Distributed Data Framework Documentation: Complete Documentation

Table of Contents

License . . . 24
Introduction . . . 25
1. About DDF . . . 25
1.1. Introducing DDF . . . 25
1.2. Component Applications . . . 25
2. Documentation Guide . . . 26
2.1. Documentation Conventions . . . 26
2.1.1. Customizable Values . . . 26
2.1.2. Code Values . . . 27
2.1.3. Hyperlinks . . . 27
2.2. Support . . . 27
2.2.1. Documentation Updates . . . 27
3. Core Concepts . . . 27
3.1. Introduction to Search . . . 27
3.2. Introduction to Metadata . . . 28
3.3. Introduction to Ingest . . . 28
3.4. Introduction to Resources . . . 28
3.5. Introduction to the Catalog Framework . . . 29
3.6. Introduction to Federation and Sources . . . 29
3.7. Introduction to Events and Subscriptions . . . 30
3.8. Introduction to Endpoints . . . 30
3.9. Standards Supported by DDF . . . 30
3.9.1. Catalog Service Standards . . . 30
3.9.2. Data Formats . . . 31
3.9.3. Security Standards . . . 32
3.10. Introduction to High Availability . . . 33
3.10.1. High Availability Supported Capabilities . . . 34
4. Quick Start Tutorial . . . 35
4.1. Installing (Quick Start) . . . 35

4.1.1. Quick Install Prerequisites . . . 35
4.1.2. Quick Install of SolrCloud . . . 36
4.1.3. Quick Install of DDF . . . 36
4.1.4. Quick Install of DDF on a remote headless server . . . 37
4.2. Certificates (Quick Start) . . . 38
4.2.1. Demo Certificate Authority (CA) . . . 39
4.2.1.1. Creating New Server Keystore Entry with the CertNew Scripts . . . 39
4.2.1.2. Dealing with Lack of DNS . . . 41
4.2.2. Creating Self-Signed Certificates . . . 41
4.2.2.1. Creating a custom CA Key and Certificate . . . 42
4.2.2.2. Sign Certificates Using the custom CA . . . 42
4.2.3. Updating Settings After Changing Certificates . . . 42
4.3. Configuring (Quick Start) . . . 42
4.4. Adding New User Roles . . . 43
Managing . . . 44
5. Securing . . . 44
5.1. Security Hardening . . . 44
5.2. Auditing . . . 45
5.2.1. Enabling Fallback Audit Logging . . . 46
6. Installing . . . 47
6.1. Installation Prerequisites . . . 47
6.1.1. Hardware Requirements . . . 47
6.1.2. Java Requirements . . . 48
6.2. Installing With the DDF Distribution Zip . . . 51
6.2.1. Configuring Operating Permissions and Allocations . . . 53
6.2.1.1. Setting Directory Permissions . . . 53
6.2.1.2. Configuring Memory Allocation for the DDF Java Virtual Machine . . . 55
6.2.1.3. Enabling JMX . . . 55
6.2.2. Managing Keystores and Certificates . . . 56
6.2.2.1. Managing Keystores . . . 57
6.2.2.1.1. Adding an Existing Server Keystore . . . 58
6.2.2.1.2. Adding an Existing Server Truststore . . . 58
6.2.2.1.3. Creating a New Keystore / Truststore with an Existing Certificate and Private Key . . . 58
6.2.2.1.4. Updating Key Store / Trust Store via the Admin Console . . . 59
6.3. Initial Startup . . . 61
6.3.1. Verifying Startup . . . 61
6.3.2. DDF Directory Contents after Installation and Initial Startup . . . 62
6.3.3. Completing Installation . . . 63

6.3.3.1. Completing Installation from the Admin Console . . . 63
6.3.3.2. Completing Installation from the Command Console . . . 66
6.3.4. Internet Explorer 11 Enhanced Security Configuration . . . 66
6.3.5. Firewall Port Configuration . . . 67
6.4. High Availability Initial Setup . . . 67
6.4.1. High Availability Initial Setup Exceptions . . . 68
6.4.1.1. Failover Proxy Integration . . . 68
6.4.1.2. Identical Directory Structures . . . 69
6.4.1.3. Highly Available Security Auditing . . . 69
6.4.1.4. Shared Storage Provider . . . 69
6.4.1.5. High Availability Certificates . . . 69
6.4.1.6. High Availability Installation Profile . . . 70
7. Configuring . . . 70
7.1. Admin Console Tutorial . . . 71
7.2. Console Command Reference . . . 72
7.2.1. Feature Commands . . . 72
7.2.1.1. Uninstalling Features from the Command Console . . . 73
7.3. Configuration Files . . . 74
7.3.1. Configuring Global Settings with custom.system.properties . . . 74
7.3.2. Configuring with .config Files . . . 86
7.4. Configuring User Access . . . 86
7.4.1. Configuring Guest Access . . . 87
7.4.1.1. Denying Guest User Access . . . 87
7.4.1.2. Allowing Guest User Access . . . 87
7.4.1.2.1. Configuring Guest Interceptor if Allowing Guest Users . . . 87
7.4.1.2.2. Configuring Guest Claim Attributes . . . 88
7.4.2. Configuring REST Services for Users . . . 88
7.4.2.1. Configuring OpenID Connect (OIDC) and OAuth 2.0 . . . 88
7.4.2.1.1. OIDC . . . 89
7.4.2.1.2. OAuth 2.0 . . . 89
7.4.2.2. Connecting to an External SAML Identity Provider . . . 90
7.4.2.3. Configuring Without SAML . . . 90
7.4.2.4. Configuring Multi Factor Authentication . . . 91
7.4.3. Connecting to an LDAP Server . . . 91
7.4.4. Updating System Users . . . 92
7.4.5. Restricting Access to Admin Console . . . 93
7.4.5.1. Restricting Feature, App, Service, and Configuration Access . . . 93
7.4.6. Restricting Access to Features, Apps, Services, and Configurations . . . 94

7.4.7. Removing Default Users . . . 95
7.4.8. Disallowing Login Without Certificates . . . 96
7.4.9. Managing Certificate Revocation . . . 96
7.4.9.1. Managing a Certificate Revocation List (CRL) . . . 97
7.4.9.1.1. Creating a CRL . . . 97
Revoke a Certificate and Create a New CRL that Contains the Revoked Certificate . . . 97
Viewing a CRL . . . 97
7.4.9.1.2. Enabling Certificate Revocation . . . 97
Add Revocation to a Web Context . . . 98
Adding Revocation to an Endpoint . . . 98
Verifying Revocation . . . 99
7.4.9.2. Managing an Online Certificate Status Protocol (OCSP) Server . . . 100
7.4.9.2.1. Enabling OCSP Revocation . . . 100
7.5. Configuring Data Management . . . 100
7.5.1. Configuring Solr . . . 100
7.5.1.1. Configuring Solr Catalog Provider Synonyms . . . 101
7.5.1.1.1. Defining synonym rules in the Solr Provider . . . 101
7.5.1.2. Hardening Solr . . . 101
7.5.1.2.1. Configuring Solr Encryption . . . 101
7.5.1.3. Accessing the Solr Admin UI . . . 101
7.5.2. Changing Catalog Providers . . . 101
7.5.3. Changing Hostname . . . 102
7.5.4. Configuring Errors and Warnings . . . 102
7.5.4.1. Enforcing Errors or Warnings . . . 102
7.5.4.2. Hiding Errors or Warnings from Queries . . . 103
7.5.4.3. Hiding Errors and Warnings from Users Based on Role . . . 103
7.5.5. Configuring Product Caching . . . 104
7.5.6. Content Directory Monitor . . . 104
7.5.6.1. Installing the Content Directory Monitor . . . 104
7.5.6.2. Configuring Permissions for the Content Directory Monitor . . . 104
7.5.6.3. Configuring the Content Directory Monitor . . . 105
7.5.6.4. Using the Content Directory Monitor . . . 105
7.5.7. Configuring System Usage Message . . . 108
7.5.8. Configuring Data Policy Plugins . . . 108
7.5.8.1. Configuring the Metacard Attribute Security Policy Plugin . . . 108
7.5.8.2. Configuring the Metacard Validation Marker Plugin . . . 109
7.5.8.3. Configuring the Metacard Validity Filter Plugin . . . 109
7.5.8.4. Configuring the XML Attribute Security Policy Plugin . . . 110

7.5.9. Configuring Data Access Plugins . . . 110
7.5.9.1. Configuring the Security Audit Plugin . . . 110
7.6. Configuring Security Policies . . . 111
7.6.1. Configuring the Web Context Policy Manager . . . 111
7.6.1.1. Guest Access . . . 111
7.6.1.2. Session Storage . . . 111
7.6.1.3. Authentication Types . . . 111
7.6.1.3.1. Terminating and Non-Terminating Authentication Types . . . 112
7.6.1.4. Required Attributes . . . 113
7.6.1.5. White Listed Contexts . . . 113
7.6.2. Configuring Catalog Filtering Policies . . . 113
7.6.2.1. Setting Internal Policies . . . 113
7.6.2.2. Setting XACML Policies . . . 114
7.6.2.3. Catalog Filter Policy Plugins . . . 114
7.7. Configuring User Interfaces . . . 114
7.8. Environment Hardening . . . 114
7.8.1. Known Issues with Environment Hardening . . . 114
7.9. Configuring Federation . . . 118
7.9.1. Enable SSL for Clients . . . 118
7.9.2. Configuring HTTP(S) Ports . . . 120
7.9.3. Federation Strategy . . . 120
7.9.3.1. Configuring Federation Strategy . . . 120
7.9.3.1.1. Catalog Federation Strategy . . . 121
7.9.4. Connecting to Sources . . . 121
7.9.4.1. CSW Specification Profile Federated Source . . . 123
7.9.4.2. Federated Source for Atlassian Confluence ® . . . 124
7.9.4.3. Content File System Storage Provider . . . 125
7.9.4.4. OpenSearch Source . . . 126
7.9.4.5. WFS 2.0 Source . . . 128
7.9.4.6. CSW Federation Profile Source . . . 130
7.9.4.7. Solr Catalog Provider . . . 130
7.9.4.8. WFS 1.1 Source . . . 133
7.9.4.9. GMD CSW Source . . . 134
7.9.5. Configuring Endpoints . . . 135
7.9.5.1. Configuring Catalog REST Endpoint . . . 135
7.9.5.2. Configuring CSW Endpoint . . . 135
7.9.5.3. Configuring KML Endpoint . . . 135
7.9.5.4. Configuring OpenSearch Endpoint . . . 136

7.10. Configuring for Special Deployments . . . 136
7.10.1. Multiple Installations . . . 137
7.10.1.1. Isolating SolrCloud and Zookeeper . . . 137
7.10.2. Configuring for a Fanout Proxy . . . 137
7.10.3. Configuring for a Highly Available Cluster . . . 137
7.11. Configuring UI Themes . . . 138
7.11.1. Landing Page . . . 138
7.11.1.1. Installing the Landing Page . . . 138
7.11.1.2. Configuring the Landing Page . . . 138
7.11.1.3. Customizing the Landing Page . . . 138
7.11.2. Configuring Logout Page . . . 139
7.11.3. Platform UI Themes . . . 139
7.11.3.1. Navigating to UI Theme Configuration . . . 139
7.11.3.2. Customizing the UI Theme . . . 139
7.12. Miscellaneous Configurations . . . 140
7.12.1. Configuring Thread Pools . . . 140
7.12.2. Configuring Jetty ThreadPool Settings . . . 140
7.12.3. Configuring Alerts . . . 140
7.12.3.1. Configuring Decanter Alerting . . . 140
7.12.4. Encrypting Passwords . . . 141
7.12.4.1. Encryption Command . . . 141
8. Running . . . 141
8.1. Starting . . . 142
8.1.1. Managed Services . . . 142
8.1.2. Run DDF as a Managed Service . . . 142
8.1.2.1. Running as a Service with Automatic Start on System Boot . . . 142
8.1.2.2. Karaf Documentation . . . 144
8.1.3. Run Solr as Managed Service . . . 144
8.1.4. Starting from Startup Scripts . . . 144
8.1.5. Starting as a Background Process . . . 144
8.1.6. Stopping DDF . . . 145
8.2. Maintaining . . . 146
8.2.1. Console Commands . . . 146
8.2.1.1. Console Command Help . . . 146
8.2.1.2. CQL Syntax . . . 147
8.2.1.3. Available Console Commands . . . 147
8.2.1.3.1. Catalog Commands . . . 148
8.2.1.3.2. Solr Commands . . . 151

