

```
org.codice.ddf.external.hostname=service.org
org.codice.ddf.external.httpsPort=8993
org.codice.ddf.external.httpPort=8181
org.codice.ddf.external.port=${org.codice.ddf.external.httpsPort}
```

2.4.1.2. Identical Directory Structures

The two DDF nodes need to be under identical root directories on their corresponding systems. On Windows, this means they must be under the same drive.

2.4.1.3. Highly Available Security Auditing

A third party tool must be used to persist the logs in a highly available manner.

- Edit the `<DDF_HOME>/etc/org.ops4j.pax.logging.cfg` file to enable log4j2, following the steps in Enabling Fallback Audit Logging.
- Then put the appropriate log4j2 appender in `<DDF_HOME>/etc/log4j2.xml` to send logs to the chosen third party tool. See Log4j Appenders.

2.4.1.4. Shared Storage Provider

The storage provider must be in a location that is shared between the two DDF nodes and must be highly available. If hardening the Highly Available Cluster, this shared storage provider must be trusted/secured. One way to accomplish this is to use the default Content File System Storage Provider and configure it to point to a highly available shared directory.

2.4.1.5. High Availability Certificates

Due to the nature of highly available environments, localhost is not suitable for use as a hostname to identify the DDF cluster. The default certificate uses localhost as the common name, so this certificate needs to be replaced. The following describes how to generate a certificate signed by the DDF Demo Certificate Authority that uses a proper hostname.

NOTE	This certificate, and any subsequent certificates signed by the Demo CA, are intended for testing purposes only, and should not be used in production.

Certificates need to have Subject Alternative Names (SANs) which include the host for the failover proxy and for both DDF nodes. A certificate with SANs signed by the Demo CA can be obtained by navigating to `<DDF_HOME>/etc/certs/` and, assuming the proxy's hostname is service.org, running the following for UNIX operating systems:

```
./CertNew.sh -cn service.org -san "DNS:service.org"
```

or the following for Windows operating systems:

```
CertNew -cn service.org -san "DNS:service.org"
```

NOTE	Systems that use DDF version 2.11.4 or later automatically get a DNS SAN entry matching the CN without the need to specify the `-san` argument to the `CertNew` command.

More customization for certs can be achieved by following the steps at Creating New Server Keystore Entry with the CertNew Scripts.

2.4.1.6. High Availability Installation Profile

Instead of having to manually turn features on and off, there is a High Availability installation profile. This profile does not show up in the UI Installer, but can be installed by executing `profile:install ha` on the command line **instead** of stepping through the UI Installer. This profile installs all of the High Availability supported features.

3. Configuring

DDF is highly configurable and many of the components of the system can be configured to use an included DDF implementation or replaced with an existing component of an integrating system.

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

Configuration Tools

DDF provides several tools for configuring the system. The Admin Console is a useful interface for configuring applications, their features, and important settings. Alternatively, many configurations can be updated through console commands entered into the Command Console. Finally, configurations are stored in configuration files within the `<DDF_HOME>` directory.

Configuration Outline

While many configurations can be set or changed in any order, for ease of use of this documentation, similar subjects have been grouped together sequentially.

See Keystores and certificates to set up the certificates needed for messaging integrity and authentication. Set up Users with security attributes, then configure data attribute handling, and finally, define the Security Policies that map between users and data and make decisions about access.

Connecting DDF to other data sources, including other instances of DDF is covered in the Configuring

Federation section.

Lastly, see the Configuring for Special Deployments section for guidance on common specialized installations, such as fanout.

3.1. Admin Console Tutorial

The Admin Console is the centralized location for administering the system. The Admin Console allows an administrator to configure and tailor system services and properties. The default address for the Admin Console is https://{FQDN}:{PORT}/admin.

System Settings Tab

The configuration and features installed can be viewed and edited from the **System** tab of the **Admin Console**.

Managing Federation in the Admin Console

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

Viewing Currently Active Applications from Admin Console

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

Table 5. Admin Console Views

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

Application Detailed View

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

Table 6. Individual Application Views

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

Managing Features Using the Admin Console

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

features.

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

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

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Uninstalled features are shown with a **play** arrow under the **Actions** column.
   a. Select the **play** arrow for the desired feature.
   b. The **Status** changes from **Uninstalled** to **Installed**.
5. Installed features are shown with a **stop** icon under the **Actions** column.
   a. Select the **stop** icon for the desired feature.
   b. The **Status** changes from **Installed** to **Uninstalled**.

3.2. Console Command Reference

DDF provides access to a powerful Command Console to use to manage and configure the system.

3.2.1. Feature Commands

Individual features can also be added via the Command Console.

1. Determine which feature to install by viewing the available features on DDF.
   `ddf@local>feature:list`
2. The console outputs a list of all features available (installed and uninstalled). A snippet of the list output is shown below (the versions may differ):

```
State         Version            Name                                     Repository
Description
[installed  ] [2.30.1  ] security-handler-api                    security-services-app-
2.30.1 API for authentication handlers for web applications.
[installed  ] [2.30.1  ] security-core                           security-services-app-
2.30.1 DDF Security Core
[uninstalled] [2.30.1  ] security-expansion                      security-services-app-
2.30.1 DDF Security Expansion
```

```
[installed  ] [2.30.1  ] security-pdp-authz                      security-services-app-
2.30.1 DDF Security PDP.
[uninstalled] [2.30.1  ] security-pep-serviceauthz                security-services-app-
2.30.1 DDF Security PEP Service AuthZ
[uninstalled] [2.30.1  ] security-expansion-user-attributes       security-services-app-
2.30.1 DDF Security Expansion User Attributes Expansion
[uninstalled] [2.30.1  ] security-expansion-metacard-attributes   security-services-app-
2.30.1 DDF Security Expansion Metacard Attributes Expansion
[installed  ] [2.30.1  ] security-realm-saml                       security-services-app-
2.30.1 DDF Security SAML Realm.
[uninstalled] [2.30.1  ] security-jaas-ldap                   security-services-app-
2.30.1 DDF Security JAAS LDAP Login.
[uninstalled] [2.30.1  ] security-claims-ldap           security-services-app-2.30.1
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>)
```

3.2.1.1. Uninstalling Features from the Command Console

1. Check the feature list to verify the feature is installed properly.
   `ddf@local>feature:list`

```
State         Version          Name                          Repository
Description
[installed  ] [2.30.1         ] ddf-core                      ddf-2.30.1
[uninstalled] [2.30.1         ] ddf-sts                       ddf-2.30.1
[installed  ] [2.30.1         ] ddf-security-common           ddf-2.30.1
[installed  ] [2.30.1         ] ddf-resource-impl             ddf-2.30.1
[installed  ] [2.30.1         ] ddf-source-dummy              ddf-2.30.1
```