8.2.1.3.3. Subscriptions Commands . . . 151
8.2.1.3.4. Platform Commands . . . 154
8.2.1.3.5. Persistence Store Commands . . . 155
8.2.1.4. Command Scheduler . . . 155
8.2.1.4.1. Schedule a Command . . . 155
8.2.1.4.2. Updating a Scheduled Command . . . 156
8.2.1.4.3. Output of Scheduled Commands . . . 156
8.3. Monitoring . . . 157
8.3.1. Metrics Reporting . . . 157
8.3.2. Managing Logging . . . 157
8.3.2.1. Configuring Logging . . . 157
8.3.2.2. DDF log file . . . 157
8.3.2.3. Controlling log level . . . 157
8.3.2.4. Controlling the size of the log file . . . 157
8.3.2.5. Number of backup log files to keep . . . 157
8.3.2.6. Enabling logging of inbound and outbound SOAP messages for the DDF SOAP endpoints . . . 157
8.3.2.7. Logging External Resources . . . 158
8.3.2.8. Enabling HTTP Access Logging . . . 158
8.3.2.9. Using the LogViewer . . . 159
8.4. Troubleshooting . . . 159
8.4.1. Deleted Records Are Being Displayed In The Search UI's Search Results . . . 162
9. Data Management . . . 162
9.1. Ingesting Data . . . 162
9.1.1. Ingest Command . . . 163
9.1.2. Content Directory Monitor Ingest . . . 163
9.1.3. External Methods of Ingesting Data . . . 164
9.1.4. Creating And Managing System Search Forms Through Karaf . . . 165
9.1.5. Other Methods of Ingesting Data . . . 166
9.2. Backing Up the Catalog . . . 166
9.3. Validating Data . . . 166
9.3.1. Validator Plugins on Ingest . . . 167
9.3.1.1. Validators run on ingest . . . 167
9.3.2. Configuring Schematron Services . . . 168
9.3.3. Injecting Attributes . . . 168
9.3.4. Overriding Attributes . . . 168
9.4. Removing Expired Records from the Catalog . . . 169
9.5. Automatically Added Metacard Attributes . . . 169

9.5.1. Attributes Added on Ingest . . . 169
9.5.1.1. Attributes Added by Input Transformers . . . 170
9.5.1.2. Attributes Added by Attribute Injection . . . 171
9.5.1.3. Attributes Added by Default Attribute Types . . . 171
9.5.1.4. Attributes Added by Attribute Overrides (Ingest) . . . 171
9.5.1.5. Attributes Added by Pre-Authorization Plugins . . . 171
9.5.1.6. Attributes Added by Pre-Ingest Plugins . . . 172
9.5.2. Attributes Added on Query . . . 172
9.5.2.1. Attributes Added by Attribute Overrides (Query) . . . 172
Using . . . 172
10. Using the Simple Search . . . 172
10.1. Search . . . 172
10.1.1. Search Criteria . . . 173
10.1.2. Results . . . 173
10.1.2.1. Results Summary . . . 173
10.1.2.2. Results Table . . . 174
10.1.3. Result View . . . 174
Integrating . . . 174
11. Endpoints . . . 175
11.1. Ingest Endpoints . . . 175
11.2. CRUD Endpoints . . . 175
11.3. Query Endpoints . . . 176
11.4. Content Retrieval Endpoints . . . 176
11.5. Pub-Sub Endpoints . . . 176
11.6. Endpoint Details . . . 176
11.6.1. Queries Endpoint . . . 177
11.6.1.1. Queries Endpoint Create Examples . . . 177
11.6.1.2. Queries Endpoint Retrieve All Examples . . . 178
11.6.1.3. Queries Endpoint Retrieve All Fuzzy Examples . . . 179
11.6.1.4. Queries Endpoint Retrieve Examples . . . 179
11.6.1.5. Queries Endpoint Update Examples . . . 180
11.6.1.6. Queries Endpoint Delete Examples . . . 181
11.6.2. Prometheus Endpoint . . . 181
11.6.2.1. Visualising Metrics . . . 182
11.6.3. OpenSearch Endpoint . . . 182
11.6.3.1. OpenSearch Contextual Queries . . . 182
11.6.3.1.1. Complex OpenSearch Contextual Query Format . . . 183
11.6.3.2. OpenSearch Temporal Queries . . . 183

11.6.3.3. OpenSearch Geospatial Queries . . . 184
11.6.3.4. Additional OpenSearch Query Parameters . . . 186
11.6.4. CSW Endpoint . . . 188
11.6.4.1. CSW Endpoint Create Examples . . . 188
11.6.4.2. CSW Endpoint Query Examples . . . 191
11.6.4.3. CSW Endpoint Update Examples . . . 199
11.6.4.4. CSW Endpoint Publication / Subscription Examples . . . 203
11.6.4.5. CSW Endpoint Delete Examples . . . 207
11.6.4.6. CSW Endpoint Get Capabilities Examples . . . 209
11.6.5. Catalog REST Endpoint . . . 214
11.6.5.1. Catalog REST Create Operation Examples . . . 215
11.6.5.2. Catalog REST Read Operation Examples . . . 216
11.6.5.3. Catalog Rest Update Operation Examples . . . 219
11.6.5.4. Catalog REST Delete Operation Examples . . . 219
11.6.5.5. Catalog REST Sources Operation Examples . . . 220
Developing . . . 220
12. Catalog Framework API . . . 221
13. Catalog API Design . . . 223
13.1. Ensuring Compatibility . . . 223
13.2. Catalog Framework Sequence Diagrams . . . 223
13.2.1. Error Handling . . . 224
13.2.2. Query . . . 224
13.2.3. Product Caching . . . 225
13.2.4. Product Download Status . . . 226
13.2.5. Catalog API . . . 226
13.2.5.1. Catalog API Search Interfaces . . . 226
13.2.5.2. Catalog Search Result Objects . . . 226
13.2.5.3. Search Programmatic Flow . . . 227
13.2.5.4. Sort Policies . . . 227
13.2.5.5. Product Retrieval . . . 228
13.2.5.6. Notifications and Activities . . . 229
13.3. Included Catalog Frameworks, Associated Components, and Configurations . . . 229
13.3.1. Standard Catalog Framework . . . 229
13.3.1.1. Installing the Standard Catalog Framework . . . 230
13.3.1.2. Configuring the Standard Catalog Framework . . . 230
13.3.1.3. Known Issues with Standard Catalog Framework . . . 231
13.3.2. Catalog Framework Camel Component . . . 231
13.3.2.1. Sending Messages to Catalog Framework Endpoint . . . 231

14. Transformers . . . 232
14.1. Available Input Transformers . . . 234
14.2. Available Metacard Transformers . . . 235
14.3. Available Query Response Transformers . . . 235
14.4. Transformers Details . . . 236
14.4.1. Metadata Metacard Transformer . . . 236
14.4.1.1. Installing the Metadata Metacard Transformer . . . 236
14.4.1.2. Configuring the Metadata Metacard Transformer . . . 236
14.4.1.3. Using the Metadata Metacard Transformer . . . 236
14.4.2. PPTX Input Transformer . . . 237
14.4.2.1. Installing the PPTX Input Transformer . . . 237
14.4.2.2. Configuring the PPTX Input Transformer . . . 237
14.4.3. GeoJSON Query Response Transformer . . . 237
14.4.3.1. Installing the GeoJSON Query Response Transformer . . . 237
14.4.3.2. Configuring the GeoJSON Query Response Transformer . . . 237
14.4.4. Thumbnail Metacard Transformer . . . 237
14.4.4.1. Installing the Thumbnail Metacard Transformer . . . 238
14.4.4.2. Configuring the Thumbnail Metacard Transformer . . . 238
14.4.4.3. Using the Thumbnail Metacard Transformer . . . 238
14.4.5. KML Style Mapper . . . 238
14.4.5.1. Installing the KML Style Mapper . . . 238
14.4.5.2. Configuring the KML Style Mapper . . . 238
14.4.6. Atom Query Response Transformer . . . 240
14.4.6.1. Installing the Atom Query Response Transformer . . . 241
14.4.6.2. Configuring the Atom Query Response Transformer . . . 241
14.4.6.3. Using the Atom Query Response Transformer . . . 241
14.4.7. XML Metacard Transformer . . . 243
14.4.7.1. Installing the XML Metacard Transformer . . . 244
14.4.7.2. Configuring the XML Metacard Transformer . . . 244
14.4.7.3. Using the XML Metacard Transformer . . . 244
14.4.8. XML Query Response Transformer . . . 245
14.4.8.1. Installing the XML Query Response Transformer . . . 245
14.4.8.2. Configuring the XML Query Response Transformer . . . 245
14.4.8.3. Using the XML Query Response Transformer . . . 245
14.4.9. Resource Metacard Transformer . . . 246
14.4.9.1. Installing the Resource Metacard Transformer . . . 246
14.4.9.2. Configuring the Resource Metacard Transformer . . . 246
14.4.9.3. Using the Resource Metacard Transformer . . . 246

14.4.10. GeoJSON Metacard Transformer . . . 247
14.4.10.1. Installing the GeoJSON Metacard Transformer . . . 247
14.4.10.2. Configuring the GeoJSON Metacard Transformer . . . 247
14.4.10.3. Using the GeoJSON Metacard Transformer . . . 247
14.4.11. KML Metacard Transformer . . . 248
14.4.11.1. Installing the KML Metacard Transformer . . . 248
14.4.11.2. Configuring the KML Metacard Transformer . . . 248
14.4.11.3. Using the KML Metacard Transformer . . . 248
14.4.12. XML Input Transformer . . . 251
14.4.12.1. Installing the XML Input Transformer . . . 251
14.4.12.2. Configuring the XML Input Transformer . . . 252
14.4.13. PDF Input Transformer . . . 252
14.4.13.1. Installing the PDF Input Transformer . . . 252
14.4.13.2. Configuring the PDF Input Transformer . . . 252
14.4.14. CSW Query Response Transformer . . . 252
14.4.14.1. Installing the CSW Query Response Transformer . . . 252
14.4.14.2. Configuring the CSW Query Response Transformer . . . 252
14.4.15. Video Input Transformer . . . 252
14.4.15.1. Installing the Video Input Transformer . . . 253
14.4.15.2. Configuring the Video Input Transformer . . . 253
14.4.16. GeoJSON Input Transformer . . . 253
14.4.16.1. Installing the GeoJSON Input Transformer . . . 253
14.4.16.2. Configuring the GeoJSON Input Transformer . . . 253
14.4.16.3. Using the GeoJSON Input Transformer . . . 254
14.4.16.4. Conversion to a Metacard . . . 254
14.4.16.4.1. Metacard Extensibility . . . 255
14.4.16.5. Usage Limitations of the GeoJSON Input Transformer . . . 256
14.4.17. Query Response Transformer Consumer . . . 256
14.4.17.1. Installing the Query Response Transformer Consumer . . . 256
14.4.17.2. Configuring the Query Response Transformer Consumer . . . 256
14.4.18. KML Query Response Transformer . . . 256
14.4.18.1. Installing the KML Query Response Transformer . . . 256
14.4.18.2. Configuring the KML Query Response Transformer . . . 256
14.4.18.3. Using the KML Query Response Transformer . . . 256
14.4.19. Tika Input Transformer . . . 260
14.4.19.1. Installing the Tika Input Transformer . . . 260
14.4.19.2. Configuring the Tika Input Transformer . . . 260
14.5. Mime Type Mapper . . . 260

14.5.1. DDF Mime Type Mapper . . . 261
14.5.1.1. Installing the DDF Mime Type Mapper . . . 261
14.5.1.2. Configuring DDF Mime Type Mapper . . . 261
14.6. Mime Type Resolver . . . 261
14.6.1. Tika Mime Type Resolver . . . 262
14.6.1.1. Installing the Tika Mime Type Resolver . . . 262
14.6.1.2. Configuring the Tika Mime Type Resolver . . . 262
14.6.2. Custom Mime Type Resolver . . . 262
14.6.2.1. Installing the Custom Mime Type Resolver . . . 262
14.6.2.2. Configuring the Custom Mime Type Resolver . . . 263
15. Catalog Plugins . . . 263
15.1. Types of Plugins . . . 264
15.1.1. Pre-Authorization Plugins . . . 275
15.1.1.1. Available Pre-Authorization Plugins . . . 275
15.1.2. Policy Plugins . . . 275
15.1.2.1. Available Policy Plugins . . . 275
15.1.3. Access Plugins . . . 276
15.1.3.1. Available Access Plugins . . . 276
15.1.4. Pre-Ingest Plugins . . . 277
15.1.4.1. Available Pre-Ingest Plugins . . . 277
15.1.5. Post-Ingest Plugins . . . 277
15.1.5.1. Available Post-Ingest Plugins . . . 277
15.1.6. Post-Process Plugins . . . 278
15.1.6.1. Available Post-Process Plugins . . . 278
15.1.7. Pre-Query Plugins . . . 278
15.1.7.1. Available Pre-Query Plugins . . . 278
15.1.8. Pre-Federated-Query Plugins . . . 279
15.1.8.1. Available Pre-Federated-Query Plugins . . . 279
15.1.9. Post-Query Plugins . . . 279
15.1.9.1. Available Post-Query Plugins . . . 279
15.1.10. Post-Federated-Query Plugins . . . 280
15.1.10.1. Available Post-Federated-Query Plugins . . . 280
15.1.11. Pre-Resource Plugins . . . 280
15.1.11.1. Available Pre-Resource Plugins . . . 280
15.1.12. Post-Resource Plugins . . . 280
15.1.12.1. Available Post-Resource Plugins . . . 281
15.1.13. Pre-Create Storage Plugins . . . 281
15.1.13.1. Available Pre-Create Storage Plugins . . . 281

15.1.14. Post-Create Storage Plugins . . . 281
15.1.14.1. Available Post-Create Storage Plugins . . . 281
15.1.15. Pre-Update Storage Plugins . . . 281
15.1.15.1. Available Pre-Update Storage Plugins . . . 281
15.1.16. Post-Update Storage Plugins . . . 281
15.1.16.1. Available Post-Update Storage Plugins . . . 282
15.1.17. Pre-Subscription Plugins . . . 282
15.1.17.1. Available Pre-Subscription Plugins . . . 282
15.1.18. Pre-Delivery Plugins . . . 282
15.1.18.1. Available Pre-Delivery Plugins . . . 282
15.2. Catalog Plugin Details . . . 282
15.2.1. Catalog Backup Plugin . . . 282
15.2.1.1. Installing the Catalog Backup Plugin . . . 283
15.2.1.2. Configuring the Catalog Backup Plugin . . . 283
15.2.1.3. Usage Limitations of the Catalog Backup Plugin . . . 283
15.2.2. Catalog Policy Plugin . . . 283
15.2.2.1. Installing the Catalog Policy Plugin . . . 283
15.2.2.2. Configuring the Catalog Policy Plugin . . . 283
15.2.3. Checksum Plugin . . . 284
15.2.3.1. Installing the Checksum Plugin . . . 284
15.2.3.2. Configuring the Checksum Plugin . . . 284
15.2.4. Client Info Plugin . . . 284
15.2.4.1. Related Components to the Client Info Plugin . . . 284
15.2.4.2. Installing the Client Info Plugin . . . 284
15.2.4.3. Configuring the Client Info Plugin . . . 284
15.2.5. Content URI Access Plugin . . . 284
15.2.5.1. Installing the Content URI Access Plugin . . . 284
15.2.5.2. Configuring the Content URI Access Plugin . . . 285
15.2.6. Event Processor . . . 285
15.2.6.1. Installing the Event Processor . . . 285
15.2.6.2. Configuring the Event Processor . . . 285
15.2.6.3. Usage Limitations of the Event Processor . . . 285
15.2.7. Expiration Date Pre-Ingest Plugin . . . 285
15.2.7.1. Installing the Expiration Date Pre-Ingest Plugin . . . 285
15.2.7.2. Configuring the Expiration Date Pre-Ingest Plugin . . . 286
15.2.8. Filter Plugin . . . 286
15.2.8.1. Installing the Filter Plugin . . . 287
15.2.8.2. Configuring the Filter Plugin . . . 287

15.2.9. GeoCoder Plugin . . . 287
15.2.9.1. Installing the GeoCoder Plugin . . . 288
15.2.9.2. Configuring the GeoCoder Plugin . . . 288
15.2.10. Historian Policy Plugin . . . 288
15.2.10.1. Installing the Historian Policy Plugin . . . 288
15.2.10.2. Configuring the Historian Policy Plugin . . . 288
15.2.11. JPEG2000 Thumbnail Converter . . . 288
15.2.11.1. Installing the JPEG2000 Thumbnail Converter . . . 288
15.2.11.2. Configuring the JPEG2000 Thumbnail Converter . . . 289
15.2.12. Metacard Attribute Security Policy Plugin . . . 289
15.2.12.1. Installing the Metacard Attribute Security Policy Plugin . . . 289
15.2.13. Metacard Backup File Storage Provider . . . 289
15.2.13.1. Installing the Metacard Backup File Storage Provider . . . 290
15.2.13.2. Configuring the Metacard Backup File Storage Provider . . . 290
15.2.14. Metacard Backup S3 Storage Provider . . . 290
15.2.14.1. Installing the Metacard S3 File Storage Provider . . . 290
15.2.14.2. Configuring the Metacard S3 File Storage Provider . . . 290
15.2.15. Metacard Groomer . . . 291
15.2.15.1. Installing the Metacard Groomer . . . 291
15.2.15.2. Configuring the Metacard Groomer . . . 291
15.2.16. Metacard Ingest Network Plugin . . . 291
15.2.16.1. Related Components to the Metacard Ingest Network Plugin . . . 292
15.2.16.2. Installing the Metacard Ingest Network Plugin . . . 292
15.2.16.3. Configuring the Metacard Ingest Network Plugin . . . 292
15.2.16.3.1. Useful Attributes . . . 293
15.2.16.4. Usage Limitations of the Metacard Ingest Network Plugin . . . 293
15.2.17. Metacard Resource Size Plugin . . . 293
15.2.17.1. Installing the Metacard Resource Size Plugin . . . 294
15.2.17.2. Configuring the Metacard Resource Size Plugin . . . 294
15.2.18. Metacard Validity Filter Plugin . . . 294
15.2.18.1. Related Components to the Metacard Validity Filter Plugin . . . 294
15.2.18.2. Installing the Metacard Validity Filter Plugin . . . 294
15.2.19. Metacard Validity Marker . . . 294
15.2.19.1. Related Components to the Metacard Validity Marker . . . 294
15.2.19.2. Installing Metacard Validity Marker . . . 295
15.2.19.3. Configuring Metacard Validity Marker . . . 295
15.2.19.4. Using Metacard Validity Marker . . . 295
15.2.20. Operation Plugin . . . 295

15.2.20.1. Installing the Operation Plugin . . . 295
15.2.20.2. Configuring the Operation Plugin . . . 295
15.2.21. Point of Contact Policy Plugin . . . 295
15.2.21.1. Related Components to Point of Contact Policy Plugin . . . 295
15.2.21.2. Installing the Point of Contact Policy Plugin . . . 295
15.2.21.3. Configuring the Point of Contact Policy Plugin . . . 295
15.2.22. Processing Post-Ingest Plugin . . . 296
15.2.22.1. Related Components to Processing Post-Ingest Plugin . . . 296
15.2.22.2. Installing the Processing Post-Ingest Plugin . . . 296
15.2.22.3. Configuring the Processing Post-Ingest Plugin . . . 296
15.2.23. Resource URI Policy Plugin . . . 296
15.2.23.1. Installing the Resource URI Policy Plugin . . . 296
15.2.23.2. Configuring the Resource URI Policy Plugin . . . 296
15.2.24. Security Audit Plugin . . . 296
15.2.24.1. Installing the Security Audit Plugin . . . 297
15.2.25. Security Logging Plugin . . . 297
15.2.25.1. Installing Security Logging Plugin . . . 297
15.2.25.2. Enhancing the Security Log . . . 297
15.2.26. Security Plugin . . . 297
15.2.26.1. Installing the Security Plugin . . . 297
15.2.26.2. Configuring the Security Plugin . . . 297
15.2.27. Tags Filter Plugin . . . 297
15.2.27.1. Related Components to Tags Filter Plugin . . . 297
15.2.27.2. Installing the Tags Filter Plugin . . . 298
15.2.27.3. Configuring the Tags Filter Plugin . . . 298
15.2.28. Video Thumbnail Plugin . . . 298
15.2.28.1. Installing the Video Thumbnail Plugin . . . 298
15.2.28.2. Configuring the Video Thumbnail Plugin . . . 298
15.2.29. Workspace Access Plugin . . . 299
15.2.29.1. Related Components to The Workspace Access Plugin . . . 299
15.2.29.2. Installing the Workspace Access Plugin . . . 299
15.2.29.3. Configuring the Workspace Access Plugin . . . 299
15.2.30. Workspace Pre-Ingest Plugin . . . 299
15.2.30.1. Related Components to The Workspace Pre-Ingest Plugin . . . 299
15.2.30.2. Installing the Workspace Pre-Ingest Plugin . . . 299
15.2.30.3. Configuring the Workspace Pre-Ingest Plugin . . . 299
15.2.31. Workspace Sharing Policy Plugin . . . 299
15.2.31.1. Related Components to The Workspace Sharing Policy Plugin . . . 300

15.2.31.2. Installing the Workspace Sharing Policy Plugin . . . 300
15.2.31.3. Configuring the Workspace Sharing Policy Plugin . . . 300
15.2.32. XML Attribute Security Policy Plugin . . . 300
15.2.32.1. Installing the XML Attribute Security Policy Plugin . . . 300
16. Data . . . 300
16.1. Metacards . . . 301
16.1.1. Metacard Type . . . 301
16.1.1.1. Default Metacard Type and Attributes . . . 302
16.1.1.2. Extensible Metacards . . . 302
16.1.2. Metacard Type Registry . . . 303
16.1.3. Attributes . . . 304
16.1.3.1. Attribute Types . . . 304
16.1.3.1.1. Attribute Format . . . 304
16.1.3.1.2. Attribute Naming Conventions . . . 305
16.1.3.2. Result . . . 305
16.1.4. Creating Metacards . . . 305
16.1.4.1. Limitations . . . 305
16.1.4.2. Processing Metacards . . . 306
16.1.4.3. Basic Types . . . 306
17. Operations . . . 307
18. Resources . . . 308
18.1. Content Item . . . 309
18.1.1. Retrieving Resources . . . 309
18.1.1.1. BinaryContent . . . 310
18.1.2. Retrieving Resource Options . . . 310
18.1.3. Storing Resources . . . 311
18.2. Resource Components . . . 311
18.3. Resource Readers . . . 312
18.3.1. URL Resource Reader . . . 312
18.3.1.1. Installing the URL Resource Reader . . . 312
18.3.1.2. Configuring Permissions for the URL Resource Reader . . . 313
18.3.1.3. Configuring the URL Resource Reader . . . 313
18.3.1.4. Using the URL Resource Reader . . . 313
18.4. Resource Writers . . . 314
19. Queries . . . 314
19.1. Filters . . . 314
19.1.1. FilterBuilder API . . . 315
19.1.1.1. Boolean Operators . . . 315

19.1.1.2. Attribute . . . 316
20. Action Framework . . . 316
20.1. Action Providers . . . 316
21. Asynchronous Processing Framework . . . 317
22. Eventing . . . 320
22.1. Eventing Components . . . 321
23. Security Framework . . . 321
23.1. Subject . . . 322
23.1.1. Security Manager . . . 322
23.1.2. Realms . . . 322
23.1.2.1. Authenticating Realms . . . 323
23.1.2.2. Authorizing Realms . . . 323
23.2. Security Core . . . 325
23.2.1. Security Core API . . . 325
23.2.1.1. Installing the Security Core API . . . 325
23.2.1.2. Configuring the Security Core API . . . 325
23.2.2. Security Core Implementation . . . 326
23.2.2.1. Installing the Security Core Implementation . . . 326
23.2.2.2. Configuring the Security Core Implementation . . . 326
23.2.3. Security Core Commons . . . 326
23.2.3.1. Configuring the Security Core Commons . . . 326
23.3. Security Encryption . . . 326
23.3.1. Security Encryption API . . . 326
23.3.1.1. Installing Security Encryption API . . . 326
23.3.1.2. Configuring the Security Encryption API . . . 327
23.3.2. Security Encryption Implementation . . . 327
23.3.2.1. Installing Security Encryption Implementation . . . 327
23.3.2.2. Configuring Security Encryption Implementation . . . 327
23.3.3. Security Encryption Commands . . . 327
23.3.3.1. Installing the Security Encryption Commands . . . 327
23.3.3.2. Configuring the Security Encryption Commands . . . 328
23.4. Security LDAP . . . 328
23.4.1. Embedded LDAP Server . . . 328
23.4.1.1. Installing the Embedded LDAP Server . . . 328
23.4.1.2. Configuring the Embedded LDAP . . . 328
23.4.1.3. Connecting to Standalone LDAP Servers . . . 329
23.4.1.4. Embedded LDAP Configuration . . . 329
23.4.1.5. Schemas . . . 330

23.4.1.6. Starting and Stopping the Embedded LDAP . . . 331
23.4.1.7. Limitations of the Embedded LDAP . . . 332
23.4.1.8. External Links for the Embedded LDAP . . . 332
23.4.1.9. LDAP Administration . . . 332
23.4.1.10. Downloading the Admin Tools . . . 332
23.4.1.11. Using the Admin Tools . . . 332
23.5. Security PDP . . . 334
23.5.1. Security PDP AuthZ Realm . . . 334
23.5.1.1. Configuring the Security PDP AuthZ Realm . . . 335
23.5.2. Guest Interceptor . . . 335
23.5.2.1. Installing Guest Interceptor . . . 335
23.5.2.2. Configuring Guest Interceptor . . . 336
23.6. Web Service Security Architecture . . . 336
23.6.1. Securing REST . . . 336
23.7. Security PEP . . . 339
23.7.1. Security PEP Interceptor . . . 339
23.7.1.1. Installing the Security PEP Interceptor . . . 339
23.7.1.2. Configuring the Security PEP Interceptor . . . 339
23.8. Filtering . . . 339
23.9. Expansion Service . . . 340
23.10. Federated Identity . . . 344
24. Developing DDF Components . . . 345
24.1. Developing Complementary Catalog Frameworks . . . 345
24.1.1. Simple Catalog API Implementations . . . 345
24.1.2. Use of the Whiteboard Design Pattern . . . 345
24.1.3. Recommendations for Framework Development . . . 346
24.1.4. Catalog Framework Reference . . . 346
24.1.4.1. Methods . . . 346
24.1.4.1.1. Create, Update, and Delete Methods . . . 346
24.1.4.1.2. Query Methods . . . 347
24.1.4.1.3. Resource Methods . . . 347
24.1.4.1.4. Source Methods . . . 348
24.1.4.1.5. Transform Methods . . . 348
24.1.4.2. Implementing Catalog Methods . . . 348
24.1.4.3. Dependency Injection . . . 349
24.1.4.4. OSGi Service Registry . . . 349
24.2. Developing Metacard Types . . . 350
24.2.1. Metacard Type Definition File . . . 350