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  ] [2.30.1         ] ddf-core                     ddf-2.30.1
[uninstalled] [2.30.1         ] ddf-sts                      ddf-2.30.1
[installed  ] [2.30.1         ] ddf-security-common          ddf-2.30.1
[installed  ] [2.30.1         ] ddf-resource-impl            ddf-2.30.1
[uninstalled] [2.30.1         ] ddf-source-dummy             ddf-2.30.1
```

3.3. Configuration Files

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

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

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

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

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

3.3.1. Configuring Global Settings with custom.system.properties

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

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

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

Table 7. Global Settings

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	`javax.net.ssl.trustStore`	String	The trust store used for SSL/TLS connections. Path is relative to `<DDF_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
Headless mode					
Headless Mode	`java.awt.headless`	Boolean	Force java to run in headless mode for when the server doesn't have a display device	`true`	No
Global URL Properties					
Internal Default Protocol	`org.codice.ddf.system.protocol`	String	Default protocol that should be used to connect to this machine.	`https://`	Yes

Title	Property	Type	Description	Default Value	Required
Internal Host	`org.codice.ddf.internal.hostname`	String	The hostname or IP address this system runs on. If the hostname is changed during the install to something other than `localhost` a new keystore and truststore must be provided. See Managing Keystores and Certificates for details.	`localhost`	Yes
Internal HTTPS Port	`org.codice.ddf.system.httpsPort`	String	The https port that the system uses. **NOTE** This **DOES** change the port the system runs on.	`8993`	Yes

Title	Property	Type	Description	Default Value	Required
Internal HTTP Port	`org.codice.ddf.system.HttpPort`	String	The http port that the system uses. **NOTE** This **DOES** change the port the system runs on.	`8181`	Yes
Internal Default Port	`org.codice.ddf.system.port`	String	The default port that the system uses. This should match either the above http or https port. **NOTE** This **DOES** change the port the system runs on.	`8993`	Yes
Internal Root Context	`org.codice.ddf.system.rootContext`	String	The base or root context that services are made available under.	`/services`	Yes

Title	Property	Type	Description	Default Value	Required
External Default Protocol	`org.codice.ddf.external.protocol`	String	Default protocol that should be used to connect to this machine.	`https://`	Yes
External Host	`org.codice.ddf.external.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. If the hostname is changed during the install to something other than `localhost` a new keystore and truststore must be provided. See Managing Keystores and Certificates for details. **NOTE** Does not change the address the system runs on.	`localhost`	Yes

Title	Property	Type	Description	Default Value	Required
HTTPS Port	`org.codice.ddf.external.httpsPort`	String	The https port used to advertise the system. **NOTE** This does not change the port the system runs on.	`8993`	Yes
External HTTP Port	`org.codice.ddf.external.httpPort`	String	The http port used to advertise the system. **NOTE** This does not change the port the system runs on.	`8181`	Yes

Title	Property	Type	Description	Default Value	Required
External Default Port	`org.codice.ddf.external.port`	String	The default port used to advertise the system. This should match either the above http or https port. **NOTE** Does not change the port the system runs on.	`8993`	Yes
External Root Context	`org.codice.ddf.external.context`	String	The base or root context that services are advertised under.	`/services`	Yes
System Information Properties					
Site Name	`org.codice.ddf.system.siteName`	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.	`2.30.1`	Yes
Organization	`org.codice.ddf.system.organization`	String	The organization responsible for this installation of DDF.	`Codice Foundation`	Yes
Registry ID	`org.codice.ddf.system.registry-id`	String	The registry id for this installation of DDF.		No

Title	Property	Type	Description	Default Value	Required
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. See Configuring Thread Pools	`128`	Yes
HTTPS Specific Settings					

Cipher Suites	`https.cipherSuites`	String	Cipher suites to use with secure sockets. If using the JCE unlimited strength policy, use this list in place of the defaults:
			
			.
				`TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,`
				
				`TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,`
				
				`TLS_DHE_RSA_WITH_AES_128_CBC_SHA,`
				
				`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,`
				
				`TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
					No

Https Protocols	`https.protocols`	String	Protocols to allow for secure connections	`TLSv1.2,TLSv1.3`	No
Https Default Protocol	`https.default.protocol`	String	The default protocol used when making outbound HTTPS connections	`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 are audited.	`false`	Yes

Title	Property	Type	Description	Default Value	Required
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
Catalog Source Retry Interval					
Initial Endpoint Contact Interval	`org.codice.ddf.platform.util.http.initialRetryInterval`	Integer	If a Catalog Source is unavailable, try to connect to it after the initial interval has elapsed. After every retry, the interval doubles, up to a given maximum interval. The interval is measured in seconds.	`10`	Yes

Title	Property	Type	Description	Default Value	Required
Maximum Endpoint Contact Interval	`Maximum seconds between attempts to establish contact with unavailable Catalog Source.`	Integer	Do not wait longer than the maximum interval to attempt to establish a connection with an unavailable Catalog Source. Smaller values result in more current information about the status of Catalog Sources, but cause more network traffic. The interval is measured in seconds.	`300`	Yes
File Upload Settings					
File extensions flagged as potentially dangerous to the host system or external clients	`bad.file.extensions`	String	Files uploaded with these bad file extensions have their file names sanitized before being saved. For example, `sample_file.exe` is renamed to `sample_file.bin` upon ingest.	`.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

Title	Property	Type	Description	Default Value	Required
File names flagged as potentially dangerous to the host system or external clients	`bad.files`	String	Files uploaded with these bad file names have their file names sanitized before being saved For example, `crossdomain.xml` is renamed to `file.bin` upon ingest	`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 are rejected from the upload	`text/html, text/javascript, text/x-javascript, application/x-shellscript, text/scriptlet, application/x-msdownload, application/x-msmetafile`	Yes
File names flagged as potentially dangerous to external clients	`ignore.files`	String	Files uploaded with these file names are rejected from the upload	`.DS_Store, Thumbs.db`	Yes
General Solr Catalog Properties					
Solr Catalog Client	`solr.client`	String	Type of Solr configuration	`CloudSolrClient`	Yes
SolrCloud Properties					
Zookeeper Nodes	`solr.cloud.zookeeper`	String	Zookeeper hostnames and port numbers	`localhost:2181`	Yes
Managed Solr Server Properties					
Solr Data Directory	`solr.data.dir`	String	Directory for Solr core files	`<DDF_HOME>/solr/server/solr`	Yes
Solr server HTTP port	`solr.http.port`	Integer	Solr server's port.	`8994`	Yes

Title	Property	Type	Description	Default Value	Required
Solr server URL	`solr.http.url`	String	URL for a HTTP Solr server (required for HTTP Solr)	`-`	Yes
Solr Heap Size	`solr.mem`	String	Memory allocated to the Solr Java process	`2g`	Yes
Encrypted Solr server password	`solr.password`	String	The password used for basic authentication to Solr. This property is only used if the `solr.client` property is `HttpSolrClient` and the `solrBasicAuth` property is `true`.	`admin`	Yes
Solr server username	`solr.username`	String	The username for basic authentication to Solr. This property is only used if the `solr.client` property is `HttpSolrClient` and the `solrBasicAuth` property is `true`.	`admin`	Yes
Use basic authentication for Solr server	`solr.useBasicAuth`	Boolean	If true, the HTTP Solr Client sends a username and password when sending requests to Solr server. This property is only used if the `solr.client` property is `HttpSolrClient`.	`true`	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://{FQDN}:{PORT}/services/csw) could be defined as:

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

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

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

3.3.2. Configuring with .config Files

The DDF is configured using `.config` files. Like the Karaf `.cfg` files, these configuration files must be located in the `<DDF_HOME>/etc/` directory. Unlike the Karaf `.cfg` files, `.config` files must follow the naming convention that includes the *configuration persistence ID* (PID) that they represent. The filenames must be the pid with a `.config` extension. This type of configuration file also supports lists within configuration values (metatype `cardinality` attribute greater than 1) and String, Boolean, Integer, Long, Float, and Double values.

IMPORTANT	This new configuration file format **must** be used for any configuration that makes use of lists. Examples include Web Context Policy Manager (`org.codice.ddf.security.policy.context.impl.PolicyManager.config`) and Guest Claims Configuration (`ddf.security.guest.realm.config`).
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. DDF 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 and processed. Changes to these configurations from the Admin Console or otherwise are persisted in the original configuration file in the `<DDF_HOME>/etc` directory.

3.4. Configuring User Access

DDF does not define accounts or types of accounts to support access. DDF uses an *attribute based access control* (**ABAC**) model. For reference, ABAC systems control access by evaluating rules against the attributes of the entities (*subject* and *object*), actions, and the environment relevant to a request.

DDF can be configured to access many different types of user stores to manage and monitor user access.

3.4.1. Configuring Guest Access

Unauthenticated access to a secured DDF system is provided by the **Guest** user. By default, DDF allows guest access.

Because DDF does not know the identity of a Guest user, it cannot assign security attributes to the Guest. The administrator must configure the attributes and values (the "claims") to be assigned to Guests. The Guest Claims become the default minimum attributes for every user, both authenticated and unauthenticated. Even if a user claim is more restrictive, the guest claim grants access, so ensure the guest claim is only as permissive as necessary.

The **Guest** user is uniquely identified with a Principal name of the format `Guest@UID`. The unique identifier is assigned to a Guest based on its source IP address and is cached so that subsequent Guest accesses from the same IP address within a 30-minute window get the same unique identifier. To support administrators' need to track the source IP Address for a given Guest user, the IP Address and unique identifier mapping are audited in the security log.

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

3.4.1.1. Denying Guest User Access

To disable guest access for all contexts, use the Web Context Policy Manager configuration and uncheck the Guest checkbox. Only authorized users are then allowed to continue to the Search UI page.

3.4.1.2. Allowing Guest User Access

Guest authentication must be enabled and configured to allow guest users. Once the guest user is configured, the redaction and filtering of metadata is done for the guest user the same way it is done for normal users.

To enable guest authentication for a context, use the Web Context Policy Manager configuration to select **Allow Guest Access**.

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

3.4.1.2.1. Configuring Guest Interceptor if Allowing Guest Users

- **Required Step for Security Hardening**

If a legacy client requires the use of the secured SOAP endpoints, the guest interceptor should be

configured. Otherwise, the guest interceptor and `public` endpoints should be uninstalled for a hardened system.

To uninstall the guest interceptor and `public` endpoints: . Navigate to the **Admin Console**. . Select the **System** tab. . Open the **Features** section. . Search for **security-interceptor-guest**. . Click the **Uninstall** button.

3.4.1.2.2. Configuring Guest Claim Attributes

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

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

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select the **Security Guest Claims Handler**.
5. Add any additional attributes desired for the guest user.
6. Save changes.

3.4.2. Configuring REST Services for Users

If using REST services or connecting to REST sources, several configuration options are available.

DDF can be configured to support an external SAML IdP or no IdP at all. The following diagram shows the configuration options.

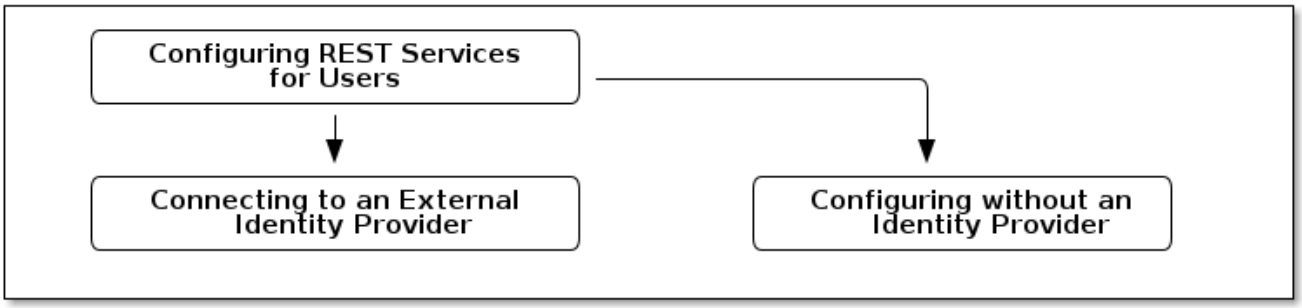

Figure 2. REST Services Configuration Options

3.4.2.1. Configuring OpenID Connect (OIDC) and OAuth 2.0

To use OpenID Connect (OIDC) and OAuth 2.0, DDF needs to be connected to an external Identity

Provider (IdP) which supports these protocols.

3.4.2.1.1. OIDC

OIDC is used to authenticate (or log in) a user. To use this protocol in DDF, DDF needs the external IdP's information.

To connect to an external IdP,

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **OIDC Handler Configuration**.
5. Populate the fields with the external IdP's information. For more detail, see the `OIDC Handler Configuration` section under Configure Single Sign On.

Once connected, the Web Context Policy Manager needs to be updated. to do so,

1. Under the **Configuration** tab in the **Security** application
2. Select the **Web Context Policy Manager**.
3. Under `Authentication Types for Web Pages` and `Authentication Types for Endpoints` add OIDC.

3.4.2.1.2. OAuth 2.0

OAuth 2.0 is an authorization protocol and DDF can use it when federating. When a user queries a source that is configured to use this protocol, DDF forwards the user's information (access token) with the request. If a source can be configured to use OAuth 2.0, `OAuth 2.0` appears as an option under `Authentication Type` in the source's configuration.

To configure a source to use OAuth 2.0, under the source's configuration,

1. Change the `Authentication Type` to `OAuth 2.0`.
2. Set the `OAuth Discovery Url` to the discovery URL where the OAuth provider's metadata is hosted.
3. Set the `OAuth Client ID` to the ID given to DDF when it was registered with the OAuth provider`.
4. Set the `OAuth Client Secret` to the secret given to DDF when it was registered with the OAuth provider`.
5. Set the `OAuth Flow` to the OAuth 2.0 flow to use when federating.

NOTE	If DDF is federating with another DDF, the DDF receiving the federated request should be connected to an external IdP and have `/services` protected by that IdP (that is, have `/service=OIDC`).

3.4.2.2. Connecting to an External SAML Identity Provider

To connect to an external SAML Identity Provider,

1. Provide the external SAML IdP with DDF's Service Provider (SP) metadata. The SP metadata can found at `https://<FQDN>:<PORT>/services/saml/sso/metadata`.
2. Replace the IdP metadata field in DDF.
   a. Navigate to the **Admin Console**.
   b. Select the **Security** application.
   c. Select the **Configuration** tab.
   d. Select **SAML Handler**.
   e. Populate the **IdP Metadata** field with the external IdP's metadata.

NOTE	The certificate that the external IdP uses for signing needs to be added to the DDF's keystore. See Updating Key Store / Trust Store via the Admin Console for details.
NOTE	DDF may not interoperate successfully with all IdPs. To idenify the ones it can interoperate with use the The Security Assertion Markup Language (SAML) Conformance Test Kit (CTK)

3.4.2.3. Configuring Without SAML

To configure DDF to not use a SAML Identity Provider (IdP),

1. Disable the 'security-handler-saml' feature.
   a. Navigate to the **Admin Console**.
   b. Select the **System** tab.
   c. Select the **Features** tab.
   d. Uninstall the `security-handler-saml` feature.
2. Change the Authentication Type if it is SAML.
   a. Navigate to the **Admin Console**.
   b. Select the **Security** application.
   c. Select the **Configuration** tab.
   d. Select **Web Context Policy Manager**
   e. Under **Authentication Types for Web Pages** and **Authentication Types for Endpoints**, remove the SAML authentication type.

3.4.2.4. Configuring Multi Factor Authentication

Multi-factor authentication, sometimes referred to as Two-factor authentication, allows for greater security. It does this by requiring users to provide multiple proofs of identity, typically through something they know (such as a password), and something they have/are (such as a randomly generated pin number sent to one of their personal devices). The IdP that comes with DDF does not support multi-factor authentication by default.

Keycloak can be used to help setup and configure multi-factor authentication. See Connecting to an External Identity Provider on how to initially hookup Keycloak.

Configuring Keycloak for MFA

1. Download and install Keycloak from here: Keycloak Downloads {external link}
2. See Choosing an Operating Mode {external link} to choose a specific operation mode.
3. Set up an Admin User following these steps here: Server Admin Initialization {external link}
4. Refer to OTP Policies {external link} for how to set up multi-factor authentication using supported authentication tools such as **FreeOTP** and **Google Authenticator**.

See the Keycloak Documentation {external link} for more information and details about how to configure Keycloack for multi-factor authentication.

3.4.3. Connecting to 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, from the Command Console to set passwords for your LDAP server. Then change the LDAP Bind User Password in the Security STS LDAP and Roles Claims Handler configurations to use the encrypted password.

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 is 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.

See the Security STS LDAP and Roles Claims Handler for all possible configurations.

3.4.4. Updating System Users

System users

By default, all system users are located in the `<DDF_HOME>/etc/users.properties` and `<DDF_HOME>/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 are 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 ends 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 is not too great when using this feature.
WARNING	If your data contains security markings, and these markings are being parsed out into the metacard security attributes 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 results in an error being

returned for any ingest request.

WARNING

The following attribute values are not allowed:

- `null`
- `""`
- a non-String (for example, `100` or `false`)
- an array including any of the above
- `[]`

Additionally, attribute names should not be repeated, and the order that the attributes are defined and the order of values within an array is ignored.

3.4.5. Restricting Access to Admin Console

- **Required Step for Security Hardening**

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.

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

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select the **Web Context Policy Manager**.
   a. A dialogue allows you to edit DDF access restrictions.
   b. If your infrastructure supports multiple authentication methods, you may specify them here.
   c. Role requirements may be enforced by configuring the required attributes for a given context.
   d. The white listed contexts allows child contexts to be excluded from the authentication constraints of their parents.

3.4.5.1. Restricting Feature, App, Service, and Configuration Access

- **Required Step for Security Hardening**

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

they are restricted.

1. Navigate to the **Admin Console**.
2. Select the **Admin** application.
3. Select the **Configuration** tab.
4. Select the **Admin Configuration Policy**.
5. To add a feature or app permission:
   a. Add a new field to "Feature and App Permissions" in the format of:

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

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

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

6. To add a configuration permission:
   a. Add a new field to "Configuration Permissions" in the format of:

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

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

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

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

3.4.6. Restricting Access to Features, Apps, Services, and Configurations

- **Required Step for Security Hardening**

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

1. Navigate to the **Admin Console**.
2. Select the **Admin** application.
3. Select the **Configuration** tab.
4. Select the **Admin Configuration Policy**.
5. To add a feature or app permission:

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

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

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

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

6. To add a configuration permission:

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

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

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

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

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

3.4.7. Removing Default Users

- **Required Step for Security Hardening**

The default security configuration uses a property file located at `<DDF_HOME>/etc/users.properties` to store users and passwords. A hardened system removes this file and manage all users externally, via an LDAP server or by other means.

NOTE	*Default Users are an Insecure Default* The Admin Console has an insecure default warning if the default users are not removed.

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

The deletion of the default users in the `users.properties` file can be done automatically after 72 hours. This feature can be found at **Admin Console** → **Admin** → **Default Users Deletion Scheduler** → **Enable default users automatic deletion**.

WARNING	Once the default users are removed, the `<DDF_HOME>/bin/client` and `<DDF_HOME>/bin/client.bat` scripts do not work. If SSH access to the Karaf shell is to be supported, edit the file `org.apache.karaf.shell.cfg` in the `<INSTALL_HOME>/etc`

directory, changing the value of the `sshRealm` property from `karaf` to `ldap`.

NOTE

Emergency Use of `users.properties` file

Typically, the DDF does not manage passwords. Authenticators are stored in an external identity management solution. However, administrators may temporarily use a `users.properties` file for emergencies.

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

- The use of this account should be for as short a time as possible.
- The default username/password of “admin/admin” should not be used.
- All organizational standards for password complexity should still apply.
- The password should be encrypted. For steps on how, see the section "Passwords Encryption" at https://karaf.apache.org/manual/latest/security.

NOTE

Compliance Reviews

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

3.4.8. Disallowing Login Without Certificates

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

- Navigate to the **Admin Console**.
- Select **Security**.
- Select **Web Context Policy Manager**.
- Add a policy for each context requiring restriction.
  - For example, just configuring `PKI` as the authentication type for web pages disallows login without certificates to DDF.
- Click **Save**.

NOTE Ensure certificates comply with organizational hardening policies.

3.4.9. Managing Certificate Revocation

- **Required Step for Security Hardening**

For hardening purposes, it is recommended to implement a way to verify a Certificate Revocation List (CRL) at least daily or an Online Certificate Status Protocol (OCSP) server.

3.4.9.1. Managing a Certificate Revocation List (CRL)

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

3.4.9.1.1. Creating a CRL

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

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

> **NOTE**
>
> *Windows and OpenSSL*
>
> Windows does not include OpenSSL by default. For Windows platforms, a additional download of OpenSSL or an alternative is required.

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

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

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

Viewing a CRL

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

3.4.9.1.2. Enabling Certificate Revocation

> **NOTE**
>
> Enabling CRL revocation or modifying the CRL file requires a restart of DDF to apply updates.

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

Adding this property also enables CRL revocation for any context policy implementing PKI authentication. For example, setting PKI as the authentication type for web pages in the Web Context Policy Manager disables basic authentication and require a certificate. If a certificate is not in the CRL,

it is allowed through. If not, it receives a 401 error. If no certificate is provided, and guest access is enabled on the web context policy, guest access is granted.

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 also enabless it for the STS, and also requires a restart.

If the CRL cannot be placed in <DDF_HOME>/etc/keystores but can be accessed via an **HTTPS** URL:

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **Certificate Revocation List (CRL)**
5. Add the **HTTPS** URL under **CRL URL address**
6. Check the **Enable CRL via URL** option

A local CRL file is created and the `encryption.properties` and `signature.properties` files are set as mentioned above.

Add Revocation to a Web Context

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

1. After enabling revocation (see above), open the **Web Context Policy Manager**.
2. Add or modify the **Authentication Types for Web Pages** and **Authentication Types for Endpoints** configurations to use PKI.
3. If guest access is also required, check the `Allow Guest Access` box in the policy.

With guest access, a user with a revoked certificate is given a 401 error, but users without a certificate is 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 becomes active after specifying the CRL file path, or the URL for the CRL, in the `encryption.properties` file and restarting DDF.

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

Adding Revocation to an Endpoint

NOTE	This section explains how to add CXF's CRL revocation method to an endpoint and not

the CRL revocation method in the `PKIHandler`.

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

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

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

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

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

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

Verifying Revocation

A **Warning** similar to the following is 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]

[ ... section removed for space]

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
```

3.4.9.2. Managing an Online Certificate Status Protocol (OCSP) Server

An Online Certificate Status Protocol is a protocol used to verify the revocation status of a certificate. An OCSP server can be queried with a certificate to verify if it is revoked.

The advantage of using an OCSP Server over a CRL is the fact that a local copy of the revoked certificates is not needed.

3.4.9.2.1. Enabling OCSP Revocation

1. Navigate to the **Admin Console**.
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select **Online Certificate Status Protocol (OCSP)**
5. Add the URL of the OCSP server under **OCSP server URL**.
6. Check the **Enable validating a certificate against an OCSP server** option.