24.3. Developing Metacard Validators . . . 352
24.3.1. Metacard Validator Definition . . . 352
24.4. Developing Global Attribute Validators . . . 353
24.4.1. Global Attribute Validators File . . . 353
24.5. Developing Attribute Types . . . 356
24.5.1. Attribute Type Definition File . . . 356
24.6. Developing Default Attribute Types . . . 358
24.6.1. Default Attribute Values . . . 358
24.7. Developing Attribute Injections . . . 360
24.7.1. Attribute Injection Definition . . . 360
24.8. Developing Endpoints . . . 362
24.9. Developing Input Transformers . . . 363
24.9.1. Create an XML Input Transformer using SaxEventHandlers . . . 364
24.9.2. Create an Input Transformer Using Apache Camel . . . 366
24.9.2.1. Input Transformer Design Pattern (Camel) . . . 366
24.9.3. Input Transformer Boot Service Flag . . . 367
24.10. Developing Metacard Transformers . . . 367
24.10.1. Creating a New Metacard Transformer . . . 367
24.11. Developing Query Response Transformers . . . 368
24.12. Developing Sources . . . 370
24.12.1. Implement a Source Interface . . . 370
24.12.1.1. Developing Catalog Providers . . . 371
24.12.1.2. Developing Federated Sources . . . 372
24.12.1.3. Developing Connected Sources . . . 372
24.12.1.4. Exception Handling . . . 373
24.12.1.4.1. Exception Examples . . . 373
24.12.1.4.2. External Resources for Developing Sources . . . 373
24.13. Developing Catalog Plugins . . . 373
24.13.1. Implementing Catalog Plugins . . . 375
24.13.1.1. Catalog Plugin Failure Behavior . . . 375
24.13.1.2. Implementing Pre-Ingest Plugins . . . 375
24.13.1.3. Implementing Post-Ingest Plugins . . . 376
24.13.1.4. Implementing Pre-Query Plugins . . . 376
24.13.1.5. Implementing Post-Query Plugins . . . 377
24.13.1.6. Implementing Pre-Delivery Plugins . . . 377
24.13.1.7. Implementing Pre-Subscription Plugins . . . 377
24.13.1.8. Implementing Pre-Resource Plugins . . . 378
24.13.1.9. Implementing Post-Resource Plugins . . . 378

24.13.1.10. Implementing Policy Plugins . . . 379
24.13.1.11. Implementing Access Plugins . . . 379
24.14. Developing Token Validators . . . 380
24.15. Developing STS Claims Handlers . . . 380
24.15.1. Example Requests and Responses for SAML Assertions . . . 388
24.15.2. BinarySecurityToken (CAS) SAML Security Token Samples . . . 388
24.15.3. UsernameToken Bearer SAML Security Token Sample . . . 394
24.15.4. X.509 PublicKey SAML Security Token Sample . . . 400
24.16. Developing Registry Clients . . . 407
24.17. Developing Resource Readers . . . 408
24.17.1. Creating a New `ResourceReader` . . . 408
24.17.1.1. Implementing the `ResourceReader` Interface . . . 408
24.17.1.2. retrieveResource . . . 408
24.17.1.3. Implement `retrieveResource()` . . . 409
24.17.1.4. `getSupportedSchemes` . . . 410
24.17.1.5. Export to OSGi Service Registry . . . 410
24.18. Developing Resource Writers . . . 410
24.18.1. Create a New `ResourceWriter` . . . 411
24.19. Developing Filters . . . 412
24.19.1. Units of Measure . . . 412
24.19.2. Filter Examples . . . 413
24.19.2.1. Contextual Searches . . . 414
24.19.2.1.1. Tree View of Creating Filters . . . 415
24.19.2.1.2. XML View of Creating Filters . . . 415
24.19.2.2. Fuzzy Operations . . . 415
24.19.3. Parsing Filters . . . 416
24.19.3.1. Interpreting a Filter to Create SQL . . . 417
24.19.3.2. Interpreting a Filter to Create XQuery . . . 417
24.19.3.2.1. FilterAdapter/Delegate Process for Figure Parsing . . . 418
24.19.3.2.2. FilterVisitor Process for Figure Parsing . . . 418
24.19.4. Filter Profile . . . 419
24.19.4.1. Role of the OGC Filter . . . 419
24.19.4.2. Catalog Filter Profile . . . 419
24.19.4.2.1. Comparison Operators . . . 420
24.19.4.2.2. Logical Operators . . . 421
24.19.4.2.3. Temporal Operators . . . 421
24.19.4.2.4. Spatial Operators . . . 422
24.20. Developing Filter Delegates . . . 423

24.20.1. Creating a New Filter Delegate . . . 423
24.20.1.1. Implementing the Filter Delegate . . . 423
24.20.1.2. Throwing Exceptions . . . 423
24.20.1.3. Using the Filter Adapter . . . 423
24.20.1.4. Filter Support . . . 424
24.21. Developing Action Components . . . 425
24.21.1. Action Component Naming Convention . . . 426
24.21.1.1. Action Component Taxonomy . . . 426
24.22. Developing Query Options . . . 427
24.22.1. Evaluating a query . . . 427
24.22.2. Commons-DDF Utilities . . . 428
24.22.2.1. FuzzyFunction . . . 428
24.22.2.2. XPathHelper . . . 428
24.23. Configuring Managed Service Factory Bundles . . . 428
24.23.1. File Format . . . 429
24.24. Developing XACML Policies . . . 432
24.24.1. XACML Policy Attributes . . . 432
24.24.2. XACML Policy Subject . . . 432
24.24.3. XACML Policy Resource . . . 432
24.24.4. Using a XACML Policy . . . 432
24.25. Assuring Authenticity of Bundles and Applications . . . 433
24.25.1. Prerequisites . . . 433
24.25.2. Signing a JAR/KAR . . . 433
24.25.2.1. Verifying a JAR/KAR . . . 434
24.26. WFS Services . . . 434
24.27. JSON Definition Files . . . 436
24.27.1. Definition File Format . . . 436
24.27.2. Deploying Definition Files . . . 436
24.28. Developing Subscriptions . . . 437
24.28.1. Subscription Lifecycle . . . 437
24.28.1.1. Creation . . . 437
24.28.1.2. Evaluation . . . 437
24.28.1.3. Update Evaluation . . . 437
24.28.1.4. Durability . . . 437
24.28.2. Creating a Subscription . . . 438
24.28.2.1. Event Processing and Notification . . . 438
24.28.2.1.1. Using DDF Implementation . . . 439
24.28.2.2. Delivery Method . . . 439

24.29. Contributing to Documentation . . . 439
24.29.1. Editing Existing Documentation . . . 440
24.29.2. Adding New Documentation Content . . . 441
24.29.3. Creating a New Documentation Template . . . 441
24.29.4. Extending Documentation in Downstream Distributions . . . 442
25. Development Guidelines . . . 442
25.1. Contributing . . . 442
25.2. OSGi Basics . . . 442
25.2.1. Packaging Capabilities as Bundles . . . 443
25.2.1.1. Creating a Bundle . . . 444
25.2.1.1.1. Bundle Development Recommendations . . . 444
25.2.1.1.2. Maven Bundle Plugin . . . 444
25.2.1.2. Third Party and Utility Bundles . . . 445
25.2.1.3. Deploying a Bundle . . . 445
25.2.1.4. Verifying Bundle State . . . 446
25.3. High Availability Guidance . . . 446
Appendices . . . 446
Appendix A: Application References . . . 447
26. Application Reference . . . 447
26.1. Admin Application Reference . . . 447
26.1.1. Admin Application Prerequisites . . . 447
26.1.2. Installing the Admin Application . . . 447
26.1.3. Configuring the Admin Application . . . 447
26.2. Catalog Application Reference . . . 449
26.2.1. Catalog Application Prerequisites . . . 449
26.2.2. Installing the Catalog Application . . . 449
26.2.3. Configuring the Catalog Application . . . 449
26.3. Platform Application Reference . . . 473
26.3.1. Platform Application Prerequisites . . . 473
26.3.2. Installing Platform . . . 473
26.3.3. Configuring the Platform Application . . . 474
26.4. Security Application Reference . . . 478
26.4.1. Security Prerequisites . . . 478
26.4.2. Installing Security . . . 478
26.4.3. Configuring the Security Application . . . 479
26.5. Solr Catalog Application Reference . . . 494
26.5.1. Solr Catalog Prerequisites . . . 494
26.5.2. Installing Solr Catalog . . . 495

26.5.3. Configuring the Solr Catalog Application . . . 495
26.6. Spatial Application Reference . . . 495
26.6.1. Offline Gazetteer Service . . . 496
26.6.1.1. Spatial Gazetteer Console Commands . . . 496
26.6.2. Standalone Solr Offline Gazetteer Feature . . . 497
26.6.2.1. Installing Standalone Solr Offline Gazetteer . . . 497
26.6.2.2. Special Note Regarding Installation . . . 497
26.6.2.3. Building the suggester index on Standalone Solr Offline Gazetteer . . . 497
26.6.3. Spatial Prerequisites . . . 498
26.6.4. Installing Spatial . . . 498
26.6.5. Configuring the Spatial Application . . . 498
26.7. Search UI Application Reference . . . 519
26.7.1. Search UI Prerequisites . . . 520
26.7.2. Installing Search UI . . . 520
26.7.3. Configuring the Search UI Application . . . 520
Appendix B: Application Whitelists . . . 523
B.1. Packages Removed From Whitelist . . . 523
B.2. Catalog Whitelist . . . 526
B.3. Platform Whitelist . . . 530
B.4. Security Whitelist . . . 531
B.5. Solr Catalog Whitelist . . . 532
B.6. Search UI Whitelist . . . 532
27. DDF Dependency List . . . 533
27.1. DDF 2.30.1 Dependency List . . . 533
28. Hardening Checklist . . . 542
Appendix C: Metadata Reference . . . 543
C.1. Common Metadata Attributes . . . 543
C.2. File Format-specific Attributes . . . 545
C.2.1. Mp4 Additional Attribute . . . 545
C.2.2. All File Formats Supported . . . 545
C.3. Catalog Taxonomy Definitions . . . 561
C.3.1. Core Attributes . . . 561
C.3.2. Associations Attributes . . . 565
C.3.3. Contact Attributes . . . 565
C.3.4. DateTime Attributes . . . 567
C.3.5. History Attributes . . . 567
C.3.6. Location Attributes . . . 568
C.3.7. Media Attributes . . . 569

C.3.8. Metacard Attributes . . . 570
C.3.9. Security Attributes . . . 571
C.3.10. Topic Attributes . . . 572
C.3.11. Validation Attributes . . . 572
Index . . . 573

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.

Introduction

1. About DDF

1.1. Introducing DDF

Distributed Data Framework (DDF) is a free and open-source common data layer that abstracts services and business logic from underlying data structures to enable rapid integration of new data sources.

Licensed under LGPL, DDF is an interoperability platform that provides secure and scalable discovery and retrieval from a wide array of disparate sources.

DDF is:

- a flexible and modular integration framework.
- built to "unzip and run" even when scaled to large enterprise systems.
- primarily focused on data integration, enabling clients to insert, query, and transform information from disparate data sources via the DDF Catalog.

1.2. Component Applications

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

Admin Application

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

Catalog Application

Provides a framework for storing, searching, processing, and transforming information. Clients typically perform local and/or federated query, create, read, update, and delete (QCRUD) operations against the Catalog. At the core of the Catalog functionality is the **Catalog Framework**, which routes all requests and responses through the system, invoking additional processing per the system configuration.

Platform Application

The Core application of the distribution. The Platform application contains the fundamental building blocks to run the distribution.

Security Application

Provides authentication, authorization, and auditing services for the DDF. It is both a framework that developers and integrators can extend and a reference implementation that meets security

requirements.

Solr Catalog Application

Includes the Solr Catalog Provider, an implementation of the Catalog Provider using Apache Solr as a data store.

Spatial Application

Provides OGC services, such as CSW, WCS, WFS, and KML.

Search UI

Allows a user to search for records in the local Catalog (provider) and federated sources. Results of the search are returned and displayed on a globe or map, providing a visual representation of where the records were found.

2. Documentation Guide

The DDF documentation is organized by audience.

Core Concepts

This introduction section is intended to give a high-level overview of the concepts and capabilities of DDF.

Administrators

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

Users

Using | Users interact with the system to search data stores. Use this section to navigate the various user interfaces available in DDF.

Integrators

Integrating | Integrators use the existing applications to support their external frameworks. This section provides details for finding, accessing, and using the components of DDF.

Developers

Developing | Developers build or extend the functionality of the applications.

2.1. Documentation Conventions

The following conventions are used within this documentation:

2.1.1. Customizable Values

Many values used in descriptions are customizable and should be changed for specific use cases. These

values are denoted by `< >`, and by `[[ ]]` when within XML syntax. When using a real value, the placeholder characters should be omitted.