NOTE	If an error occurs while communicating with the OCSP server, an alert is posted to the Admin Console. Until the error is resolved, certificates are not verified against the server.

3.5. Configuring Data Management

Data ingested into DDF has security attributes that can be mapped to users' permissions to ensure proper access. This section covers configurations that ensure only the appropriate data is contained in or exposed by DDF.

3.5.1. Configuring Solr

The default catalog provider for DDF is Solr. If using another catalog provider, see Changing Catalog

Providers.

3.5.1.1. Configuring Solr Catalog Provider Synonyms

When configured, text searches in Solr utilize synonyms when attempting to match text within the catalog. Synonyms are used during keyword/anyText searches as well as when searching on specific text attributes when using the `like` / `contains` operator. Text searches using the `equality` / `exact match` operator do not utilize synonyms.

Solr utilizes a `synonyms.txt` file which exists for each Solr core. Synonym matching is most pertinent to metacards which are contained within 2 cores: `catalog` and `metacard_cache`.

3.5.1.1.1. Defining synonym rules in the Solr Provider

- Edit the `synonyms.txt` file under the `catalog` core. For each synonym group you want to define, add a line with the synonyms separated by a comma. For example:

```
United States, United States of America, the States, US, U.S., USA, U.S.A
```

- Save the file
- Repeat the above steps for the `metacard_cache` core.
- Restart the DDF.

NOTE	Data does not have to be re-indexed for the synonyms to take effect.

3.5.1.2. Hardening Solr

Follow instructions on Securing Solr.

3.5.1.2.1. Configuring Solr Encryption

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

3.5.1.3. Accessing the Solr Admin UI

The Solr Admin UI for Solr server configurations can be accessed from a web browser. See Using the Solr Administration User Interface for more details.

3.5.2. Changing Catalog Providers

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

This scenario assumes DDF is already running.

Uninstall Catalog Provider (if installed).

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

Install the new Catalog Provider

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

3.5.3. Changing Hostname

By default, the STS server, STS client and the rest of the services use the system property `org.codice.ddf.system.hostname` which is defaulted to 'localhost' and not to the fully qualified domain name of the DDF instance. Assuming the DDF instance is providing these services, the configuration must be updated to use the **fully qualified domain name** as the service provider. If the DDF is being accessed from behind a proxy or load balancer, set the system property `org.codice.ddf.external.hostname` to the hostname users use to access the DDF.

This can be changed during Initial Configuration or later by editing the `<DDF_HOME>/etc/custom.system.properties` file.

3.5.4. Configuring Errors and Warnings

DDF performs several types of validation on metadata ingested into the catalog. Depending on need, configure DDF to act on the warnings or errors discovered.

3.5.4.1. Enforcing Errors or Warnings

Prevent data with errors or warnings from being ingested at all.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select **Configuration**.
4. Select **Metacard Validation Marker Plugin**.

5. Enter **ID** of each validator to enforce.
6. Select **Enforce errors** to prevent ingest for errors.
7. Select **Enforce warnings** to prevent ingest for warnings.

3.5.4.2. Hiding Errors or Warnings from Queries

Prevent invalid metacards from being displayed in query results, unless specifically queried.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select **Configuration**.
4. Select **Catalog Federation Strategy**.
5. Deselect **Show Validations Errors** to hide metacards with errors.
6. Deselect **Show Validations Warnings** to hide metacards with warnings.

3.5.4.3. Hiding Errors and Warnings from Users Based on Role

- **Required Step for Security Hardening**

Prevent certain users from seeing data with certain types of errors or warnings. Typically, this is used for security markings. If the **Metacard Validation Filter Plugin** is configured to **Filter errors** and/or **Filter warnings**, metacards with errors/warnings are hidden from users without the specified user attributes.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select **Configuration**.
4. Select **Metacard Validation Filter Plugin**.
5. For **Attribute map**, enter both the metacard `SECURITY` attribute to filter and the user attribute to filter.
   a. The default attribute for viewing invalid metacards is `invalid-state`
      i. `invalid-state=<USER ROLE>`.
      ii. Replace `<USER ROLE>` with the roles that should be allowed to view invalid metacards.

NOTE	To harden the system and prevent other DDF systems from querying invalid data in the local catalog, it is recommended to create and set user roles that are unique to the local system (ie. a user role that includes a UUID).

6. Select **Filter errors** to filter errors. Users without the `invalid-state` attribute cannot see metacards with errors.

7. Select **Filter warnings** to filter warnings. Users without the `invalid-state` attribute cannot see metacards with warnings.

3.5.5. Configuring Product Caching

To configure product caching:

1. Navigate to the **Admin Console**.
2. Select Catalog.
3. Select **Configuration**.
4. Select **Resource Download Settings**.
5. Select / Deselect **Enable Product Caching**.

See Resource Download Settings configurations for all possible configurations.

3.5.6. Content Directory Monitor

The Content Directory Monitor (CDM) provides the capability to easily add content and metacards into the Catalog by placing a file in a directory.

3.5.6.1. Installing the Content Directory Monitor

The Content Directory Monitor is installed by default with a standard installation of the Catalog application.

3.5.6.2. Configuring Permissions for the Content Directory Monitor

TIP	If monitoring a WebDav server, then adding these permissions is not required and this section can be skipped.

Configuring a Content Directory Monitor requires adding permissions to the Security Manager before CDM configuration.

Configuring a CDM requires adding read and write permissions to the directory being monitored. The following permissions, replacing <DIRECTORY_PATH> with the path of the directory being monitored, are required for each configured CDM and should be placed in the CDM section inside <DDF_HOME>/security/configurations.policy.

WARNING	*Adding New Permissions* After adding permissions, a system restart is required for them to take effect.

```
. permission java.io.FilePermission "<DIRECTORY_PATH>", "read";
```

```
. permission java.io.FilePermission "<DIRECTORY_PATH>${/}-", "read, write";
```

Trailing slashes after `<DIRECTORY_PATH>` have no effect on the permissions granted. For example, adding a permission for `"${/}test${/}path"` and `"${/}test${/}path${/}"` are equivalent. The recursive forms `"${/}test${/}path${/}-"`, and `"${/}test${/}path${/}${/}-"` are also equivalent.

Line 1 gives the CDM the permissions to read from the monitored directory path. Line 2 gives the CDM the permissions to recursively read and write from the monitored directory path, specified by the directory path's suffix `"${/}-"`.

If a CDM configuration is deleted, then the corresponding permissions that were added should be deleted to avoid granting unnecessary permissions to parts of the system.

3.5.6.3. Configuring the Content Directory Monitor

IMPORTANT

Content Directory Monitor Permissions

When configuring a Content Directory Monitor, make sure to set permissions on the new directory to allow DDF to access it. Setting permissions should be done **before** configuring a CDM. Also, don't forget to add permissions for resources outside of the monitored directory. See Configuring Permissions for the Content Directory Monitor for in-depth instructions on configuring permissions.

NOTE

If there's a metacard that points to a resource outside of the CDM, then you must configure the URL Resource Reader to be able to download it.

WARNING

Monitoring Directories In Place

If monitoring a directory in place, then the URL Resource Reader must be configured prior to configuring the CDM to allow reading from the configured directory. This allows the Catalog to download the resources.

Configure the CDM from the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select **Catalog Content Directory Monitor**.

See Content Directory Monitor configurations for all possible configurations.

3.5.6.4. Using the Content Directory Monitor

The CDM processes files in a directory, and all of its sub-directories. The CDM offers three options:

- Delete

- Move
- Monitor in place

Regardless of the option, the DDF takes each file in a monitored directory structure and creates a metacard for it. The metacard is linked to the file. The behavior of each option is given below.

Delete

- Copies the file into the Content Repository.
- Creates a metacard in the Catalog from the file.
- **Erases** the original file from the monitored directory.

Move

- Copies the file into the directory `.\ingested` **(this doubles the disk space used)**
- Copies the file into the Content Repository.
- Creates a metacard in the Catalog from the file.
- **Erases** the original file from the monitored directory.

Monitor in place

- Creates a metacard in the Catalog from the file.
- Creates a reference from the metacard to the original file in the monitored directory.
- If the original file is deleted, the metacard is removed from the Catalog.
- If the original file is modified, the metacard is updated to reflect the new content.
- If the original file is renamed, the old metacard is deleted and a new metacard is created.

Parallel Processing

The CDM supports parallel processing of files (up to 8 files processed concurrently). This is configured by setting the number of **Maximum Concurrent Files** in the configuration. A maximum of 8 is imposed to protect system resources.

Read Lock

When the CDM is set up, the directory specified is continuously scanned, and files are locked for processing based on the **ReadLock Time Interval**. This does not apply to the **Monitor in place** processing directive. Files are not ingested without having a ReadLock that has observed no change in the file size. This is done so that files that are in transit are not ingested prematurely. The interval should be dependent on the speed of the copy to the directory monitor (ex. network drive vs local disk). For local files, the default value of 500 milliseconds is recommended. The recommended interval for network drives is 1000 - 2000 milliseconds. If the value provided is less than 100, 100 milliseconds is used. It is also recommended that the **ReadLock Time Interval** be set to a lower amount of time when the **Maximum Concurrent Files** is set above 1 so that files are locked in a timely manner and processed as soon as possible. When a higher **ReadLock Time Interval** is set, the time it takes for files to be processed is increased.

Attribute Overrides

The CDM supports setting metacard attributes directly when DDF ingests a file. Custom overrides are entered in the form:

```
attribute-name=attribute-value
```

For example, to set the contact email for all metacards, add the attribute override:

```
contact.point-of-contact-email=doctor@clinic.com
```

Each override sets the value of a single metacard attribute. To set the value of an additional attribute, select the "plus" icon in the UI. This creates an empty line for the entry.

To set multi-valued attributes, use a separate override for each value. For example, to add the keywords *PPI* and *radiology* to each metacard, add the custom attribute overrides:

```
topic.keyword=PPI
topic.keyword=radiology
```

Attributes are only overridden if they are part of the metacard type or are injected.

All attributes in the catalog taxonomy tables are injected into all metacards by default and can be overridden.

IMPORTANT	If an overridden attribute is not part of the metacard type or injected the attribute is not added to the metacard.

For example, if the metacard type contains contact email,

```
contact.point-of-contact-email
```

but the value is not currently set, adding an attribute override sets the attribute value. To override attributes that are not part of the metacard type, attribute injection can be used.

Blacklist

The CDM blacklist uses the "bad.files" and "bad.file.extensions" properties from the custom.system.properties file in "etc/" in order to prevent malicious or unwanted data from being ingested into DDF. While the CDM automatically omits hidden files, this is particularly useful when an operating system automatically generates files that should not be ingested. One such example of this is "thumbs.db" in Windows. This file type and any temporary files are included in the blacklist.

Errors

If the CDM fails to read the file, an error is logged in the CDM log. If the directory monitor is configured to **Delete** or **Move**, the original file is also moved to the `\.errors` directory.

Logging

CDM sends log entries detailing the processing of files in the directory to `cdm.log`.

TIP

The log level for CDM can be set with the following console command. At the DEBUG level, CDM periodically logs the list of files still currently processing.

```
log:set DEBUG cdmLogger
```

Other

- Multiple directories can be monitored. Each directory has an independent configuration.
- To support the monitoring in place behavior, DDF indexes the files to track their names and modification timestamps. This enables the Content Directory Monitor to take appropriate action when files are changed or deleted.
- The Content Directory Monitor recursively processes all subdirectories.

3.5.7. Configuring System Usage Message

The Platform UI configuration contains the settings for displaying messages to users at login or in banners in the headers and footers of all pages. For, example this configuration can provide warnings that system usage is monitored or controlled.

Configuring System Usage Message

1. Navigate to the **Admin Console**.
2. Select the **Platform** application.
3. Select **Configuration**.
4. Select **Platform UI Configuration**.
5. Select **Enable System Usage Message**.
6. Enter text in the remaining fields and save.

See the Platform UI for all possible configurations.

3.5.8. Configuring Data Policy Plugins

Configure the data-related policy plugins to determine the accessibility of data held by DDF.

3.5.8.1. Configuring the Metacard Attribute Security Policy Plugin

The Metacard Attribute Security Policy Plugin combines existing metacard attributes to make new attributes and adds them to the metacard.

1. Navigate to the **Admin Console**.

2. Select the **Catalog** application tile
3. Select the **Configuration** tab
4. Select the **Metacard Attribute Security Policy Plugin**.

Sample configuration of the Metacard Attribute Security Policy Plugin.

To configure the plugin to combine the attributes `sourceattribute1` and `sourceattribute2` into a new attribute `destinationattribute1` using the union, enter these two lines under the title **Metacard Union Attributes**

Metacard Union Attributes
`sourceattribute1=destinationattribute1`
`sourceattribute2=destinationattribute1`

See Metacard Attribute Security Policy Plugin configurations for all possible configurations.

3.5.8.2. Configuring the Metacard Validation Marker Plugin

By default, the Metacard Validation Marker Plugin marks metacards with validation errors and warnings as they are reported by each metacard validator and then allows the ingest. To prevent the ingest of certain invalid metacards, the `Metacard Validity Marker` plugin can be configured to "enforce" one or more validators. Metacards that are invalid according to an "enforced" validator are not ingested.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **Metacard Validity Marker Plugin**.
    a. If desired, enter the ID of any metacard validator to enforce. This prevents ingest of metacards that fail validation.
    b. If desired, check **Enforce Errors** or **Enforce Warnings**, or both.

See Metacard Validity Marker Plugin configurations for all possible configurations.

3.5.8.3. Configuring the Metacard Validity Filter Plugin

The Metacard Validity Filter Plugin determines whether metacards with validation errors or warnings are filtered from query results.

1. Navigate to the **Admin Console**.

2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **Metacard Validity Filter Plugin**.
    a. Check **Filter Errors** to hide metacards with errors from users.
    b. Check **Filter Warnings** to hide metacards with warnings from users.

See Metacard Validity Filter Plugin configurations for all possible configurations.

3.5.8.4. Configuring the XML Attribute Security Policy Plugin

The XML Attribute Security Policy Plugin finds security attributes contained in a metacard's metadata.

1. Navigate to the Admin Console.
2. Select the **Catalog** application tile.
3. Select the **Configuration** tab.
4. Select the **XML Attribute Security Policy Plugin** configuration.

See XML Attribute Security Policy Plugin configurations for all possible configurations.

3.5.9. Configuring Data Access Plugins

Configure access plugins to act upon the rules and attributes configured by the policy plugins and user attributes.

3.5.9.1. Configuring the Security Audit Plugin

The Security Audit Plugin audits specific metacard attributes.

To configure the Security Audit Plugin:

1. Navigate to the **Admin Console**.
2. Select **Catalog** application.
3. Select **Configuration** tab.
4. Select **Security Audit Plugin**.

Add the desired metacard attributes that are audited when modified.

See Security Audit Plugin configurations for all possible configurations.

3.6. Configuring Security Policies

User attributes and Data attributes are matched by security policiesSecurity policies defined within DDF.

3.6.1. Configuring the Web Context Policy Manager

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

- the type of authentication for web pages (such as /admin and /search) and endpoints (context paths that start with /services)
- any user attributes required for authorization
- a list of whitelisted contexts

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

3.6.1.1. Guest Access

Guest access is a toggleable configuration. Enabling guest access causes all users to be assigned a guest principal for use throughout the entire system. The guest principal is used either by itself or along with any other principals acquired from configured authentication types.

3.6.1.2. Session Storage

Enabling session storage allows the system to persist the user login through the use of cookies. Note that the `SAML` and `OIDC` authentication types require session storage to be enabled.

3.6.1.3. Authentication Types

Through the Web Context Policy Manager, authentication types for DDF may be configured with the following format:

```
<AUTH_TYPE>|<AUTH_TYPE|...
```

Authentication types can be configured separately for: * web pages (these are all context paths that do not start with `/services` such as `/admin` and `/search`) * endpoints (these are all context paths that start with `/services`)

Configuring separate authentication types for specific contexts is supported through the `web-context-policy-config.properties` configuration file. * Navigate to `<DDF_HOME>/etc/` * Edit the `web-context-policy-config.properties` file * Restart DDF

The `web-context-policy-config.properties` file format is:

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

WARNING	If the file-based configuration is used, the **authentication type** configuration for web pages and endpoints in the Admin Console is ignored. All other configurations in the Web Context Policy Manager such as Guest Access, Required Attributes, and White Listed Contexts are **not** ignored.

Table 8. Default Types of Authentication

Authentication Type	Description
`BASIC`	Activates basic authentication.
`PKI`	Activates public key infrastructure authentication.
`SAML`	Activates single-sign on (SSO) across all REST endpoints that use SAML.
`OIDC`	Activates single-sign on (SSO) across all REST endpoints that use OIDC.

3.6.1.3.1. Terminating and Non-Terminating Authentication Types

Terminating authentication types are authentication types where, once hit, must either allow or forbid access to the system. No other authentication types are checked once a terminating authentication type is hit.

Non-Terminating authentication types are authentication types where, once hit, must first verify that the client supports the authentication type's method of obtaining credentials. If the client supports the non-terminating authentication type's method of obtaining credentials, it either allows or forbids access to the system. However if the client does not support the non-terminating authentication type's method of obtaining credentials, the system continues to the next configured authentication type.

`PKI` is the only non-terminating authentication type. `BASIC`, `OIDC`, and `SAML` are terminating authentication types.

For example: assume DDF is protected by the terminating `SAML` authorization type. The system first checks to see if the client supports the acquisition of SAML credentials.

- If the connecting client is a browser, the system can acquire SAML credentials.
- If the connecting client is a machine that supports SAML ECP, the system can acquire SAML credentials.
- If the connecting client is a machine that does not support SAML ECP, the system cannot acquire SAML credentials.

If the system can acquire SAML credentials from the client, the system attempts to acquire said

credentials and either allow or forbid access. If the system cannot acquire SAML credentials from the client, the system forbids access.

Otherwise, if DDF is protected by the non-terminating `PKI` authorization type and the system cannot acquire PKI certificates from the client, the system continues to the next configured authentication type.

3.6.1.4. 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

3.6.1.5. White Listed Contexts

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

3.6.2. Configuring Catalog Filtering Policies

Filtering is the process of evaluating security markings on data resources, comparing them to the users permissions and protecting resources from inappropriate access.

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

3.6.2.1. Setting Internal Policies

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

With the `security-pdp-authz` feature configured in this way, the above 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.

3.6.2.2. Setting XACML Policies

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 looks for XACML policies every 60 seconds.

See Developing XACML Policies for more information about custom XACML policies.

3.6.2.3. Catalog Filter Policy Plugins

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

3.7. Configuring User Interfaces

DDF has several user interfaces available for users.

3.8. Environment Hardening

- **Required Step for Security Hardening**

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

3.8.1. Known Issues with Environment Hardening

The session timeout should be configured longer than the UI polling time or you may get session timeout errors in the UI.

Protocol/ Type	Risk	Mitigation
JMX	tampering, information disclosure, and unauthorized access	• Stop the management feature using the command line console: `feature:stop management`.

File System Access	tampering, information disclosure, and denial of service	Set OS File permissions under the `<DDF_HOME>` directory (for example, `/deploy` or `/etc`) to ensure unauthorized viewing and writing is not allowed. `If Caching is installed:` • Set permissions for the installation directory `/data/product-cache` such that only the DDF process and users with the appropriate permissions can view any stored product. • Caching can be turned off as well to mitigate this risk. ◦ To disable caching, navigate to Admin Console. ◦ Select the **Catalog** application. ◦ Select **Resource Download Settings**. ◦ Uncheck the `Enable Product Caching` box. • Install Security to ensure only the appropriate users are accessing the resources. ◦ Navigate to the Admin Console ◦ Select **Manage**. ◦ Install the Security application, if applicable. • Cached files are written by the user running the DDF `process/application`. On system: ensure that not everyone can change ACLs on your object.

SSH	tampering, information disclosure, and denial of service	By default, SSH access to DDF is only enabled to connections originating from the same host running DDF. For remote access to DDF, first establish an SSH session with the host running DDF. From within that session, initiate a new SSH connection (to **localhost**), and use the `sshPort` as configured in the file `<DDF_HOME>/etc/org.apache.karaf.shell.cfg`. To allow direct remote access to the DDF shell from any host, change the value of the `sshHost` property to `0.0.0.0` in the `<DDF_HOME>/etc/org.apache.karaf.shell.cfg` file. SSH can also be authenticated and authorized through an external Realm, such as LDAP. This can be accomplished by editing the `<DDF_HOME>/etc/org.apache.karaf.shell.cfg` file and setting the value for `sshRealm` to the appropriate realm, such as `ldap`. No restart of DDF is necessary after this change. By definition, all connections over SSH are authenticated and authorized and secure from eavesdropping. **WARNING**: Enabling SSH exposes your file system because any user with access to your DDF shell has read/write/execute access to all directories and files accessible to your installation user. **Because of this, SSH is not recommended in a secure environment and should be turned off in a fully hardened system.** Set `karaf.shutdown.port=-1` in `<DDF_HOME>/etc/custom.properties` or `<DDF_HOME>/etc/config.properties`.

SSL/TLS	man-in-the-middle, information disclosure	Update the `<DDF_HOME>/etc/org.ops4j.pax.web.cfg` file to add the entry `org.ops4j.pax.web.ssl.clientauth.needed=true`. **WARNING** Setting this configuration may break compatibility to legacy systems that do not support two-way SSL. **WARNING** Setting this configuration requires a certificate to be installed in the browser.
Session Inactivity Timeout	unauthorized access	Update the Session configuration to have no greater than a 10 minute Session Timeout. • Navigate to the **Admin Console**. • Select the **Security** application. • Select the **Configuration** tab. • Select `Session`. • Set `Session Timeout (in minutes)` to `10` (or less).

Shell Command Access	command injection	By default, some shell commands are disabled in order to secure the system. DDF includes a whitelist of allowed shell commands in `<DDF_HOME>/etc/org.apache.karaf.command.acl.shell.cfg`. By default, this list includes commands that are whitelisted only to administrators: • `complete` • `echo` • `format` • `grep` • `if` • `keymap` • `less` • `set` • `setopt` • `sleep` • `tac` • `wc` • `while` • `.invoke` • `unsetopt`

3.9. Configuring Federation

DDF is able to federate to other data sources, including other instances of DDF, with some simple configuration.

3.9.1. 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_HOME>/etc/custom.system.properties` file. The following values can be set:

Property	Sample Value	Description
`javax.net.ssl.trustStore`	`etc/keystores/serverTruststore.jks`	The java keystore that contains the trusted public certificates for Certificate Authorities (CAs) that can be used to validate SSL Connections for outbound TLS/SSL connections (such as HTTPS). When making outbound secure connections a handshake is done with the remote secure server and the CA 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_GCM_SHA256`, `TLS_DHE_RSA_WITH_AES_128_CBC_SHA256`, `TLS_DHE_RSA_WITH_AES_128_CBC_SHA`, `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`, `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`	The cipher suites that are supported when making outbound HTTPS connections
`https.protocols`	`TLSv1.2`,`TLSv1.3`	The protocols that are supported when making outbound HTTPS connections
`https.default.protocol`	`TLSv1.2`	The default protocol used when making outbound HTTPS connections
`jdk.tls.client.protocols`	`TLSv1.2`,`TLSv1.3`	The protocols that are supported when making inbound HTTPS connections

Property	Sample Value	Description
`jdk.tls.ephemeralDHKeySize`	'matched'	For X.509 certificate based authentication (of non-exportable cipher suites), the DH key size matching the corresponding authentication key is used, except that the size must be between 1024 bits and 2048 bits. For example, if the public key size of an authentication certificate is 2048 bits, then the ephemeral DH key size should be 2048 bits unless the cipher suite is exportable. This key sizing scheme keeps the cryptographic strength consistent between authentication keys and key-exchange keys.

NOTE

`<DDF_HOME>` *Directory*

DDF is installed in the `<DDF_HOME>` directory.

3.9.2. Configuring HTTP(S) Ports

To change HTTP or HTTPS ports from the default values, edit the `custom.system.properties` file.

1. Open the file at <DDF_HOME>/etc/custom.system.properties
2. Change the value after the `=` to the desired port numbers:
   a. `org.codice.ddf.system.httpsPort=8993` to `org.codice.ddf.system.httpsPort=<PORT>`
   b. `org.codice.ddf.system.httpPort=8181` to `org.codice.ddf.system.httpPort=<PORT>`
3. Restart DDF for changes to take effect.

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.

3.9.3. Federation Strategy

A federation strategy federates a query to all of the Remote Sources in the query's list, processes the results in a unique way, and then returns the results to the client. For example, implementations can choose to halt processing until all results return and then perform a mass sort or return the results back to the client as soon as they are received back from a Federated Source.

An endpoint can optionally specify the federation strategy to use when it invokes the query operation. Otherwise, the Catalog provides a default federation strategy to use: the Catalog Federation Strategy.

3.9.3.1. Configuring Federation Strategy

The Catalog Federation Strategy configuration can be found in the Admin Console.

1. Navigate to Admin Console.

2. Select **Catalog**
3. Select **Configuration**
4. Select **Catalog Federation Strategy**.

See Federation Strategy configurations for all possible configurations.

3.9.3.1.1. Catalog Federation Strategy

The Catalog Federation Strategy is the default federation strategy and is based on sorting metacards by the sorting parameter specified in the federated query.

The possible sorting values are:

- metacard's effective date/time
- temporal data in the query result
- distance data in the query result
- relevance of the query result

The supported sorting orders are ascending and descending.

The default sorting value/order automatically used is relevance descending.

WARNING	The Catalog Federation Strategy expects the results returned from the Source to be sorted based on whatever sorting criteria were specified. If a metadata record in the query results contains null values for the sorting criteria elements, the Catalog Federation Strategy expects that result to come at the end of the result list.

3.9.4. Connecting to Sources

A **source** is a system consisting of a catalog containing Metacards.

Catalog sources are used to connect Catalog components to data sources, local and remote. Sources act as proxies to the actual external data sources, such as an RDBMS database or a NoSQL database.

Types of Sources

Remote Source

Read-only data sources that support query operations but cannot be used to create, update, or delete metacards.

Federated Sources

A federated source is a remote source that can be included in federated queries by request or as part of an enterprise query. Federated sources support query and site information operations only.

Catalog modification operations, such as create, update, and delete, are not allowed. Federated sources also expose an event service, which allows the Catalog Framework to subscribe to event notifications when metacards are created, updated, and deleted.

Catalog instances can also be federated to each other. Therefore, a Catalog can also act as a federated source to another Catalog.

Connected Sources

A Connected Source is a local or remote source that is always included in every local and enterprise query, but is hidden from being queried individually. A connected source's identifier is removed in all query results by replacing it with DDF's source identifier. The Catalog Framework does not reveal a connected source as a separate source when returning source information responses.

Catalog Providers

A Catalog Provider is used to interact with data providers, such as files systems or databases, to query, create, update, or delete data. The provider also translates between DDF objects and native data formats.

All sources, including federated source and connected source, support queries, but a Catalog provider also allows metacards to be created, updated, and deleted. A Catalog provider typically connects to an external application or a storage system (such as a database), acting as a proxy for all catalog operations.

Catalog Stores

A Catalog Store is an editable store that is either local or remote.

Available Federated Sources

The following Federated Sources are available in a standard installation of DDF:

CSW Specification Profile Federated Source

Queries a CSW version 2.0.2 compliant service.

Federated Source for Atlassian Confluence ®

Retrieve pages, comments, and attachments from an Atlassian Confluence® REST API.

OpenSearch Source

Performs OpenSearch queries for metadata.

WFS 2.0 Source

Allows for requests for geographical features across the web.

CSW Federation Profile Source

Queries a CSW version 2.0.2 compliant service.

WFS 1.1 Source

Allows for requests for geographical features across the web.

GMD CSW Source

Queries a GMD CSW APISO compliant service.

Available Connected Sources

The following Connected Sources are available in a standard installation of DDF:

WFS 2.0 Source

Allows for requests for geographical features across the web.

WFS 1.1 Source

Allows for requests for geographical features across the web.

Available Catalog Stores

The following Catalog Stores are available in a standard installation of DDF:

None.

Available Catalog Providers

The following Catalog Providers are available in a standard installation of DDF:

Solr Catalog Provider

Uses Solr as a catalog.

Available Storage Providers

The following Storage Providers are available in a standard installation of DDF:

Content File System Storage Provider

.Sources Details Availability and configuration details of available sources.

3.9.4.1. CSW Specification Profile Federated Source

The CSW Specification Profile Federated Source should be used when federating to an *external* (non-DDF-based) CSW (version 2.0.2) compliant service.

Installing the CSW Specification Profile Federated Source

Add a New CSW Specification Profile Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Add a New source.

5. Name the New source.
6. Select **CSW Specification Profile Federated Source** from **Source Type**.

Configuring the CSW Specification Profile Federated Source

Configure an Existing CSW Specification Profile Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Select the name of the source to edit.

See CSW Specification Profile Federated Source configurations for all possible configurations.

Usage Limitations of the CSW Specification Profile Federated Source

- Nearest neighbor spatial searches are not supported.

3.9.4.2. Federated Source for Atlassian Confluence ®

The Confluence source provides a Federated Source to retrieve pages, comments, and attachments from an Atlassian Confluence® REST API and turns the results into Metacards the system can use. The Confluence source does provide a Connected Source interface but its functionality has not been verified.