2.1.2. Code Values

Java objects, lines of code, or file properties are denoted with the `Monospace` font style. Example: `ddf.catalog.CatalogFramework`

2.1.3. Hyperlinks

Some hyperlinks (such as `/admin`) within the documentation assume a locally running installation of DDF. Simply change the hostname if accessing a remote host.

Hyperlinks that take the user away from the DDF documentation are marked with an `external link` (↗) icon.

2.2. Support

Questions about DDF should be posted to the ddf-users forum ↗ or ddf-developers forum ↗, to be responded to quickly by a member of the DDF team.

2.2.1. Documentation Updates

The most current DDF documentation is available at DDF Documentation ↗.

3. Core Concepts

This introduction section is intended to give a high-level overview of the concepts and capabilities of DDF.

3.1. Introduction to Search

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

Search Types

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

Text Search

A text search is used when searching for textual information. It searches all textual fields by default, although it is possible to refine searches to a text search on a single metadata attribute. Text

searches may use wildcards, logical operators, and approximate matches.

Spatial Search

A spatial search is used for Area of Interest (AOI) searches. Polygon and point radius searches are supported.

Temporal Search

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

Datatype Search

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

3.2. Introduction to Metadata

In DDF, resources are the files, reports, or documents of interest to users of the system.

Metadata is information about those resources, organized into a schema to make search possible. The Catalog stores this metadata and allows access to it. Metacards are single instances of metadata, representing a single resource, in the Catalog. Metacards follow one of several schemas to ensure reliable, accurate, and complete metadata. Essentially, Metacards function as containers of metadata.

3.3. Introduction to Ingest

Ingest is the process of bringing data resources, metadata, or both into the catalog to enable search, sharing, and discovery. Ingested files are transformed into a neutral format that can be searched against as well as migrated to other formats and systems. See Ingesting Data for the various methods of ingesting data.

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

3.4. Introduction to Resources

The Catalog Framework can interface with storage providers to provide storage of resources to specific types of storage, such as file system, relational database, XML database. A default file system implementation is provided by default.

Storage providers act as a proxy between the Catalog Framework and the mechanism storing the content. Storage providers expose the storage mechanism to the Catalog Framework. Storage plugins provide pluggable functionality that can be executed either immediately before or immediately after content has been stored or updated.

Storage providers provide the capability to the Catalog Framework to create, read, update, and delete resources in the content repository.

See Data Management for more information on specific file types supported by DDF.

3.5. Introduction to the Catalog Framework

The Catalog Framework wires all the Catalog components together.

It is responsible for routing Catalog requests and responses to the appropriate source, destination, federated system, etc.

Endpoints send Catalog requests to the Catalog Framework. The Catalog Framework then invokes Catalog Plugins, Transformers, and Resource Components as needed before sending requests to the intended destination, such as one or more Sources.

The Catalog Framework decouples clients from service implementations and provides integration points for Catalog Plugins and convenience methods for Endpoint developers.

3.6. Introduction to Federation and Sources

Federation is the ability of the DDF to query other data sources, including other DDFs. By default, the DDF is able to federate using OpenSearch and CSW protocols. The minimum configuration necessary to configure those federations is a query address.

Federation enables constructing dynamic networks of data sources that can be queried individually or aggregated into specific configuration to enable a wider range of accessibility for data and data resources.

Federation provides the capability to extend the DDF enterprise to include Remote Sources, which may include other instances of DDF. The Catalog handles all aspects of federated queries as they are sent to the Catalog Provider and Remote Sources, as they are processed, and as the query results are returned. Queries can be scoped to include only the local Catalog Provider (and any Connected Sources), only specific Federated Sources, or the entire enterprise (which includes all local and Remote Sources). If the query is federated, the Catalog Framework passes the query to a Federation Strategy, which is responsible for querying each federated source that is specified. The Catalog Framework is also responsible for receiving the query results from each federated source and returning them to the client in the order specified by the particular federation strategy used. After the federation strategy handles the results, the Catalog returns them to the client through the Endpoint. Query results are returned from a federated query as a list of metacards. The source ID in each metacard identifies

the Source from which the metacard originated.

3.7. Introduction to Events and Subscriptions

DDF can be configured to receive notifications whenever metadata is created, updated, or deleted in any federated sources. Creations, updates, and deletions are collectively called **Events**, and the process of registering to receive them is called **Subscription**.

The behavior of these subscriptions is consistent, but the method of configuring them is specific to the Endpoint used.

3.8. Introduction to Endpoints

Endpoints expose the Catalog Framework to clients using protocols and formats that the clients understand.

Endpoint interface formats encompass a variety of protocols, including (but not limited to):

- SOAP Web services
- RESTful services
- JMS
- JSON
- OpenSearch

The endpoint may transform a client request into a compatible Catalog format and then transform the response into a compatible client format. Endpoints may use Transformers to perform these transformations. This allows an endpoint to interact with Source(s) that have different interfaces. For example, an OpenSearch Endpoint can send a query to the Catalog Framework, which could then query a federated source that has no OpenSearch interface.

Endpoints are meant to be the only client-accessible components in the Catalog.

3.9. Standards Supported by DDF

DDF incorporates support for many common Service, Metadata, and Security standards, as well as many common Data Formats.

3.9.1. Catalog Service Standards

Service standards are implemented within Endpoints and/or Sources. Standards marked **Experimental** are functional and have been tested, but are subject to change or removal during the incubation period.

Table 1. Catalog Service Standards Included with DDF

Standard (public standards linked where available)	Endpoints	Sources	Status
Open Geospatial Consortium Catalog Service for the Web (OGC CSW) 2.0.1/2.0.2	CSW Endpoint	Geographic MetaData extensible markup language (GMD) CSW Source	Supported
OGC Web Feature Service WFS 1.1/2.0		WFS 1.1 Source, WFS 2.0 Source	Supported
OpenSearch	OpenSearch Endpoint	OpenSearch Source	Supported
Atlassian Confluence®		Atlassian Confluence® Federated Source	Supported

3.9.2. Data Formats

DDF has extended capabilities to extract rich metadata from many common data formats if those attributes are populated in the source document. See appendix for a complete list of file formats that can be ingested with limited metadata coverage. Metadata standards use XML or JSON, or both.

Table 2. Data Formats Included in DDF

Format	File Extensions	Additional Metadata Attributes Available (if populated)
Word Document	`doc, docx, dotx, docm`	Standard attributes
PowerPoint	`ppt, pptx`	Standard attributes
Excel	`xls, xlsx`	Standard attributes
PDF	`pdf`	Standard attributes
GeoPDF	`pdf`	Standard attributes
geojson	`json,js`	Standard attributes
html	`htm, html`	Standard attributes
jpeg	`jpeg, jpeg2000`	Standard attributes and additional Media attributes
mp2	`mp2, MPEG2`	Standard attributes and additional Media attributes
mp4	`mp4`	Standard attributes, additional Media attributes, and mp4 additional attribute
WMV	`wmv`	Standard attributes

Format	File Extensions	Additional Metadata Attributes Available (if populated)
AVIs	`avi`	Standard attributes
Keyhole Markup Language (KML)	`kml`	Standard attributes
Dublin Core	`n/a`	Standard attributes

3.9.3. Security Standards

DDF makes use of these security standards to protect the system and interactions with it.

Table 3. Attribute Stores Provided by DDF

Standard	Support Status
Lightweight Directory Access Protocol (LDAP/LDAPS)	Supported
Azure Active Directory	Supported

Table 4. Cryptography Standards Provided by DDF

Standard	Support Status
Transport Layer Security (TLS) v1.1 & v1.2	Supported
Cipher Suites • 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	Supported

Table 5. Transport Protocols Provided by DDF

Standard	Support Status
HyperText Transport Protocol (HTTP) / HyperText Transport Protocol Secure (HTTPS)	Supported
Lightweight Directory Access (LDAP/LDAPS)	Supported

Table 6. Single Sign On Standards Provided by DDF

Standard	Support Status
SAML 2.0 Web SSO Profile	Supported
SAML Enhanced Client or Proxy (ECP)	Supported

Standard	Support Status
Central Authentication Service (CAS)	Supported

Table 7. Security and SSO Endpoints Provided by DDF

Standard	Support Status
Security Token Service (STS)	Supported
Identity Provider (IdP)	Supported
Service Provider (SP)	Supported

Table 8. Authentication Standards Provided by DDF

Standard	Support Status
Public Key Infrastructure (PKI)	Supported
Basic Authentication	Supported
SAML	Supported
Central Authentication Service (CAS)	Supported

3.10. Introduction to High Availability

DDF can be made highly available. In this context, High Availability is defined as the ability for DDF to be continuously operational with very little down time.

In a Highly Available Cluster, DDF has failover capabilities when a DDF node fails.

NOTE The word "node," from a High Availability perspective, describes one of the two DDF systems running within the Highly Available Cluster. Though there are multiple systems running with the Highly Available Cluster, it is still considered a single DDF from a user's perspective or from other DDFs' perspectives.

This setup consists of a SolrCloud instance, 2 DDF nodes connected to that SolrCloud, and a failover proxy that sits in front of those 2 nodes. One of the DDF nodes is arbitrarily chosen to be the active node, and the other is the "hot standby" node. It is called a "hot standby" node because it is ready to receive traffic even though it's not currently receiving any. The failover proxy routes all traffic to the active node. If the active node fails for any reason, the standby node becomes active and the failover proxy routes all traffic to the new active node. See the below diagrams for more detail.

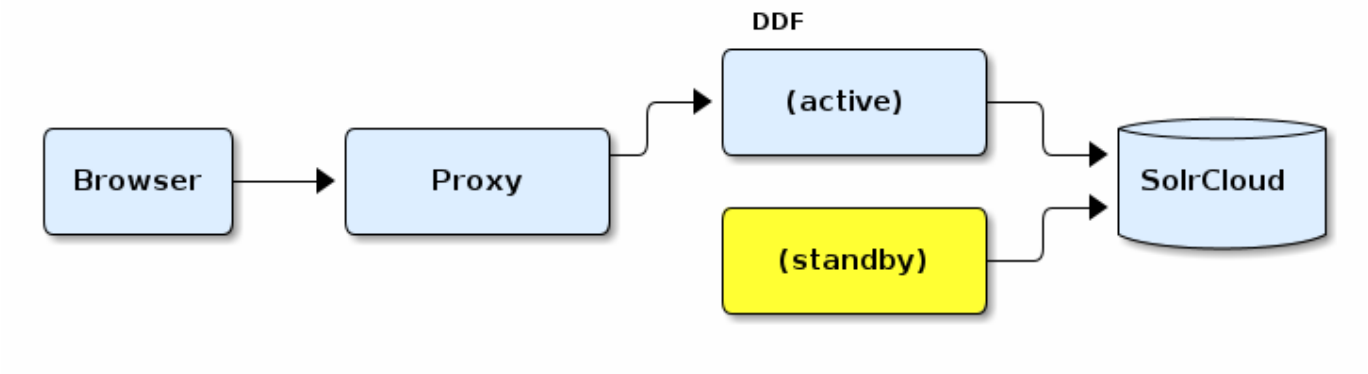


Figure 1. Highly Available Cluster

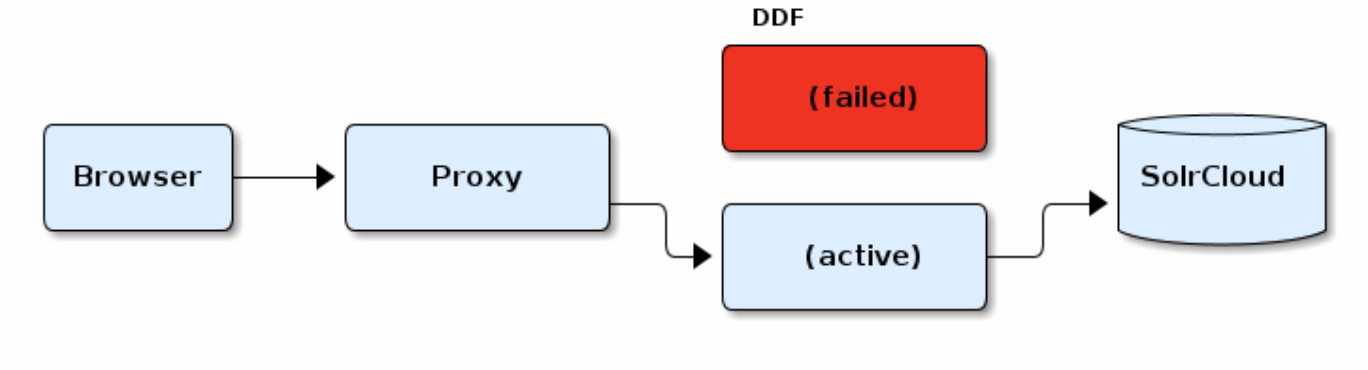


Figure 2. Highly Available Cluster (after failover)

There are special procedures for initial setup and configuration of a highly available DDF. See High Availability Initial Setup and High Availability Configuration for those procedures.

3.10.1. High Availability Supported Capabilities

Only these capabilities are supported in a Highly Available Cluster. For a detailed list of features, look at the `ha.json` file located in `<DDF_HOME>/etc/profiles/`.