Confluence Source has been tested against the following versions of Confluence with REST API v2

- Confluence 1000.444.5 (Cloud)
- Confluence 5.10.6 (Server)
- Confluence 5.10.7 (Server)

Installing the Confluence Federated Source

The Confluence Federated Source is installed by default with a standard installation in the Catalog application.

Add a New Confluence Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Add a New source.
5. Name the New source.
6. Select **Confluence Federated Source** from **Binding Configurations**.

Configuring the Confluence Federated Source

Configure an Existing Confluence Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Select the name of the source to edit.

See Confluence Federated Source configurations for all possible configurations.

IMPORTANT	If an additional attribute is not part of the Confluence metacard type or injected, the attribute is not added to the metacard.

Usage Limitations of the Confluence Federated Source

Most of the fields that can be queried on Confluence have some sort of restriction on them. Most of the fields do not support the `like` aka `~` operation so the source converts `like` queries to `equal` queries for attributes that don't support `like`. If the source receives a query with attributes it doesn't understand, it just ignores them. If the query doesn't contain any attributes that map to Confluence search attributes, an empty result set is returned.

Depending on your version of Confluence, when downloading attachments you might get redirected to a different download URL. The default `URLResourceReader` configuration allows redirects, but if the option was disabled in the past, the download fails. This can be fixed by re-enabling redirects in the `URLResourceReader` configuration.

3.9.4.3. Content File System Storage Provider

The Content File System Storage Provider is the default Storage Provider included with DDF

Installing the Content File System Storage Provider

The Content File System Storage Provider is installed by default with the Catalog application.

Configuring Content File System Storage Provider

To configure the Content File System Storage Provider:

1. Navigate to the **Admin Console**.
2. Select **Catalog**.
3. Select **Configuration**.
4. Select **Content File System Storage Provider**.

See Content File System Storage Provider configurations for all possible configurations.

3.9.4.4. OpenSearch Source

The OpenSearch source provides a Federated Source that has the capability to do OpenSearch queries for metadata from Content Discovery and Retrieval (CDR) Search V1.1 compliant sources. The OpenSearch source does not provide a Connected Source interface.

Installing an OpenSearch Source

The OpenSearch Source is installed by default with a standard installation in the Catalog application.

Configure a new OpenSearch Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **OpenSearch Source** from **Binding Configurations**.

Configuring an OpenSearch Source

Configure an existing OpenSearch Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

See OpenSearch Source configurations for all possible configurations.

Using OpenSearch Source

Use the OpenSearch source if querying a CDR-compliant search service is desired.

Table 9. Query to OpenSearch Parameter Mapping

Element	OpenSearch HTTP Parameter	DDF Data Location
searchTerms	`q`	Pulled from the query and encoded in UTF-8.
routeTo	`src`	Pulled from the query.
maxResults	`mr`	Pulled from the query.
count	`count`	Pulled from the query.
startIndex	`start`	Pulled from the query.

Element	OpenSearch HTTP Parameter	DDF Data Location
maxTimeout	`mt`	Pulled from the query.
userDN	`dn`	DDF subject
lat	`lat`	Pulled from the query if it is a point-radius query and the radius is > 0. If multiple point radius searches are encountered, each point radius is converted to an approximate polygon as geometry criteria.
lon	`lon`	
radius	`radius`	
box	`bbox`	Pulled from the query if it is a bounding-box query. Or else, calculated from the query if it is a single geometry or polygon query and the `shouldConvertToBBox` configuration option is `true`. NOTE: Converting a polygon that crosses the antimeridian to a bounding box produces an incorrect bounding box. Or else, calculated from the query if it is a geometry collection and the `shouldConvertToBBox` configuration option is `true`. Note: An approximate bounding box is used to represent the geometry collection encompassing all of the geometries within it Area between the geometries are also included in the bounding box. Hence widen the search area.
geometry	`geometry`	Pulled from the DDF query and combined as a geometry collection if multiple spatial query exist.
polygon	`polygon`	According to the OpenSearch Geo Specification this is deprecated. Use the geometry parameter instead.
start	`dtstart`	Pulled from the query if the query has temporal criteria for modified.
end	`dtend`	
filter	`filter`	Pulled from the query.
sort	`sort`	Calculated from the query. Format: `relevance` or `date`. Supports `asc` and `desc` using `:` as delimiter.

Usage Limitations of the OpenSearch Source

The OpenSearch source does not provide a Connected Source interface.

3.9.4.5. WFS 2.0 Source

The WFS 2.0 Source allows for requests for geographical features across the web using platform-independent calls.

Use the WFS Source if querying a WFS version 2.0.0 compliant service. Also see Working with WFS Sources.

Installing the WFS v2.0.0 Source

The WFS v2.0.0 Source is installed by default with a standard installation in the Spatial application.

Configure a new WFS v2.0.0 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a new source.
- Name the new source.
- Select **WFS v2.0.0 Source** from **Binding Configurations**.

Configuring the WFS v2.0.0 Source

Configure an existing WFS v2.0.0 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

See WFS v.2.0 Federated source configurations or WFS v2.0 Connected source configurations for all possible configurations.

WFS URL

The WFS URL must match the endpoint for the service being used. The type of service and version is added automatically, so they do not need to be included. Some servers throw an exception if they are included twice, so do not include those.

The syntax depends on the server. However, in most cases, the syntax is everything before the `?` character in the URL that corresponds to the `GetCapabilities` query.

Example GeoServer 2.5 Syntax

```
http://www.example.org:8080/geoserver/ows?service=wfs&version=2.0.0&request=GetCapabiliti
```

```
es
```

In this case, the WFS URL would be

```
http://www.example.org:8080/geoserver/ows
```

Mapping WFS Feature Properties to Metacard Attributes

The WFS 2.0 Source allows for virtually any schema to be used to describe a feature. A feature is roughly equivalent to a metacard. The `MetacardMapper` was added to allow an administrator to configure which feature properties map to which metacard attributes.

Using the WFS `MetacardMapper`

Use the WFS `MetacardMapper` to configure which feature properties map to which metacard attributes when querying a WFS version 2.0.0 compliant service. When feature collection responses are returned from WFS sources, a default mapping occurs which places the feature properties into metacard attributes, which are then presented to the user via DDF. There can be situations where this automatic mapping is not optimal for your solution. Custom mappings of feature property responses to metacard attributes can be achieved through the `MetacardMapper`. The `MetacardMapper` is set by creating a feature file configuration which specifies the appropriate mapping. The mappings are specific to a given feature type.

Installing the WFS `MetacardMapper`

The WFS `MetacardMapper` is installed by default with a standard installation in the Spatial application.

Configuring the WFS `MetacardMapper`

There are two ways to configure the `MetacardMapper`:

1. The Configuration Admin available in the Admin Console.
2. Placing a `feature.xml` file in the `deploy` directory.

Example WFS MetacardMapper Configuration

The following shows how to configure the `MetacardMapper` to be used with the sample data provided with GeoServer. This configuration shows a custom mapping for the feature type ‘states’. For the given type, the feature property ‘STATE_NAME’ is mapped to the metacard attribute ‘title’. In this particular case, since the state name is mapped to title in the metacard, it is now fully searchable.

Example `MetacardMapper` *Configuration Within a* `feature.xml` *file:*

```
<feature name="geoserver-states" version="2.30.1"
    description="WFS Feature to Metacard mappings for GeoServer Example
{http://www.openplans.org/topp}states">
    <config name="org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetacardMapper-
geoserver.http://www.openplans.org/topp.states">
        featureType = {http://www.openplans.org/topp}states
```

```
        titleMapping = STATE_NAME
    </config>
</feature>
```

3.9.4.6. CSW Federation Profile Source

The CSW Federation Profile Source is DDF's CSW Federation Profile which supports the ability to search collections of descriptive information (metadata) for data, services, and related information objects.

Use the CSW Federation Profile Source when federating to a DDF-based system.

Installing the CSW Federation Profile Source

Configure the CSW Federation Profile Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Add a New source.
4. Name the New source.
5. Select **CSW Specification Profile Federated Source** from **Source Type**.

Configuring the CSW Federation Profile Source

Configure an Existing CSW Federated Source through the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Sources** tab.
4. Select the name of the source to edit.

See CSW Federation Profile Source configurations for all possible configurations.

Usage Limitations of the CSW Federation Profile Source

- Nearest neighbor spatial searches are not supported.

3.9.4.7. Solr Catalog Provider

The Solr Catalog Provider is included with a standard installation of DDF.

SolrCloud

SolrCloud is a cluster of distributed Solr servers used for high availability and scalability. Configuration

shared between Solr Server instances is managed by Zookeeper.

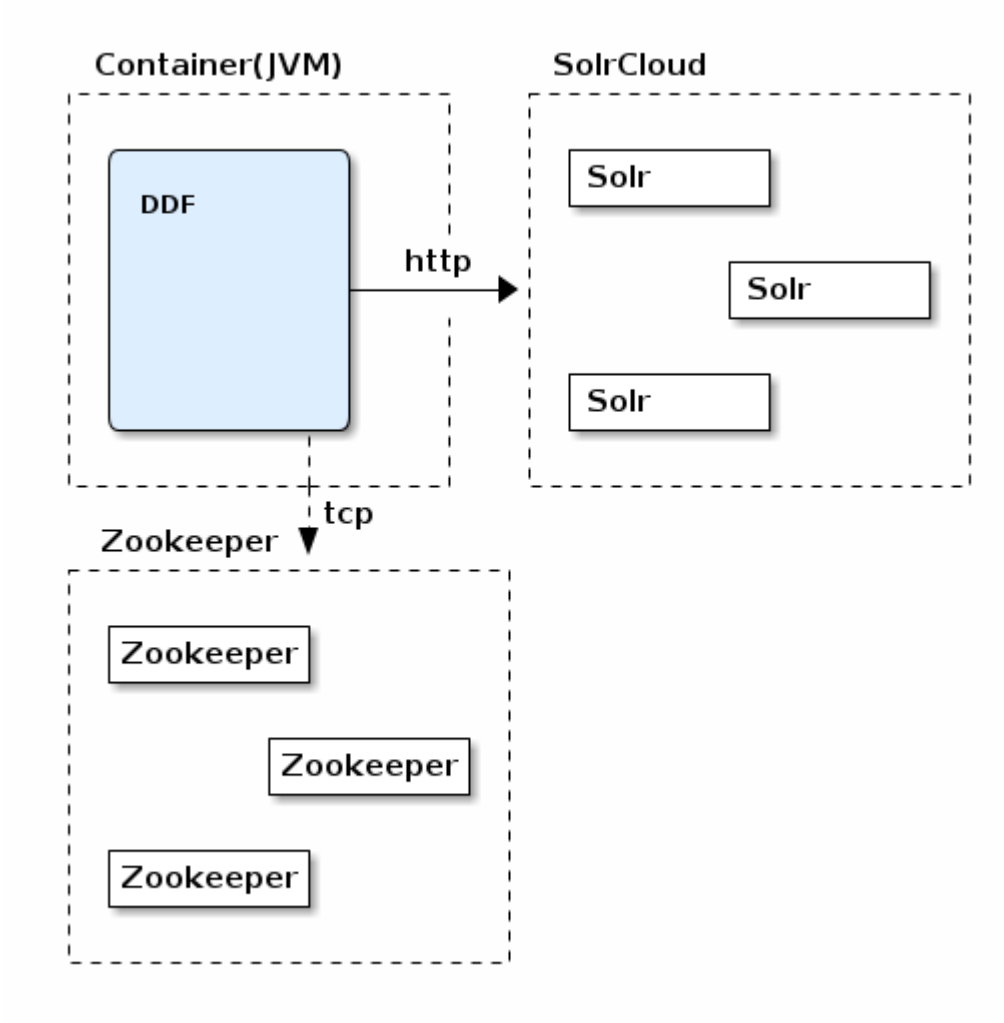

Figure 3. SolrCloud Deployment

SolrCloud Prerequisites

- Solr 9.10.0
- ZooKeeper 3.8.6
- JRE 8 or greater

NOTE	A minimum of three Zookeeper nodes required. Three Zookeeper nodes are needed to form a quorum. A three Zookeeper ensemble allows for a single server to fail and the service to still be available. More Zookeeper nodes can be added to achieve greater fault tolerance. The total number of nodes must always be an odd number. See Setting Up an External Zoo Keeper Ensemble for more information.

Installing SolrCloud

Review and complete the following Zookeeper and Solr documentation:

- Getting Started

- ZooKeeper Getting Started Guide
- Setting Up an External Zookeeper Ensemble
- Taking Solr to Production
- Securing Solr

NOTE	A minimum of two Solr server instances is required. Each Solr server instance must have a minimum of two shards. Having two Solr server instances guarantees that at least one Solr server is available if one fails. The two shards enables the document mapping to be restored if one shard becomes unavailable.

Configuring SolrCloud

The `jute.maxbuffer` property needs to be set on Zookeeper and SolrCloud nodes to support large dictionary files. The following JAR is needed to support WKT geospatial queries and needs to be installed on every Solr server instance.

The JAR can be downloaded from:

a. jts-core-1.17.1.jar

Repeat the following procedure for each Zookeeper and SolrCloud node instance:

1. Add `jute.maxbuffer=0x30D40` to `<ZOOKEEPER_INSTALL_DIR>/conf/zoo.cfg`.
2. Add `SOLR_OPTS="$SOLR_OPTS -Djute.maxbuffer=0x30D40"` to `<SOLR_INSTALL_DIR>/bin/solr.in.cmd`.
3. Copy `jts-core-1.17.1.jar` to: `<SOLR_INSTALL_DIR>/server/solr-webapp/webapp/WEB-INF/lib/`.

Configuring DDF for SolrCloud

1. On the DDF server, edit `<DDF_HOME>/etc/custom.system.properties`:
   a. Comment out the Solr Client Configuration for **Http Solr Client** section.
   b. Uncomment the section for the **Cloud Solr Client**:
   c. Set `solr.cloud.zookeeper` to `<ZOOKEEPER_1_HOSTNAME>:<PORT_NUMBER>`, `<ZOOKEEPER_2_HOSTNAME>:<PORT_NUMBER>`, `<ZOOKEEPER_n_HOSTNAME>:<PORT_NUMBER>`
   d. Set `solr.data.dir` to the desired data directory.

SolrCloud System Properties

```
solr.client = CloudSolrClient
solr.data.dir = ${karaf.home}/data/solr
solr.cloud.zookeeper = zk1:2181,zk2:2181,zk3:2181
```

3.9.4.8. WFS 1.1 Source

The WFS Source allows for requests for geographical features across the web using platform-independent calls.

A Web Feature Service (WFS) source is an implementation of the `FederatedSource` interface provided by the DDF Framework.

Use the WFS Source if querying a WFS version 1.1.0 compliant service.

Installing the WFS v1.1.0 Source

The WFS v1.1.0 Source is installed by default with a standard installation in the Spatial application.

Configure a new WFS v1.1.0 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **WFS v1.1.0 Source** from **Binding Configurations**.