- User Interface:
  - Simple
- Catalog:
  - Validation
  - Plug-ins: Expiration Date, JPEG2000, Metacard Validation, Schematron, Versioning
  - Transformers
  - Content File System Storage Provider
- Platform:
  - Actions

    - Configuration
    - Notifications
    - Persistence
    - Security: Audit, Encryption
- Solr
- Security
- Thirdy Party:
    - CXF
    - Camel
- Endpoints:
    - REST Endpoint
    - CSW Endpoint
    - OpenSearch Endpoint

4. Quick Start Tutorial

This quick tutorial enables installation, configuration, and usage of a basic instance of DDF.

NOTE	This tutorial is intended for setting up a test, demonstration, or trial installation of DDF. For complete installation and configuration steps, see Installing.

These steps demonstrate:

- ☑ Prerequisites.
- ☑ Quick Install of SolrCloud.
- ☑ Quick Install of DDF.

4.1. Installing (Quick Start)

These are the basic requirements to set up the environment to run a DDF.

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

4.1.1. Quick Install Prerequisites

Hardware Requirements (Quick Install)

- At least 4096 MB of memory for DDF.
  - This amount can be increased to support memory-intensive applications. See Memory Considerations.

Java Requirements (Quick Install)

Follow the instructions outlined in Java Requirements.

WARNING	*Check System Time* Prior to installing DDF, ensure the system time is accurate to prevent federation issues.

4.1.2. Quick Install of SolrCloud

1. Download a preconfigured Solr distribution zip file.
2. Unzip the Solr zip file.
3. Run `<solr_directory>/bin/solr -e cloud`.
   a. Press enter to default to 2 nodes.
   b. Enter 8994 for node 1 port.
   c. Enter 8995 for node 2 port.
   d. Press enter for all other prompts to accept defaults.

4.1.3. Quick Install of DDF

WARNING	The DDF installation cannot succeed unless SolrCloud is running. See Quick Install of SolrCloud for instructions.

1. Download the DDF zip file.
2. Install DDF by unzipping the zip file.

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.

NOTE	If and only if a JDK is installed, the jar command may be used; otherwise, another archiving utility that does not have issue with long paths should be installed

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
```

3. This creates an installation directory, which is typically created with the name and version of the application. This installation directory is referred to as `<DDF_HOME>`. (Substitute the actual directory name.)
4. Edit `<DDF_HOME>/etc/custom.system.properties` and update `solr.cloud.zookeeper=localhost:2181` to `solr.cloud.zookeeper=localhost:9994`
5. Start DDF by running the `<DDF_HOME>/bin/ddf` script (or `ddf.bat` on Windows).
6. Startup may take a few minutes.
    a. Optionally, a `system:wait-for-ready` command (aliased to `wfr`) can be used to wait for startup to complete.
7. The Command Console displays.

Command Console Prompt

```
ddf@local>
```

4.1.4. Quick Install of DDF on a remote headless server

If DDF is being installed on a remote server that has no user interface, the hostname must be updated in the configuration files and certificates.

NOTE	Do not replace *all* instances of `localhost`, only those specified.

Configuring with a new hostname

1. Update the <DDF_HOME>/etc/custom.system.properties file. The entry `org.codice.ddf.system.hostname=localhost` should be updated to `org.codice.ddf.system.hostname=<HOSTNAME>`.
2. Update the <DDF_HOME>/etc/users.properties file. Change the `localhost=localhost[…]` entry to <HOSTNAME>=<HOSTNAME>. (Keep the rest of the line as is.)
3. Update the <DDF_HOME>/etc/users.attributes file. Change the `"localhost"` entry to `"<HOSTNAME>"`.

4. From the console go to <DDF_HOME>/etc/certs and run the appropriate script.
    a. *NIX: `sh CertNew.sh -cn <hostname> -san "DNS:<hostname>"`.
    b. Windows: `CertNew -cn <hostname> -san "DNS:<hostname>"`.
5. Proceed with starting the system and continue as usual.

Configuring with an IP address

1. Update the <DDF_HOME>/etc/custom.system.properties file. The entry `org.codice.ddf.system.hostname=localhost` should be updated to `org.codice.ddf.system.hostname=<IP>`.
2. Update the <DDF_HOME>/etc/users.properties file. Change the `localhost=localhost[…]` entry to `<IP>=<IP>`. (Keep the rest of the line as is.)
3. Update the <DDF_HOME>/etc/users.attributes file. Change the `"localhost"` entry to `"<IP>"`.
4. From the console go to <DDF_HOME>/etc/certs and run the appropriate script.
    a. *NIX: `sh CertNew.sh -cn <IP> -san "IP:<IP>"`.
    b. Windows: `CertNew -cn <IP> -san "IP:<IP>"`.
5. Proceed with starting the system and continue as usual.

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
```

- (This file may need permissions changed to allow write access).
- For the change to take effect, a restart is required.
    1. *nix Restart Command

```
init 6
```

4.2. Certificates (Quick Start)

DDF comes with a default keystore that contains certificates. This allows the distribution to be unzipped and run immediately. If these certificates are sufficient for testing purposes, proceed to Configuring (Quick Start).

To test federation using 2-way TLS, the default keystore certificates needs to be replaced, using either

the included Demo Certificate Authority or by Creating Self-signed Certificates.

If the installer was used to install the DDF and a hostname other than "localhost" was given, the user is prompted to upload new trust/key stores.

If the hostname is `localhost` or, if the hostname was changed *after* installation, the default certificates do not allow access to the DDF instance from another machine over HTTPS (now the default for many services). The Demo Certificate Authority must be replaced with certificates that use the fully-qualified hostname of the server running the DDF instance.

4.2.1. Demo Certificate Authority (CA)

DDF comes with a populated truststore containing entries for many public certificate authorities, such as Go Daddy and Verisign. It also includes an entry for the DDF Demo Root CA. This entry is a self-signed certificate used for testing. It enables DDF to run immediately after unzipping the distribution. The keys and certificates for the DDF Demo Root CA are included as part of the DDF distribution. This entry must be removed from the truststore before DDF can operate securely.

4.2.1.1. Creating New Server Keystore Entry with the CertNew Scripts

To create a private key and certificate signed by the Demo Certificate Authority, use the provided scripts. To use the scripts, run them out of the `<DDF_HOME>/etc/certs` directory.

***NIX Demo CA Script**

For *NIX, use the `CertNew.sh` script.

```
sh CertNew.sh [-cn <cn>|-dn <dn>] [-san <tag:name,tag:name,⋯>]
```

where:

- `<cn>` represents a fully qualified common name (such as `"<FQDN>"`, where `<FQDN>` could be something like `cluster.yoyo.com`)
- `<dn>` represents a distinguished name as a comma-delimited string (such as `"c=US, st=California, o=Yoyodyne, l=San Narciso, cn=<FQDN>"`)
- `<tag:name,tag:name,⋯>` represents optional subject alternative names to be added to the generated certificate (such as `"DNS:<FQDN>,DNS:node1.<FQDN>,DNS:node2.<FQDN>"`). The format for subject alternative names is similar to the OpenSSL X509 configuration format. Supported tags are:
  - `email` - email subject
  - `URI` - uniformed resource identifier
  - `RID` - registered id
  - `DNS` - hostname

- `IP` - ip address (V4 or V6)
- `dirName` - directory name

If no arguments specified on the command line, `hostname -f` is used as the common-name for the certificate.

Windows Demo CA Script

For Windows, use the `CertNew.cmd` script.

```
CertNew (-cn <cn>|-dn <dn>) [-san "<tag:name,tag:name,⋯>"]
```

where:

- `<cn>` represents a fully qualified common name (such as `"<FQDN>"`, where `<FQDN>` could be something like `cluster.yoyo.com`)
- `<dn>` represents a distinguished name as a comma-delimited string (such as `"c=US, st=California, o=Yoyodyne, l=San Narciso, cn=<FQDN>"`)
- `<tag:name,tag:name,⋯>` represents optional subject alternative names to be added to the generated certificate (such as `"DNS:<FQDN>,DNS:node1.<FQDN>,DNS:node2.<FQDN>"`). The format for subject alternative names is similar to the OpenSSL X509 configuration format. Supported tags are:
  - `email` - email subject
  - `URI` - uniformed resource identifier
  - `RID` - registered id
  - `DNS` - hostname
  - `IP` - ip address (V4 or V6)
  - `dirName` - directory name

The `CertNew` scripts:

- Create a new entry in the server keystore.
- Use the hostname as the fully qualified domain name (FQDN) when creating the certificate.
- Adds the specified subject alternative names if any.
- Use the Demo Certificate Authority to sign the certificate so that it is trusted by the default configuration.

To install a certificate signed by a different Certificate Authority, see Managing Keystores.

After this proceed to Updating Settings After Changing Certificates.

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

4.2.1.2. Dealing with Lack of DNS

In some cases DNS may not be available, and the system needs to be configured to work with IP addresses.

Options can be given to the `CertNew` Scripts to generate certs that work in this scenario.

***NIX**

From <DDF_HOME>/etc/certs/ run:

```
sh CertNew.sh -cn <IP> -san "IP:<IP>"
```

Windows

From <DDF_HOME>/etc/certs/ run:

```
CertNew -cn <IP> -san "IP:<IP>"
```

After this proceed to Updating Settings After Changing Certificates, and be sure to use the IP address instead of the FQDN.

4.2.2. Creating Self-Signed Certificates

If using the Demo CA is not desired, DDF supports creating self-signed certificates with a self-signed certificate authority. This is considered an advanced configuration.

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

For this example, the following tools are used:

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

4.2.2.1. Creating a custom CA Key and Certificate

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

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

4.2.2.2. Sign Certificates Using the custom CA

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

1. Generate a private key and a Certificate Signing Request (CSR).
   ```
   $> openssl req -newkey rsa:1024 -keyout tokenissuer.key -out tokenissuer.req
   ```
2. Sign the certificate by the CA.
   ```
   $> openssl ca -out tokenissuer.crt -infiles tokenissuer.req
   ```

These certificates are used during system configuration to replace the default certificates.

4.2.3. Updating Settings After Changing Certificates

After changing the certificates, it is necessary to update the system user and the `org.codice.ddf.system.hostname` property with the value of either the FQDN or the IP.

FQDNs should be used wherever possible. In the absence of DNS, however, IP addresses can be used.

Replace `localhost` with the FQDN or the IP in `<DDF_HOME>/etc/users.properties`, `<DDF_HOME>/etc/users.attributes`, and `<DDF_HOME>/etc/custom.system.properties`.

TIP	On linux this can be accomplished with a single command: `sed -i 's/localhost/<FQDN|IP>/g' <DDF_HOME>/etc/users.* <DDF_HOME>/etc/custom.system.properties`

Finally, restart the DDF instance. Navigate to the Admin Console to test changes.

4.3. Configuring (Quick Start)

Set the configurations needed to run DDF.

1. In a browser, navigate to the Admin Console at https://{FQDN}:{PORT}/admin.
   a. The Admin Console may take a few minutes to start up.
2. Enter the default username of `admin` and the password of `admin`.

3. Follow the installer prompts for a standard installation.
    a. Click start to begin the setup process.
    b. Configure guest claims attributes or use defaults.
        i. See Configuring Guest Access for more information about the Guest user.
        ii. **All users are automatically granted these permissions.**
        iii. **Guest users are not able to ingest data with more restrictive markings than the guest claims.**
        iv. **Any data ingested that has more restrictive markings than these guest claims is not visible to Guest users.**
    c. Select **Standard Installation**.
        i. This step may take several minutes to complete.
    d. On the System Configuration page, configure any port or protocol changes desired and add any keystores/truststores needed.
        i. See Certificates (Quick Start) for more details.
    e. Click **Next**
    f. Click **Finish**

4.4. Adding New User Roles

Users or user groups can be customized to enable or restrict access to specific certain system functions. Define a custom role and assign it to users who need specific access.

For an example, limit the replication admin view to users with a `sr_analyst` role.

Define a new role

1. Navigate to the Admin Console.
2. Navigate to the Security app.
3. Navigate to the **Configuration** tab.
4. Select the **Web Context Policy Manager** Configuration.
5. Select **Required Attributes**.
6. Define the contexts to be restricted.
    a. Enter `<CONTEXT>={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=<ROLE_NAME>`
        i. For example, enter `/admin/replication={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=sr_analyst}`

ii. Select the + sign to add another required attribute.

iii. Enter `/admin/hub/graphql={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=sr_analyst}`

Assign the new role to users

1. Add the new role (`sr_analyst`) to selected users in the `users.properties` file. Add the role to the line of each chosen user.
   a. For example, `user1=password,sr_analyst`
2. Restart Alliance to apply the changes.
   a. `/app/bin/stop`
   b. `/app/bin/start`

In the example given, the `/admin/replication` context is only accessible to users with the `sr_analyst` role. Other roles can be designated as needed by substituting the `<CONTEXT>` and the `<ROLE_NAME>` as needed.

Managing

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

5. 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.

5.1. 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.

5.2. 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.

5.2.1. 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.

6. 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.

6.1. 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.