Configuring the WFS v1.1.0 Source

Configure an existing WFS v1.1.0 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

See WFS v.1.1 Federated Source configurations for all possible configurations.

WFS URL

The WFS URL must match the endpoint for the service being used. The type of service and version are added automatically, so they do not need to be included. Some servers throw an exception if they are included twice, so do not include those.

The syntax depends on the server. However, in most cases, the syntax is everything before the `?` character in the URL that corresponds to the `GetCapabilities` query.

Example GeoServer 2.12.1 Syntax

```
http://www.example.org:8080/geoserver/wfs?service=wfs&version=1.1.0&request=GetCapabiliti
```

```
es
```

In this case, the WFS URL would be: `http://www.example.org:8080/geoserver/wfs`

Mapping Metacard Attributes to WFS Feature Properties for Queries

The WFS v1.1.0 Source supports mapping metacard attributes to WFS feature properties for queries (GetFeature requests) to the WFS server. The source uses a `MetacardMapper` service to determine how to map a given metacard attribute in a query to a feature property the WFS server understands. It looks for a `MetacardMapper` whose `getFeatureType()` matches the feature type being queried. Any `MetacardMapper` service implementation can be used, but DDF provides one in the Spatial application called Metacard to WFS Feature Map.

3.9.4.9. GMD CSW Source

The Geographic MetaData extensible markup language (GMD) CSW source supports the ability to search collections of descriptive information (metadata) for data, services, and related information objects, based on the Application Profile ISO 19115/ISO19119.

Use the GMD CSW source if querying a GMD CSW APISO compliant service.

Installing the GMD CSW APISO v2.0.2 Source

The GMD CSW source is installed by default with a standard installation in the Spatial application.

Configure a new GMD CSW APISO v2.0.2 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Add a New source.
- Name the New source.
- Select **GMD CSW ISO Federated Source** from **Binding Configurations**.

Configuring the GMD CSW APISO v2.0.2 Source

Configure an existing GMD CSW APISO v2.0.2 Source through the Admin Console:

- Navigate to the **Admin Console**.
- Select the **Catalog** application.
- Select the **Sources** tab.
- Select the name of the source to edit.

See GMD CSW APISO v2.0.2 Source configurations for all possible configurations.

3.9.5. Configuring Endpoints

Configure endpoints to enable external systems to send and receive content and metadata from DDF.

3.9.5.1. Configuring Catalog REST Endpoint

The Catalog REST endpoint allows clients to perform operations on the Catalog using REST.

To install the Catalog REST endpoint:

1. Navigate to the **Admin Console**.
2. Select **System**.
3. Select **Features**.
4. Install the `catalog-rest-endpoint` feature.

The Catalog REST endpoint has no configurable properties. It can only be installed or uninstalled.

3.9.5.2. Configuring CSW Endpoint

The CSW endpoint enables a client to search collections of descriptive information (metadata) about geospatial data and services.

To install the CSW endpoint:

1. Navigate to the **Admin Console**.
2. Select **System**.
3. Select **Features**.
4. Install the `csw-endpoint` feature.

To control the number of threads used for parallel processing of transactions, set the `org.codice.ddf.spatial.ogc.csw.catalog.endpoint.threadpool` system property in custom.system.properties. The default thread pool is 2 * Number Processors.

3.9.5.3. Configuring KML Endpoint

Keyhole Markup Language (*KML*) is an XML notation for describing geographic annotation and visualization for 2- and 3- dimensional maps.

The root network link creates a network link for each configured source, including the local catalog. The individual source network links perform a query against the OpenSearch Endpoint periodically based on the current view in the KML client. The query parameters for this query are obtained by a bounding box generated by Google Earth. The root network link refreshes every 12 hours or can be forced to refresh. As a user changes their current view, the query is re-executed with the bounding box

of the new view. (This query gets re-executed two seconds after the user stops moving the view.)

This KML Network Link endpoint has the ability to serve up custom KML style documents and Icons to be used within that document. The KML style document must be a valid XML document containing a KML style. The KML Icons should be placed in a single level directory and must be an image type (png, jpg, tif, etc.). The Description is displayed as a pop-up from the root network link on Google Earth. This may contain the general purpose of the network and URLs to external resources.

To install the KML endpoint:

1. Navigate to the **Admin Console**.
2. Select **System**.
3. Select **Features**.
4. Install the `spatial-kml` feature.

To configure the KML endpoint:

1. Navigate to the **Admin Console**.
2. Select **System**.
3. Select **Features**.
4. Select **KML Endpoint**.

See KML Endpoint configurations for all possible configurations.

3.9.5.4. Configuring OpenSearch Endpoint

The OpenSearch endpoint enables a client to send query parameters and receive search results. This endpoint uses the input query parameters to create an OpenSearch query. The client does not need to specify all of the query parameters, only the query parameters of interest.

To install the OpenSearch endpoint:

1. Navigate to the **Admin Console**.
2. Select **System**.
3. Select **Features**.
4. Install the `catalog-opensearch-endpoint` feature.

The OpenSearch endpoint has no configurable properties. It can only be installed or uninstalled.

3.10. Configuring for Special Deployments

In addition to standard configurations, several specialized configurations are possible for specific uses of DDF.

3.10.1. Multiple Installations

One common specialized configuration is installing multiple instances of DDF.

3.10.1.1. Isolating SolrCloud and Zookeeper

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

Zookeeper clients cannot use secure (SSL/TLS) connections. The configuration information that Zookeeper sends and receives is vulnerable to network sniffing. Any unencrypted network traffic is vulnerable to sniffing attacks. To use SolrCloud with Zookeeper securely, these processes must be isolated on the network, or their communications must be encrypted by other means. The DDF process must be visible on the network to allow authorized parties to interact with it.

Examples of Isolation:

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

3.10.2. Configuring for a Fanout Proxy

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

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select **Catalog Standard Framework**.
5. Select **Enable Fanout Proxy**.
6. Save changes.

DDF is now operating as a fanout proxy. Only queries and resource retrieval requests are allowed. All queries are federated. Create, update, and delete requests are not allowed, even if a Catalog Provider was configured prior to the reconfiguration as a fanout.

3.10.3. Configuring for a Highly Available Cluster

This section describes how to make configuration changes after the initial setup for a DDF in a Highly Available Cluster.

In a, configuration changes must be made on both DDF nodes. The changes can still be made in the standard ways via the Admin Console, the Command Line, or the file system.

NOTE

Changes made in the Admin Console must be made through the HTTP proxy. This means that the below steps should be followed to make a change in the Admin Console:

- Make a configuration change on the currently active DDF node
- Shut down the active DDF node, making the failover proxy switch to the standby DDF node
- Make the same configuration change on the newly active DDF node
- Start the DDF node that was just shut down

3.11. Configuring UI Themes

The optional configurations in this section cover minor changes that can be made to optimize DDF appearance.

3.11.1. Landing Page

The Landing Page is the first page presented to users of DDF. It is customizable to allow adding organizationally relevant content.

3.11.1.1. Installing the Landing Page

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

3.11.1.2. Configuring the Landing Page

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

3.11.1.3. Customizing the Landing Page

Configure the Landing Page from the Admin Console:

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

Configure important landing page items such as branding logo, contact information, description, and additional links.

See Landing Page configurations for all possible configurations.

3.11.2. Configuring Logout Page

The logout pages is presented to users through the navigation of DDF and has a changeable timeout value.

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

The customizable feature of the logout page is the **Logout Page Time Out**. This is the time limit the IDP client waits for a user to click log out on the logout page. Any requests that take longer than this time for the user to submit are rejected.

1. **Default value**: 3600000 (milliseconds)

See Logout Configuration for detailed information.

3.11.3. Platform UI Themes

The Platform UI Configuration allows for the customization of attributes of all pages within DDF. It contains settings to display messages to users at login or in banners in the headers and footers of all pages, along with changing the colors of text and backgrounds.

3.11.3.1. Navigating to UI Theme Configuration

1. Navigate to the **Admin Console**.
2. Select the **Platform** application.
3. Select **Configuration**.
4. Select **Platform UI Configuration**.

3.11.3.2. Customizing the UI Theme

The customization of the UI theme across DDF is available through the capabilities of Platform UI Configuration. The banner has four items to configure:

1. **Header** (text)
2. **Footer** (text)
3. **Text Color**
4. **Background Color**

See the Platform UI for all possible configurations of the Platform UI Configuration.

3.12. Miscellaneous Configurations

The optional configurations in this section cover minor changes that can be made to optimize DDF.

3.12.1. Configuring Thread Pools

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

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

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

3.12.2. Configuring Jetty ThreadPool Settings

To prevent resource shortages in the event of concurrent requests, DDF allows configuring Jetty ThreadPool settings to specify the minimum and maximum available threads.

1. The settings can be changed at `etc/org.ops4j.pax.web.cfg` under Jetty Server ThreadPool Settings.
2. Specify the maximum thread amount with `org.ops4j.pax.web.server.maxThreads`
3. Specify the minimum thread amount with `org.ops4j.pax.web.server.minThreads`
4. Specify the allotted time for a thread to complete with `org.ops4j.pax.web.server.idleTimeout`

DDF does not support changing ThreadPool settings from the Command Console or the Admin Console.

3.12.3. Configuring Alerts

By default, DDF uses Karaf Decanter for alerts that can be configured by configuration file. Further information on Karaf Decanter services and configurations can be found here.

3.12.3.1. Configuring Decanter Alerting

Decanter Alerting provides a way to create alerts based on configurable conditions in events posted to `decanter/collect/*` and can be configured by editing the file `<DDF_HOME>/etc/org.apache.karaf.decanter.alerting.service.cfg`. By default, there is only one check that produce alerts, and it is based on the `SystemNotice` event property of `priority` of important or critical.

3.12.4. Encrypting Passwords

DDF includes an encryption service to encrypt plain text such as passwords.

3.12.4.1. Encryption Command

An encrypt security command is provided with DDF to encrypt text. This is useful when displaying password fields to users.

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

1. Navigate to the Command Console.

   security:encrypt Command Example

   ```
   ddf@local>security:encrypt myPasswordToEncrypt
   ```

2. The output is the encrypted value.

   security:encrypt Command Output

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

4. Running

Find directions here for running an installation of DDF.

Starting
: Getting an instance of DDF up and running.

Managing Services
: Running DDF as a managed service.

Maintaining
: Keeping DDF running with useful tasks.

Monitoring
: Tracking system health and usage.

Troubleshooting
: Common tips for unexpected behavior.

4.1. Starting

4.1.1. Managed Services

The lifecycle of DDF and Solr processes can be managed by the operating system. The DDF documentation provides instructions to install DDF as a managed services on supported unix platforms. However, the documentation cannot account for all possible configurations. Please consult the documentation for the operating system and its init manager if the instructions in this document are inadequate.

- Configure Solr to run as a managed service
- Configure DDF to run as a managed service

4.1.2. Run DDF as a Managed Service

4.1.2.1. Running as a Service with Automatic Start on System Boot

Because DDF is built on top of Apache Karaf, DDF can use the Karaf Wrapper to run DDF as a service and enable automatic startup and shutdown. When DDF is started using Karaf Wrapper, new `wrapper.log` and `wrapper.log.n` (where n goes from 1 to 5 by default) log files are generated to include wrapper and console specific information.

WARNING	When installing as a service on *NIX, do not use spaces in the path for <DDF_HOME> as the service scripts that are generated by the wrapper cannot handle spaces.
WARNING	Ensure that JAVA_HOME is properly set before beginning this process. See Java Requirements.
WARNING	Install a profile on the first start of the system. Attempting to install a profile after the first start (such as when starting as a service) may cause issues.
WARNING	If using the default Security Manager, the service should be ran as the user who installed the service in order to avoid permission issues.

1. Create the service wrapper.

   DDF can create native scripts and executable files to run itself as an operating system service. This is an optional feature that is not installed by default. To install the service wrapper feature, go the DDF console and enter the command:

```
ddf@local> feature:install -r wrapper
```

2. Generate the script, configuration, and executable files:

   **NIX*

   ```
   ddf@local> wrapper:install -i setenv-wrapper.conf -n ddf -d ddf -D "DDF Service"
   ```

   Windows

   ```
   ddf@local> wrapper:install -i setenv-windows-wrapper.conf -n ddf -d ddf -D "DDF
   Service"
   ```

3. (Windows users skip this step) (All *NIX) If DDF was installed to run as a non-root user (as-recommended,) edit `<DDF_HOME>/bin/ddf-service` and change the property `#RUN_AS_USER=` to:

   <DDF_HOME>/bin/ddf-service

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

   where <ddf-user> is the intended username:

4. (Windows users skip this step) (All *NIX) Edit `<DDF_HOME>/bin/ddf-service`. Add `LimitNOFILE` to the [Service] section.

   <DDF_HOME>/bin/ddf.service

   ```
   LimitNOFILE=6815744
   ```

5. (Windows users skip this step) (*NIX *with* `systemd`) Install the wrapper startup/shutdown scripts.

   Install the service and start it when the system boots, use `systemctl` From an OS console, execute:

   ```
   root@localhost# systemctl enable <DDF_HOME>/bin/ddf.service
   ```

6. (Windows users skip this step) (*NIX *without* `systemd`) Install the wrapper startup/shutdown scripts.

   If the system does not use `systemd`, use the `init.d` system to install and configure the service. Execute these commands as root or superuser:

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

7. (Windows only) Install the wrapper startup/shutdown scripts.

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

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

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

4.1.2.2. Karaf Documentation

Because DDF is built on top of Apache Karaf, more information on operating DDF can be found in the Karaf documentation.

4.1.3. Run Solr as Managed Service

Refer to Taking Solr to Production in the Solr documentation for configuring Solr as a Windows or init.d service.

Follow the below steps to start and stop DDF.

4.1.4. Starting from Startup Scripts

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

*Start Script (*NIX)*

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

Start Script (Windows)

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

4.1.5. Starting as a Background Process

Alternatively, to run DDF as a background process, run the `start` script:

**NIX*

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

Windows

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

NOTE

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

**NIX*

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

Windows

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

Use the `-h` option followed by the name (`<FQDN>`) or IP of the host where DDF is running.

4.1.6. Stopping DDF

There are two options to stop a running instance:

- Call shutdown from the console:

Shut down with a prompt

```
ddf@local>shutdown
```

Force Shutdown without prompt

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

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

**NIX*

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

Windows

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

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.

4.2. Maintaining

4.2.1. Console Commands

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

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

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

4.2.1.1. 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.

4.2.1.2. CQL Syntax

The CQL syntax used with console commands should follow the OGC CQL format. GeoServer provides a description of the grammar and examples in this CQL Tutorial.

CQL Syntax Examples

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

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

4.2.1.3. Available Console Commands

Many console commands are available, including DDF commands and the core Karaf console commands. For more information about these core Karaf commands and using the console, see the Commands documentation for Karaf 4.4.10 at Karaf documentation.

For a complete list of all available commands, from the Command Console, press **TAB** and confirm when prompted.