6.1.1. Hardware Requirements

Table 9. 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 10. 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

6.1.2. 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
```

6.2. 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.

6.2.1. 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.

6.2.1.1. 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

6.2.1.2. 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
```

6.2.1.3. 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
```

6.2.2. 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.

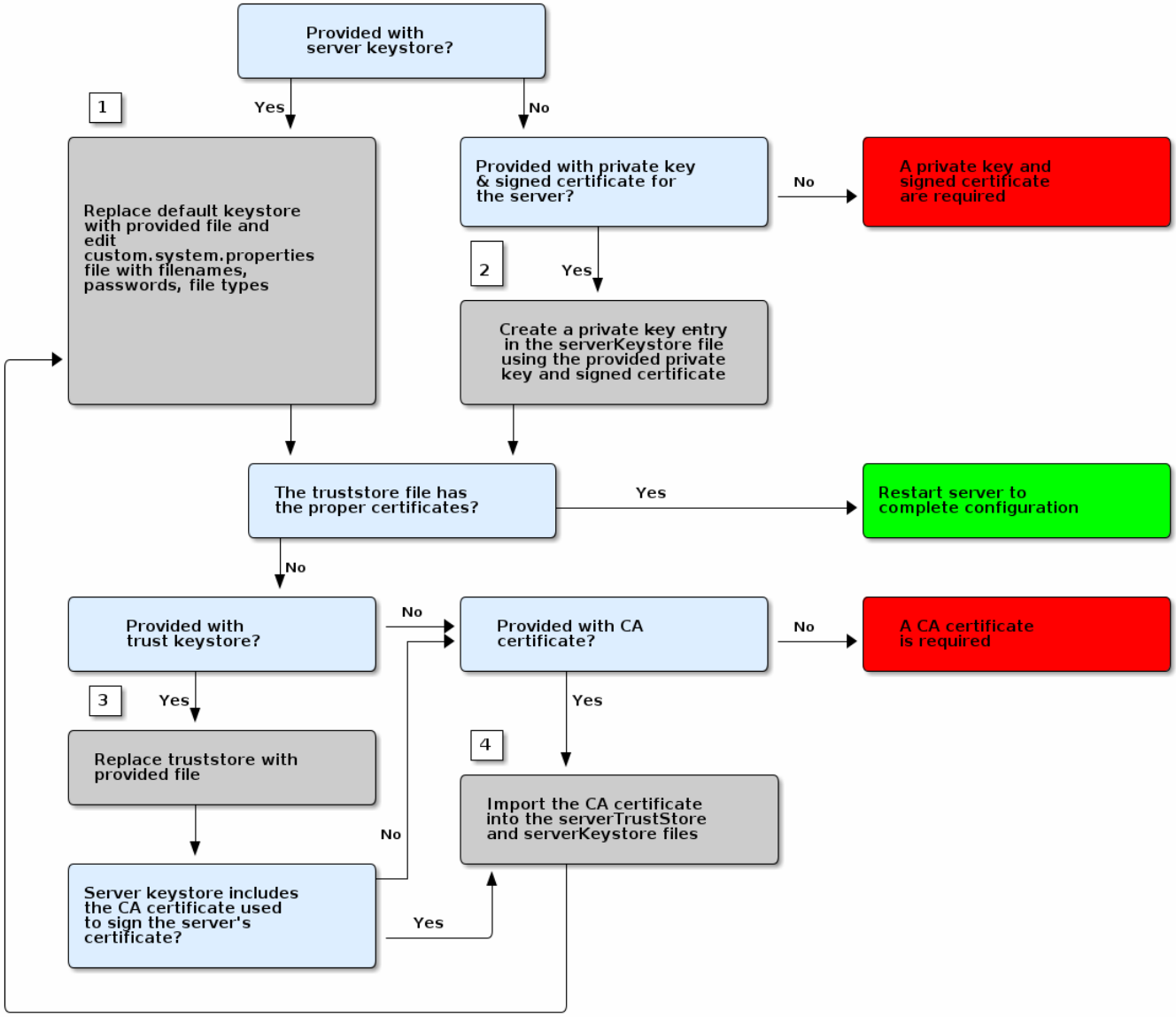


Figure 3. Configuring Certificates Workflow

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

6.2.2.1. 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.

6.2.2.1.1. 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.

6.2.2.1.2. 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.

6.2.2.1.3. 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.

6.2.2.1.4. 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`

6.3. 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.

6.3.1. 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.

6.3.2. 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 11. 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.

6.3.3. Completing Installation

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

6.3.3.1. 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.

6.3.3.2. 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.

6.3.4. 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.

6.3.5. 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 12. 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.

6.4. 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.

6.4.1. High Availability Initial Setup Exceptions

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

6.4.1.1. 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}
```

6.4.1.2. 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.

6.4.1.3. 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.

6.4.1.4. 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.

6.4.1.5. 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.

6.4.1.6. 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.

7. 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.

7.1. 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 13. 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 14. 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**.

7.2. Console Command Reference

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

7.2.1. 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>)
```

7.2.1.1. 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
```

7.3. 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`)

7.3.1. 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 15. 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.

7.3.2. 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.

7.4. 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.

7.4.1. 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.

7.4.1.1. 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.

7.4.1.2. 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**

7.4.1.2.1. 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.

7.4.1.2.2. 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.

7.4.2. 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.

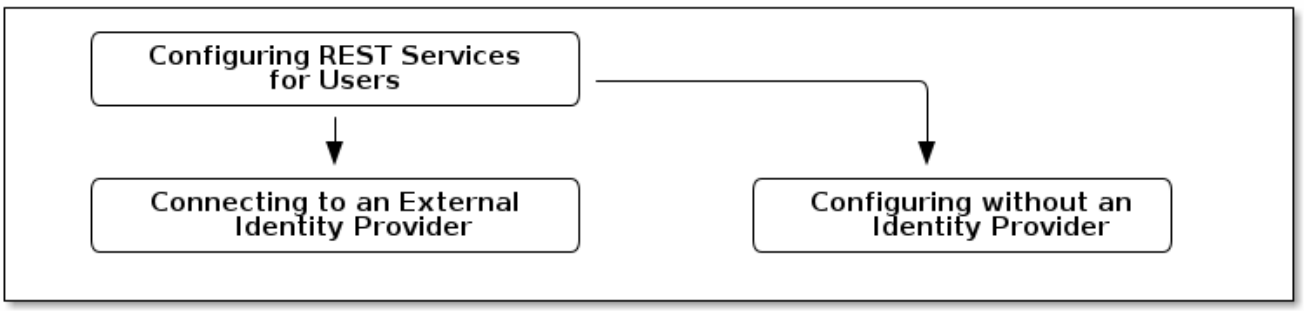


Figure 4. REST Services Configuration Options

7.4.2.1. 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.

7.4.2.1.1. 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.

7.4.2.1.2. 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`).

7.4.2.2. 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)

7.4.2.3. 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.

7.4.2.4. 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.

7.4.3. 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.

7.4.4. 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.

7.4.5. 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.

7.4.5.1. 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.

7.4.6. 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.

7.4.7. 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.

7.4.8. 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.

7.4.9. 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.

7.4.9.1. Managing a Certificate Revocation List (CRL)

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

7.4.9.1.1. 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`

7.4.9.1.2. 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 enableless 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
```

7.4.9.2. 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.

7.4.9.2.1. 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.

7.5. 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.

7.5.1. Configuring Solr

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

Providers.

7.5.1.1. 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`.

7.5.1.1.1. 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.

7.5.1.2. Hardening Solr

Follow instructions on Securing Solr.

7.5.1.2.1. 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.

7.5.1.3. 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.

7.5.2. 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.

7.5.3. 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.

7.5.4. 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.

7.5.4.1. 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.

7.5.4.2. 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.

7.5.4.3. 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.

7.5.5. 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.

7.5.6. 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.

7.5.6.1. Installing the Content Directory Monitor

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

7.5.6.2. 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.

7.5.6.3. 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.

7.5.6.4. 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.

7.5.7. 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.

7.5.8. Configuring Data Policy Plugins

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

7.5.8.1. 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.

7.5.8.2. 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.

7.5.8.3. 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.

7.5.8.4. 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.

7.5.9. Configuring Data Access Plugins

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

7.5.9.1. 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.

7.6. Configuring Security Policies

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

7.6.1. 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.

7.6.1.1. 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.

7.6.1.2. 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.

7.6.1.3. 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 16. 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.

7.6.1.3.1. 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.

7.6.1.4. 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

7.6.1.5. 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.

7.6.2. 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.

7.6.2.1. 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.

7.6.2.2. 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.

7.6.2.3. 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.

7.7. Configuring User Interfaces

DDF has several user interfaces available for users.

7.8. Environment Hardening

- **Required Step for Security Hardening**

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

7.8.1. 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`

7.9. Configuring Federation

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

7.9.1. 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.

7.9.2. 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.

7.9.3. 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.

7.9.3.1. 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.

7.9.3.1.1. 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.

7.9.4. 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.

7.9.4.1. 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.

7.9.4.2. 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.

7.9.4.3. 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.

7.9.4.4. 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 17. 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.

7.9.4.5. 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>
```

7.9.4.6. 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.

7.9.4.7. 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.

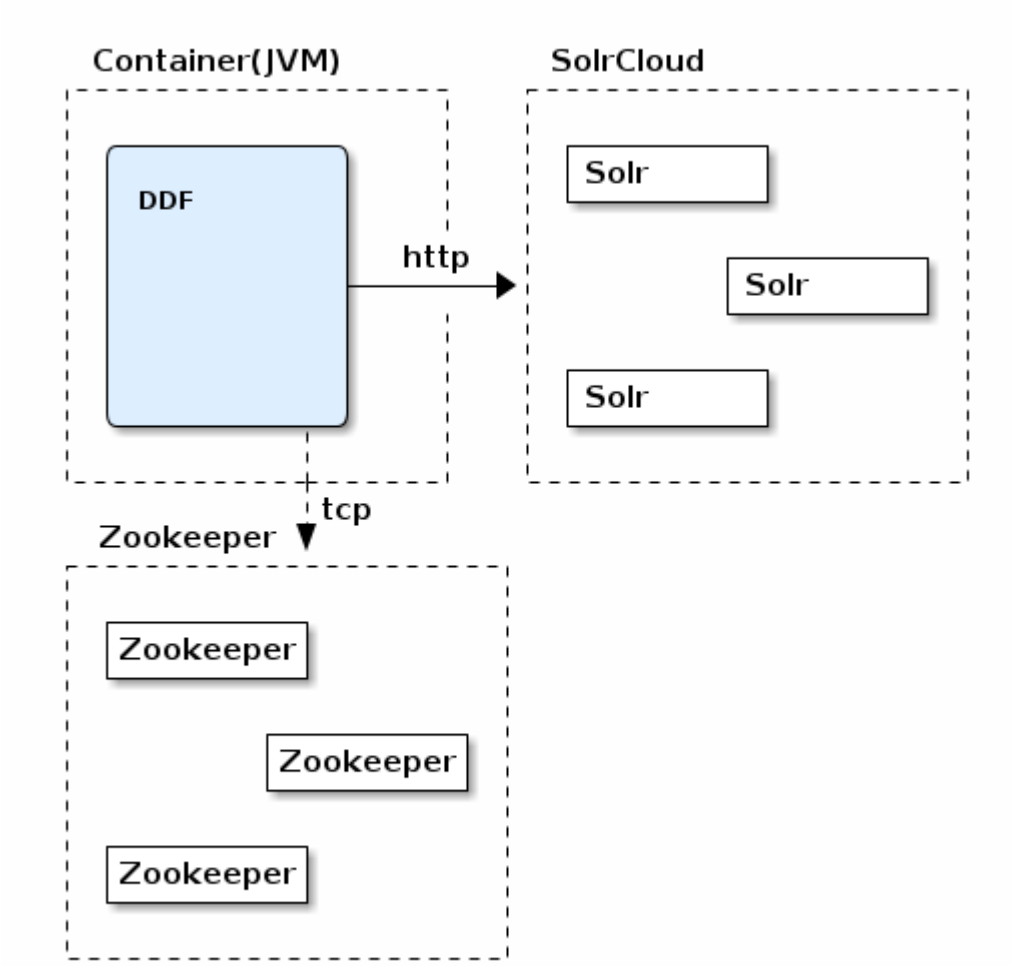


Figure 5. SolrCloud Deployment

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
```

7.9.4.8. 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.

7.9.4.9. 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.

7.9.5. Configuring Endpoints

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

7.9.5.1. 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.

7.9.5.2. 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.

7.9.5.3. 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.

7.9.5.4. 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.

7.10. Configuring for Special Deployments

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

7.10.1. Multiple Installations

One common specialized configuration is installing multiple instances of DDF.

7.10.1.1. 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.

7.10.2. 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.

7.10.3. 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

7.11. Configuring UI Themes

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

7.11.1. Landing Page

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

7.11.1.1. Installing the Landing Page

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

7.11.1.2. 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))`.

7.11.1.3. 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.

7.11.2. 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.

7.11.3. 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.

7.11.3.1. Navigating to UI Theme Configuration

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

7.11.3.2. 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.

7.12. Miscellaneous Configurations

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

7.12.1. 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.

7.12.2. 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.

7.12.3. 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.

7.12.3.1. 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.

7.12.4. Encrypting Passwords

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

7.12.4.1. 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=
   ```

8. 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.

8.1. Starting

8.1.1. 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

8.1.2. Run DDF as a Managed Service

8.1.2.1. 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**.