Console commands follow a format of `namespace:command`.

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

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

Table 10. DDF Console Command Namespaces

Namespace	Description
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.
platform	The DDF Platform Shell Commands provide generic platform management functions
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.
subscription	The DDF PubSub shell commands provide functions to list the registered subscriptions in DDF and to delete subscriptions.

Namespace	Description
solr	The Solr commands are used for the Solr `CatalogProvider` implementation. They provide commands specific to that provider.

4.2.1.3.1. Catalog Commands

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

Table 11. Catalog Command Descriptions

Command	Description
`catalog:describe`	Provides a basic description of the Catalog implementation.
`catalog:dump`	Exports metacards from the local Catalog. Does not remove them. See date filtering options below.
`catalog:envlist`	**IMPORTANT** Deprecated as of ddf-catalog 2.5.0. Please use `platform:envlist`. Provides a list of environment variables.
`catalog:export`	Exports metacards, history, and their associated resources from the current Catalog.
`catalog:import`	Imports Metacards and history into the current Catalog.
`catalog:ingest`	Ingests data files into the Catalog. XML is the default transformer used. See Ingest Command for detailed instructions on ingesting data and Input Transformers for all available transformers.
`catalog:inspect`	Provides the various fields of a metacard for inspection.
`catalog:latest`	Retrieves the latest records from the Catalog based on the `Core.METACARD_MODIFIED` date.
`catalog:range`	Searches by the given range arguments (exclusively).
`catalog:remove`	Deletes a record from the local Catalog.
`catalog:removeall`	Attempts to delete all records from the local Catalog.

Command	Description
`catalog:replicate`	Replicates data from a federated source into the local Catalog.
`catalog:search`	Searches records in the local Catalog.
`catalog:spatial`	Searches spatially the local Catalog.
`catalog:transformers`	Provides information on available transformers.
`catalog:validate`	Validates an XML file against all installed validators and prints out human readable errors and warnings.

catalog:dump Options

The `catalog:dump` command provides 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 (that is, if the date and time match exactly, the metacard is not included). The date filtering options (`--created-after`, `--created-before`, `--modified-after`, and `--modified-before`) can be used in any combination, with the export result including only metacards that match all of the provided conditions.

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

Date Syntax

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

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

catalog:dump Examples

```
ddf@local>// Given we've ingested a few metacards
ddf@local>catalog:latest
#       ID                                Modified Date              Title
1       a6e9ae09c792438e92a3c9d7452a449f  2026-04-24
2       b4aced45103a400da42f3b319e58c3ed  2026-04-24
3       a63ab22361e14cee9970f5284e8eb4e0  2026-04-24 myTitle

ddf@local>// Filter out older files
ddf@local>catalog:dump --created-after 2026-04-24 /home/user/ddf-catalog-dump
 1 file(s) dumped in 0.015 seconds

ddf@local>// Filter out new file
ddf@local>catalog:dump --created-before 2026-04-24 /home/user/ddf-catalog-dump
 2 file(s) dumped in 0.023 seconds

ddf@local>// Choose middle file
ddf@local>catalog:dump --created-after 2026-04-24 /home/user/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 2026-04-24 /home/user/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 2026-04-24 /home/user/ddf-catalog-dump
 1 file(s) dumped in 0.024 seconds

ddf@local>// Can use UTC instead of (or in combination with) explicit time zone offset
ddf@local>catalog:dump --modified-after 2026-04-24 /home/user/ddf-catalog-dump
 2 file(s) dumped in 0.020 seconds
ddf@local>catalog:dump --modified-after 2026-04-24 /home/user/ddf-catalog-dump
 1 file(s) dumped in 0.015 seconds

ddf@local>// Can leave off time zone, but default (local time on server) may not match
what you expect!
ddf@local>catalog:dump --modified-after 2026-04-24 /home/user/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 2026-04-24 /home/user/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 2026-04-24 /home/user/ddf-catalog-dump
 2 file(s) dumped in 0.027 seconds
```

4.2.1.3.2. Solr Commands

Table 12. Solr Command Descriptions

Command	Description
`solr:backup`	Creates a backup of the selected Solr core/collection. This uses the Solr interface for creating the backup. In SolrCloud deployments the selected backup directory must exist and be shared on all Solr nodes.
`solr:restore`	Restores a Solr backup to the selected core/collection. This uses the Solr interfaces for restoring the backup. In SolrCloud deployments the directory containing the files to restore must exist and be shared on all Solr nodes.

4.2.1.3.3. Subscriptions Commands

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

Table 13. Subscription Command Descriptions

Command	Description
`subscriptions:delete`	Deletes the subscriptions specified by the search phrase or LDAP filter.
`subscriptions:list`	List the subscriptions specified by the search phrase or LDAP filter.

subscriptions:list Command Usage Examples

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

List All Subscriptions

```
ddf@local>subscriptions:list

Total subscriptions found: 3

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

List a Specific Subscription by ID

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

Total subscriptions found: 1

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

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

List Subscriptions Using Wildcards

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

Total subscriptions found: 3

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


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

Total subscriptions found: 1

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


ddf@local>subscriptions:list "*WSDL"

Total subscriptions found: 2

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

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

List Subscriptions Using an LDAP Filter

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

Total subscriptions found: 2

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

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

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

Total subscriptions found: 1

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

subscriptions:delete Command Usage

The arguments for the `subscriptions:delete` command are the same as for the `list` command, except that a search phrase or LDAP filter must be specified. If one of these is not specified, an error is displayed. When the `delete` command is executed, it displays each subscription ID it is deleting. If a subscription matches the search phrase but cannot be deleted, a message in red is 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.
```

4.2.1.3.4. Platform Commands

Table 14. Platform Command Descriptions

Command	Description
`platform:describe`	Shows the current platform configuration.
`platform:envlist`	Provides a list of environment variables.

4.2.1.3.5. Persistence Store Commands

Table 15. Persistence Store Command Descriptions

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

4.2.1.4. Command Scheduler

The Command Scheduler allows administrators to schedule Command Line Commands to be run at specified intervals.

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

NOTE	Further attempts are made to execute the command according to the configured interval even if an attempt fails. See the log for details about failures.

4.2.1.4.1. Schedule a Command

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

1. Navigate to the **Admin Console** (https://{FQDN}:{PORT}/admin).
2. Select the **Platform** application.
3. Click on the **Configuration** tab.

4. Select **Platform Command Scheduler**.
5. Enter the command or commands to be executed in the **Command** text field. Commands can be separated by a semicolon and execute in order from left to right.
6. Enter an interval in the **Interval** field. This can either be a Quartz Cron expression or a positive integer (seconds) (e.x. `0 0 0 1/1 * ? *` or `12`).
7. Select the interval type in the **Interval Type** drop-down.
8. Click the **Save changes** button.

NOTE	Scheduling commands are delayed by 1 minute to allow time for bundles to load when DDF is starting up.

4.2.1.4.2. Updating a Scheduled Command

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

1. Navigate to the **Admin Console**.
2. Click on the **Platform** application.
3. Click on the **Configuration** tab.
4. Under the **Platform Command Scheduler** configuration are all of 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 fields.
6. Click the **Save changes** button.

4.2.1.4.3. Output of Scheduled Commands

Commands that normally write out to the console write out to the log. For example, if an `echo "Hello World"` command is set to run every five seconds, the log contains 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.

4.3. Monitoring

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

4.3.1. Metrics Reporting

Metrics are exposed over a Prometheus endpoint at `/metrics`. In order to extract and store the metrics, a Prometheus server is required.

A user interface like Grafana can be used to display metrics.

4.3.2. Managing Logging

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

4.3.2.1. Configuring Logging

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

4.3.2.2. DDF log file

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

```
log4j.appender.out.file=<DDF_HOME>/data/log/ddf.log
```

4.3.2.3. Controlling log level

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

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

4.3.2.4. 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
```

4.3.2.5. Number of backup log files to keep

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

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

4.3.2.6. 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`

4.3.2.7. Logging External Resources

Other appenders can be selected and configured.

For more detail on configuring the log file and what is logged to the console see Log in the Karaf documentation.

4.3.2.8. Enabling HTTP Access Logging

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

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

```
<Get name="handler">
    <Call name="addHandler">
      <Arg>
        <New class="org.eclipse.jetty.server.handler.RequestLogHandler">
          <Set name="requestLog">
            <New id="RequestLogImpl" class="org.eclipse.jetty.server.NCSARequestLog">
              <Arg><SystemProperty name="jetty.logs" default="data/log/"
/>/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 defaults 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
```

4.3.2.9. Using the LogViewer

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

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

Log events can be filtered by:

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

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

4.4. Troubleshooting

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

Table 16. General Troubleshooting

Issue	Solution
Unable to unzip distribution on Windows platform	The default Windows zip utility is not compatible with the DDF distribution zip file. Use Java or a third-party zip utility.
Unable to federate on Windows Platform	Windows default firewall is not compatible with DDF.
Ingesting more than 200,000 data files stored NFS shares may cause Java Heap Space error (Linux-only issue).	This is an NFS bug where it creates duplicate entries for some files when doing a file list. Depending on the OS, some Linux machines can handle the bug better and able get a list of files but get an incorrect number of files. Others would have a Java Heap Space error because there are too many file to list. As a workaround, ingest files in batches smaller than 200,000.
Ingesting serialized data file with scientific notation in WKT string causes RuntimeException.	WKT string with scientific notation such as POINT (-34.8932113039107 -4.77974239601E-5) won't ingest. This occurs with serialized data format only.
Exception Starting DDF (Windows) An exception is sometimes thrown starting DDF on a Windows machine (x86). If using an unsupported terminal, `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. 1. Shut down DDF: `ddf@local>shutdown` 2. Start up DDF: `./ddf`

Issue	Solution

Distribution Does Not Start
 	
DDF does not start when calling the start script defined during installation.
	Complete the following procedure.
	
	1. Verify that Java is correctly installed.
	
	`java -version`
	
	2. 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)`
	
	3. If running *nix, verify that bash is installed.
	
	`echo $SHELL`
	
	4. 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 (and use kill/close commands, if necessary). • Utilize automated start/stop scripts to run DDF as a service.
Troubleshooting TLS/SSL Connectivity	
Connection error messages found in log files, for example: • `java.net.SocketException : Broken pipe` • `java.net.SocketException : Connection reset`	Ensure all the Certificate Authorities (CAs) and the whole trust chain (all intermediate certificates are included in the walk-up to the CA) have been added to the trust store.
	Ensure none of the certificates or CAs have expired.
	Ensure the alias of the private key in the keystore matches the hostname.
	Ensure the time on all hosts and VMs are the same (no time drifts).
	Ensure the server's private key password matches the password of the keystore (if the key has a password).
	Ensure the Subject Alternative Names (SANS) includes the fully qualified domain name (FQDN) and IP address of the system (not required but helpful).

4.4.1. Deleted Records Are Being Displayed In The Search UI's Search Results

When queries are issued by the 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 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://{FQDN}:{PORT}/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://{FQDN}:{PORT}/solr/metacard_cache/update?stream.body=<delete><query>original_id_t
xt:50ffd32b21254c8a90c15fccfb98f139</query></delete>&commit=true
```

- To delete records in the Solr cache using a query on a field in the records - 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://{FQDN}:{PORT}/solr/metacard_cache/update?stream.body=<delete><query>title_txt:*re
mote*</query></delete>&commit=true
```

5. Data Management

5.1. Ingesting Data

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

NOTE *Guest Claims Attributes and Ingest*

> Ensure that appropriate Guest Claims are configured to allow guest users to ingest data and query the catalog.

5.1.1. Ingest Command

The Command Console has a command-line option for ingesting data.

NOTE

> Ingesting with the console ingest command creates a metacard in the catalog, but does not copy the resource to the content store. The Ingest Command requires read access to the directory being ingested. See the URL Resource Reader for configuring read permission entries to the directory.

The syntax for the ingest command is

```
ingest -t <transformer type> <file path>
```

Select the `<transformer type>` based on the type of files ingested. Metadata is extracted if it exists in a format compatible with the transformer. The default transformer is the XML input transformer, which supports the metadata schema `catalog:metacard`. To see a list of all transformers currently installed, and the file types supported by each, run the `catalog:transformers` command.

For more information on the schemas and file types(mime-types) supported by each transformer see the Input Transformers.

The `<file path>` is relative to the <DDF_HOME> directory. This can be the path to a file or a directory containing the desired files.

NOTE

> *Windows Users*
>
> On Windows, put the file path in quotes: `"path/to/file"`.

Successful command line ingest operations are accompanied with messaging indicating how many files were ingested and how long the operations took. The ingest command also prints which files could not be ingested with additional details recorded in the ingest log. The default location of the log is `<DDF_HOME>/data/log/ingest_error.log`.

5.1.2. Content Directory Monitor Ingest

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

Files placed in the monitored directory are ingested automatically. If a file cannot be ingested, they are moved to an automatically created directory named `.errors`. More information about the ingest operations can be found in the ingest log. The default location of the log is

`<DDF_HOME>/data/log/ingest_error.log`. Optionally, ingested files can be automatically moved to a directory called `.ingested`.

5.1.3. External Methods of Ingesting Data

Third-party tools, such as cURL.exe and the Chrome Advanced Rest Client, can be used to send files 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://{FQDN}:{PORT}/services/catalog
```

**NIX Example*

```
curl -H "Content-type: application/json;id=geojson" -i -X POST -d @geojson_valid.json
https://{FQDN}:{PORT}/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 receiving the data.

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

Sample Response

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

1. Use a web browser to verify a file was successfully ingested. Enter the URL returned in the response's HTTP header in a web browser. For instance in the example, it was `/services/catalog/44dc84da101c4f9d9f751e38d9c4d97b`. The browser displays the catalog entry as XML in the browser.

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

A resource can also be ingested with metacard metadata associated with it using the multipart/mixed content type.

Example

```
curl -k -X POST -i -H "Content-Type: multipart/mixed" -F
parse.resource=@/path/to/resource -F parse.metadata=@/path/to/metacard
https://{FQDN}:{PORT}/services/catalog
```

More information about the ingest operations can be found in the ingest log. The default location of the log is `<DDF_HOME>/data/log/ingest_error.log`.

5.1.4. Creating And Managing System Search Forms Through Karaf

System search provide a way to execute queries with pre-defined templates and search criteria. System search forms are loaded via the system and are read-only. This command allows an administrator to ingest, modify, or remove system search forms within the system.

Loading Forms With Defaults

```
forms:load
```

Loading Forms With Overrides

```
forms:load --formsDirectory "/etc/forms" --forms "forms.json" --results "results.json"
```

Where:
-formsDirectory Specifies the directory in which the forms JSON and XML reside.

-results Specifies the file name of the `results.json` file.

-forms Specifies the file name of the `forms.json` file.

It's important to note that `forms:load` falls back to the system default location for forms, results and the forms directory. The defaults are as follows:

```
formsDirectory: "/etc/forms"
forms: "forms.json"
results: "results.json"
```

Example search forms and result form data can be found in `<DDF_HOME>/etc/forms/readme.md`.

Managing Forms

In addition to ingesting new system forms into the system, you have the capability to manage the forms, view the forms and remove them.

Viewing All Forms

```
forms:manage --list
```

Removing Single Form

```
forms:manage --remove-single "METACARD_ID"
```

Removing All Forms

```
forms:manage --remove-all
```

Where:
-list Displays the titles and IDs of all system forms in the system

-remove-single Takes in a metacard ID as an argument and removes it

-remove-all Removes all system forms from the system

5.1.5. Other Methods of Ingesting Data

The DDF provides endpoints for integration with other data systems and to further automate ingesting data into the catalog. See Endpoints for more information.

5.2. Backing Up the Catalog

To backup local catalog records, a Catalog Backup Plugin is available. It is not installed by default for performance reasons.

See Catalog Backup Plugin for installation and configuration instructions).

5.3. Validating Data

Configure DDF to perform validation on ingested documents to verify the integrity of the metadata brought into the catalog.

Isolate metacards with data validation issues and edit the metacard to correct validation errors. Additional attributes can be added to metacards as needed.

5.3.1. Validator Plugins on Ingest

When Enforce Errors is enabled within the Admin Console, validator plugins ensure the data being ingested is valid. Below is a list of the validators run against the data ingested.

NOTE

Enforcing errors*:*

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Configuration** tab.
4. Select **Metacard Validation Marker Plugin**.
   a. If **Enforce errors** is checked, these validators below are run on ingest.
   b. If **Enforce errors** is not checked, these validators below are **not** run on ingest.

5.3.1.1. Validators run on ingest

- **Size Validator**: Validates the size of an attribute's values.
- **Range Validator**: Validates an attribute's values against an **inclusive** numeric range.
- **Enumeration Validator**: Validates an attribute's values against a set of acceptable values.
- **Future Date Validator**: Validates an attribute's values against the current date and time, validating that they are in the future.
- **Past Date Validator**: Validates an attribute's values against the current date and time, validating that they are in the past.
- **ISO3 Country Code Validator**: Validates an attribute's values against the ISO_3166-1 Alpha3 country codes.
- **Pattern Validator**: Validates an attribute's values against a regular expression.
- **Required Attributes Metacard Validator**: Validates that a metacard contains certain attributes.
  - ID: `ddf.catalog.validation.impl.validator.RequiredAttributesMetacardValidator`
- **Duplication Validator**: Validates metacard against the local catalog for duplicates based on configurable attributes.
  - ID: `org.codice.ddf.validator.metacard.duplication.DuplicationValidator`
- **Relationship Validator**: Validates values that an attribute **must have**, **can only have**, and/or **can't have**.
  - ID: `ddf.catalog.validation.impl.validator.RelationshipValidator`
- **Metacard WKT Validator**: Validates a location metacard attribute (WKT string) against valid geometric shapes.

- ID: `org.codice.ddf.validator.metacard.wkt.MetacardWktValidator`

5.3.2. Configuring Schematron Services

DDF uses Schematron Validation to validate metadata ingested into the catalog.

Custom schematron rulesets can be used to validate metacard metadata. Multiple services can be created, and each service can have multiple rulesets associated with it. Namespaces are used to distinguish services. The root schematron files may be placed anywhere on the file system as long as they are configured with an absolute path. Any root schematron files with a relative path are assumed to be relative to `<DDF_HOME>/schematron`.

TIP	Schematron files may reference other schematron files using an include statement with a relative path. However, when using the document function within a schematron ruleset to reference another file, the path must be absolute or relative to the DDF installation home directory.

Schematron validation services are configured with a namespace and one or more schematron rulesets. Additionally, warnings may be suppressed so that only errors are reported.

To create a new service:

- Navigate to the **Admin Console**.
- Select the **Catalog**.
- Select **Configuration**.
- Ensure that `catalog-schematron-plugin` is started.
- Select **Schematron Validation Services**.

5.3.3. Injecting Attributes

To create a new attribute, it must be injected into the metacard before it is available to edit or override.

Injections are defined in a JSON-formatted file See Developing Attribute Injections for details on creating an attribute injection file.

5.3.4. Overriding Attributes

Automatically change the value of an existing attribute on ingest by setting an attribute override.

NOTE	Attribute overrides are available for the following ingest methods: • Content Directory Monitor.

- Confluence source.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select **Configuration**.
4. Select the configuration for the desired ingest method.
   a. **Catalog Content Directory Monitor**.
   b. **Confluence Connected Source**.
   c. **Confluence Federated Source**.
5. Select **Attribute Overrides**.
6. Enter the key-value pair for the attribute to override and the values to set.

5.4. Removing Expired Records from the 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.

5.5. Automatically Added Metacard Attributes

This section describes how attributes are automatically added to metacards.

5.5.1. Attributes Added on Ingest

A metacard is first created and populated by parsing the ingested resource with an Input Transformer. Then Attributes Are Injected, Default Attribute Types are applied, and Attribute are Overridden. Finally the metacard is passed through a series of Pre-Authorization Plugins and Pre-Ingest Plugins.

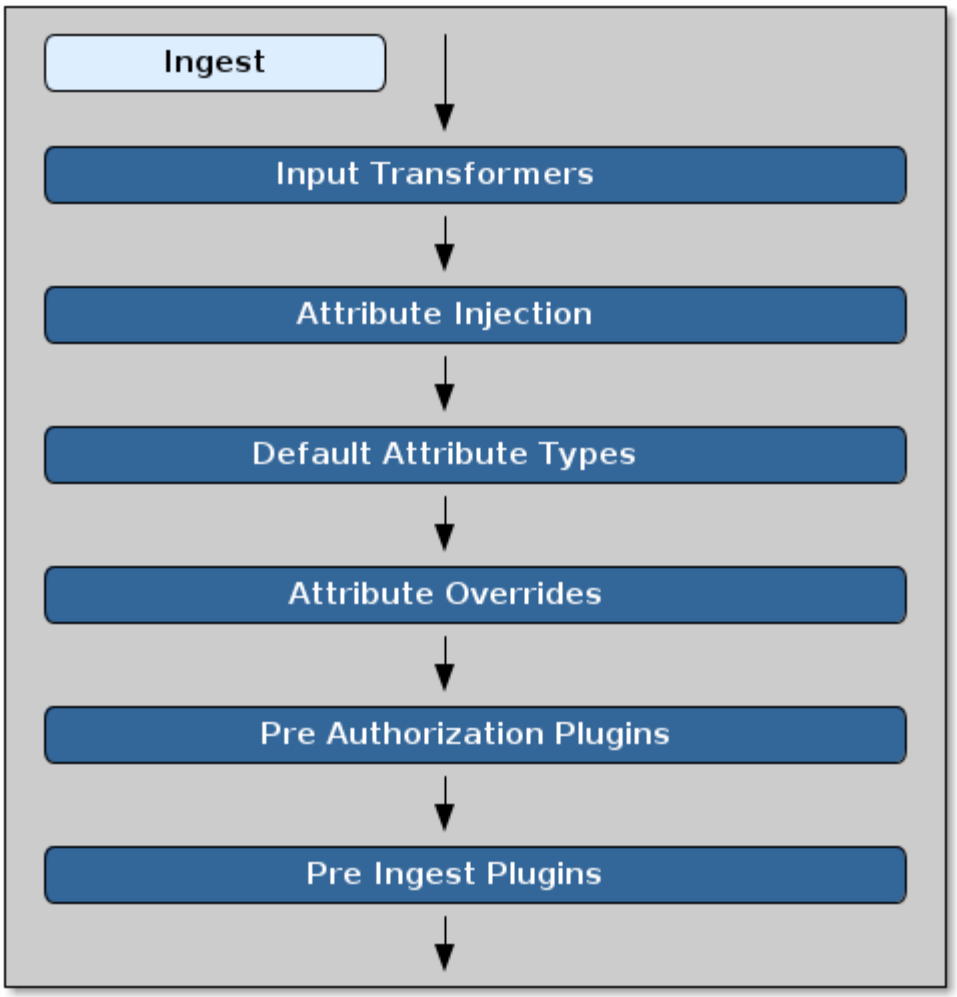

Figure 4. Ingest Attribute Flow

5.5.1.1. Attributes Added by Input Transformers

Input Transformers create and populate metacards by parsing a resource. See File Format Specific Attributes to see the attributes used for specific file formats.

DDF chooses which input transformer to use by:

1. Resolving the mimetype for the resource.
2. Gathering all of the input transformers associated with the resolved mimetype. See Supported File Formats for a list of supported mimetypes.
3. Iterating through the transformers until a successful transformation is performed.

The first transformer that can successfully create a metacard from the ingested resource is chosen. If no transformer can successfully transform the resource, the ingest process fails.

IMPORTANT Each of the ingest methods may have subtle differences when resolving the

resource's mimetype/input transformer.

5.5.1.2. Attributes Added by Attribute Injection

Attribute Injection is the act of adding attributes to a metacard's Metacard Type. A Metacard Type indicates the attributes available for a particular metacard, and is created at the same time as the metacard.

NOTE

Attribute values can only be set/modified if the attribute exists in the metacard's metacard type.

Attributes are initially injected with blank values. However, if an attempt is made to inject an attribute that already exists, the attribute retains the original value.

See Catalog Taxonomy Definitions for a list of attributes injected by default.
See Developing Attribute Injections to learn how to configure attribute injections.

5.5.1.3. Attributes Added by Default Attribute Types

Developing Default Attribute Types is a configurable way to assign default values to a metacard's attributes.

Note that the attribute must be part of the metacard's Metacard Type before it can be assigned a default value.
See Attributes Added By Attribute Injection for more information about injecting attributes into the metacard type.

5.5.1.4. Attributes Added by Attribute Overrides (Ingest)

Attribute Overriding is the act of replacing existing attribute values with a new value.

Attribute overrides can be configured for the Content Directory Monitor.

Note that the attribute must be part of the metacard's Metacard Type before it can be overridden.
See Attributes Added By Attribute Injection for more information about injecting attributes into the metacard type.

5.5.1.5. Attributes Added by Pre-Authorization Plugins

The Pre-Authorization Plugins provide an opportunity to take action before any security rules are applied.

- The Metacard Ingest Network Plugin is a configurable plugin that allows the conditional insertion of new attributes on metacards during ingest based on network information from the ingest request. See Configuring the Metacard Ingest Network Plugin for configuration details.

5.5.1.6. Attributes Added by Pre-Ingest Plugins

The Pre-Ingest Plugins are responsible for setting attribute fields on metacards before they are stored in the catalog.

- The Expiration Date Pre-Ingest Plugin adds or updates expiration dates which can be used later for archiving old data.
- The Geocoder Plugin is responsible for populating the metacard's `Location.COUNTRY_CODE` attribute if the metacard has an associated location. If the metacard's country code is already populated, the plugin does not override it.
- The Metacard Groomer plugin adds/updates IDs and timestamps to the created metacard.

5.5.2. Attributes Added on Query

Metacards resulting from a query undergo Attribute Injection, then have their Attributes Overridden.

5.5.2.1. Attributes Added by Attribute Overrides (Query)

Attribute Overriding is the act of replacing existing attribute values with a new value.

Attribute overrides can be configured for query results from the following Sources:

- Federated Source For Atlassian Confluence.
- CSW Specification Profile Federated Source.
- GMD CSW Federated Source.

Note that the attribute must be part of the metacard's Metacard Type before it can be overridden.
See Attributes Added By Attribute Injection for more information about injecting attributes into the metacard type.

Index

A

Admin Console, 34
Alerts, 103
Audit logging, 9
Authentication Types, 74

C

Catalog Backup Plugin, 129
Catalog Providers, 64, 85
Catalog REST endpoint, 98
Catalog sources, 84
Catalog Stores, 85
Certificate Revocation List, 59
Command Scheduler, 118
Connected Sources, 85
Console command, 35
Console Commands, 109
Content Directory Monitor, 67, 126
CQL syntax, 110
CSW endpoint, 98
`custom.system.properties`, 37

D

Data Access Plugins, 73
Data attributes, 74
Data Management, 63
Disallowing Login Without Certificates, 59
Duplication Validator, 130

E

Encrypting Passwords, 104
Enumeration Validator, 130
Environment Hardening, 77

F

Failover Proxy, 31
Fanout Proxy, 100
Feature Commands, 35
Federated Sources, 84
Federation, 81
Filtering, 76
Firewall Port Configuration, 30
Future Date Validator, 130

G

Global configuration settings, 37
Guest Access, 50
Guest access, 74

H

Hardware Requirements, 11
High Availability, 30
Highly Available Cluster, 101
Hostname, 65

I

Ingest Command, 126
Ingesting, 125
Initial Startup, 24
Insecure Defaults Service, 8
Internet Explorer 11 Enhanced Security Configuration, 29
ISO3 Country Code Validator, 130

J

Java Requirements, 11
Jetty ThreadPool settings, 103

K

Keycloak, 54
Keyhole Markup Language, 98
Keystores, 20
KML, 98
KML Network Link endpoint, 98

L

Landing Page, 101
LDAP, 54
Logout Page, 102
LogViewer, 122

M

Managed Services, 105
Metacard Attribute Security Policy Plugin, 71

Metacard Validation Marker Plugin, 72
Metacard Validity Filter Plugin, 72
Metacard WKT Validator, 130
Metrics Reporting, 120
MFA, 54
Monitoring, 120
Multi-factor authentication, 54

O

OAuth 2.0, 51
Online Certificate Status Protocol, 63
Online Certificate Status Protocol (OCSP) server, 59
OpenID Connect
  OIDC, 51
OpenSearch endpoint, 99

P

Past Date Validator, 130
Pattern Validator, 130
Persistence Store Commands, 118
Platform Commands, 117
Platform UI Themes, 102
Policy Plugins, 77

R

Range Validator, 130
Relationship Validator, 130
Remote Source, 84
Removing Default Users, 58
Required Attributes Metacard Validator, 130
REST services, 51
REST sources, 51
Restricting Access to Admin Console, 56
Restricting Access to Features
  Apps
    Services, 57
Run DDF as a Managed Service, 105

S

SAML Identity Provider, 53
Schematron, 131
Security Audit Plugin, 73
Security Hardening, 8
Security policies, 74
Setting Directory Permissions, 17
Setting Internal Policies, 76
Single Sign On
  SSO, 27
Size Validator, 130
Solr Admin UI, 64
Solr Commands, 114
Sources, 84
SSL, 81
Startup, 107
Subscriptions Commands, 114
System search, 128
System Usage Message, 71
System users, 55

T

Troubleshooting, 122
Truststore, 21
Two-factor authentication, 54

U

User attributes, 74

V

Validator Plugins, 130

W

Web Context Policy Manager, 74

X

XACML policy, 77
XML Attribute Security Policy Plugin, 73

Z

Zookeeper, 100