8.1.2.2. Karaf Documentation

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

8.1.3. 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.

8.1.4. 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
```

8.1.5. 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.

8.1.6. 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.

8.2. Maintaining

8.2.1. 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.

8.2.1.1. 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.

8.2.1.2. 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
```

8.2.1.3. 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 18. 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.

8.2.1.3.1. 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 19. 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
```

8.2.1.3.2. Solr Commands

Table 20. 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.

8.2.1.3.3. Subscriptions Commands

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

Table 21. 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.
```

8.2.1.3.4. Platform Commands

Table 22. Platform Command Descriptions

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

8.2.1.3.5. Persistence Store Commands

Table 23. 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.

8.2.1.4. 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.

8.2.1.4.1. 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.

8.2.1.4.2. 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.

8.2.1.4.3. 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.

8.3. Monitoring

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

8.3.1. 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.

8.3.2. Managing Logging

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

8.3.2.1. Configuring Logging

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

8.3.2.2. 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
```

8.3.2.3. Controlling log level

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

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

8.3.2.4. 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
```

8.3.2.5. 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
```

8.3.2.6. 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
```

8.3.2.7. 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.

8.3.2.8. 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
```

8.3.2.9. 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.

8.4. Troubleshooting

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

Table 24. 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).

8.4.1. 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
```

9. Data Management

9.1. 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.

9.1.1. 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`.

9.1.2. 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`.

9.1.3. 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`.

9.1.4. 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

9.1.5. 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.

9.2. 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).

9.3. 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.

9.3.1. 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.

9.3.1.1. 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`

9.3.2. 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**.

9.3.3. 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.

9.3.4. 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.

9.4. 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.

9.5. Automatically Added Metacard Attributes

This section describes how attributes are automatically added to metacards.

9.5.1. 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.

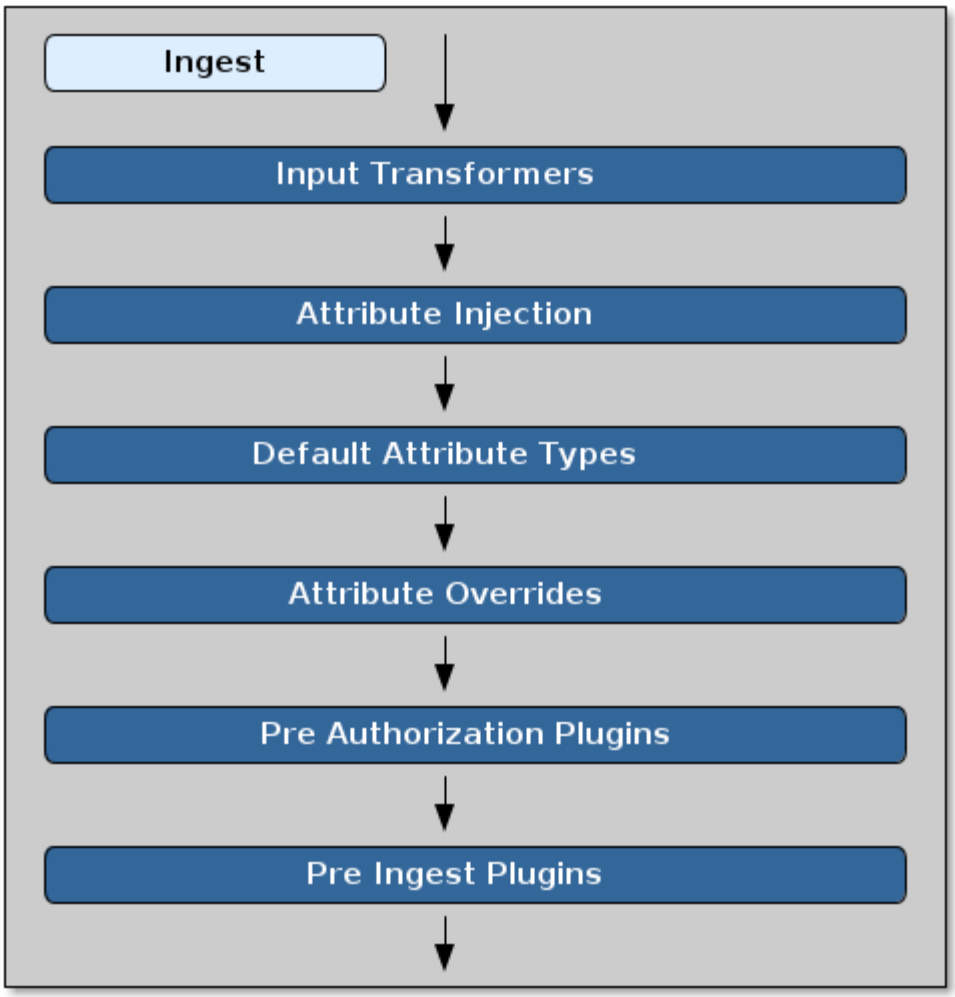


Figure 6. Ingest Attribute Flow

9.5.1.1. 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.

9.5.1.2. 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.

9.5.1.3. 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.

9.5.1.4. 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.

9.5.1.5. 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.

9.5.1.6. 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.

9.5.2. Attributes Added on Query

Metacards resulting from a query undergo Attribute Injection, then have their Attributes Overridden.

9.5.2.1. 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.

Using

10. Using the Simple Search

The DDF Simple Search UI application provides a low-bandwidth option for searching records in the local Catalog (provider) and federated sources. Results are returned in HTML format.

10.1. Search

The **Input** form allows the user to specify keyword, geospatial, temporal, and type query parameters. It also allows the user to select the sources to search and the number of results to return.

10.1.1. Search Criteria

Enter one or more of the available search criteria to execute a query:

Keyword Search

A text box allowing the user to enter a textual query. This supports the use of (*) wildcards. If blank, the query contains a contextual component.

Temporal Query

Select from **any**, **relative**, or **absolute**. Selecting **Any** results in no temporal restrictions on the query, selecting **relative** allows the user to query a period from some length of time in the past until now, and selecting **absolute** allows the user to specify a **start** and **stop** date range.

Spatial Search

Select from **any**, **point-radius**, and **bounding box**. Selecting **Any** results in no spatial restrictions on the query, selecting **point-radius** allows the user to specify a **lat/lon** and **radius** to search, and selecting a **bounding box** allows the user to specify an **eastern**, **western**, **southern**, and **northern** boundary to search within.

Type Search

Select from **any**, or a specific type. Selecting **Any** results in no type restrictions on the query, and Selecting **Specific Types** shows a list of known content types on the federation, and allows the user to select a specific type to search for.

Sources

Select from **none**, **all sources**, or **specific sources**. Selelcting **None** results in querying only the local provider, Selecting **All Sources** results in an enterprise search where all federations are queried, and selecting **Specific Sources** allows the user to select which sources are queried.

Results per Page

Select the number of results to be returned by a single query.

10.1.2. Results

The table of results shows the details of the results found, as well as a link to download the resource if applicable.

10.1.2.1. Results Summary

Total Results

Total Number of Results available for this query. If there are more results than the number displayed per page then a page navigation link appears to the right.

Pages

Provides page navigation, which generate queries for requesting additional pages of results.

10.1.2.2. Results Table

The Results table provides a preview of and links to the results. The table consists of these columns:

Title
: Displays title of the metacard. This is a link which can clicked to view the metacard in the Metacard View.

Source
: Displays where the metadata came from, which could be the local provider or a federated source.

Location
: Displays the WKT Location of the metacard, if available.

Time
: Shows the Received (Created) and Effective times of the metacard, if available.

Thumbnail
: Shows the thumbnail of the metacard, if available.

Download
: A download link to retrieve the resource associated with the metacard, when applicable, if available.

10.1.3. Result View

This view shows more detailed look at a result.

Back to Results Button
: Returns the view back to the Results Table.

Previous & Next
: Navigation to page through the results one by one.

Result Table
: Provides the list of properties and associated values of a single search result.

Metadata
: The metadata, when expanded, displays a tree structure representing the result's custom metadata.

Integrating

DDF is structured to enable flexible integration with external clients and into larger component systems.

If integrating with an existing installation of DDF, continue to the following sections on endpoints and data/metadata management.

If a new installation of DDF is required, first see the Managing section for installation and configuration instructions, then return to this section for guidance on connecting external clients.

If you would like to set up a test or demo installation to use while developing an external client, see the Quick Start Tutorial for demo instructions.

For troubleshooting and known issues, see the Release Notes.

WARNING	If integrating with a Highly Available Cluster of DDF, see High Availability Guidance.

11. Endpoints

with DDF is primarily accomplished through Endpoints accessible through https requests and responses.

NOTE	Not all installations expose all available endpoints. Check with DDF administrator to confirm availability of these endpoints.

11.1. Ingest Endpoints

Ingest is the process of getting data and/or metadata into the DDF catalog framework.

These endpoints are provided by DDF to be used by integrators to ingest content or metadata.

CSW Endpoint
: Searches collections of descriptive information (metadata) about geospatial data and services.

Catalog REST Endpoint
: Uses REST to interact with the catalog.

11.2. CRUD Endpoints

To perform CRUD (Create, Read, Update, Delete) operations on data or metadata in the catalog, work with one of these endpoints.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

Catalog REST Endpoint

Uses REST to interact with the catalog.

11.3. Query Endpoints

Query data or metadata stored within an instance of DDF using one of these endpoints.

Queries Endpoint

To perform CRUD (Create, Read, Update, Delete) operations on query metacards in the catalog, work with one of these endpoints.

OpenSearch Endpoint

Sends query parameters and receives search results.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

11.4. Content Retrieval Endpoints

To retrieve content from an instance of DDF, use one of these endpoints.

Catalog REST Endpoint

Uses REST to interact with the catalog.

11.5. Pub-Sub Endpoints

These endpoints provide publication and subscription services to allow notifications when certain events happen within DDF.

CSW Endpoint

Searches collections of descriptive information (metadata) about geospatial data and services.

11.6. Endpoint Details

11.6.1. Queries Endpoint

The queries endpoint enables an application to create, retrieve, update, and delete query metacards.

Query metacards represent queries within the UI. A query metacard is what is persisted in the data store.

The queries endpoint can be used for one or more of these operations on an instance of DDF:

- Create query metacards and store them in the DDF catalog.
- Retrieve all query metacards stored in the DDF catalog and sort them based on attribute and sort order.
- Retrieve a specific query metacard stored in the DDF catalog.
- Update query metacards that are stored in the DDF catalog.
- Delete query metacards that are stored in the DDF catalog.

Queries Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries
```

11.6.1.1. Queries Endpoint Create Examples

To create a query metacard through the queries endpoint, send a `POST` request to the queries endpoint.

Queries Endpoint Create Request Body

```
{
    "cql":"(\"anyText\" ILIKE 'foo bar')",
    "filterTree":"{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "federation":"enterprise",
    "sorts":[
        {
            "attribute":"modified",
            "direction":"descending"
        }
    ],
    "type":"advanced",
    "title":"Search Title"
}
```

A successful create request returns a status of `201 CREATED`.

Queries Endpoint Create Success Response Body


```
{
    "id": "12bfc601cda449d58733eacaf613b93d",
    "title": "Search Title",
    "created": "Apr 18, 2019 10:20:55 AM",
    "modified": "Apr 18, 2019 10:20:55 AM",
    "owner": "admin@localhost.local",
    "cql": "(\"anyText\" ILIKE 'foo bar')",
    "filterTree": "{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "enterprise": null,
    "sources": [],
    "sorts": [
        {
            "attribute": "modified",
            "direction": "descending"
        }
    ],
    "polling": null,
    "federation": "enterprise",
    "type": "advanced",
    "detailLevel": null,
    "schedules": [],
    "facets": []
}
```


An unsuccessful create request returns a status of `500 SERVER ERROR`.

Queries Endpoint Create Failure Response Body


```
{
    "message": "Something went wrong."
}
```


11.6.1.2. Queries Endpoint Retrieve All Examples

To retrieve a query metacard through the queries endpoint, send a `GET` request to the queries endpoint.

Table 25. Path Parameters

Query Param	Description	Default Value	Valid Values	Type
`start`	`The starting index of the query to receive.`	`1`	`Integer`	`[1, 2^31)`

Query Param	Description	Default Value	Valid Values	Type
count	The number of queries to return.	100	Integer	All integers
attr	The attribute to sort the queries by.	modified	All strings	String
sort_by	The sort order to return the queries in.	desc	asc, desc	String
text	A text field to search against a few attributes.	None	All strings	String

A successful retrieval request returns a status of `200 OK`.

11.6.1.3. Queries Endpoint Retrieve All Fuzzy Examples

To retrieve all query metacards based on some text based value through the queries endpoint, send a `GET` request to the queries endpoint specifying a value for `text` as a query parameters.

Retrieve All Queries Fuzzy Search Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries?text=<VALUE>
```

A fuzzy search is only performed against the `title`, `modified`, `owner`, and `description` attributes.

11.6.1.4. Queries Endpoint Retrieve Examples

Retrieve Specific Query Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries/<ID>
```

To retrieve a specific query metacard through the queries endpoint, send a `GET` request to the queries endpoint with an id.

A successful retrieval request returns a status of `200 OK`.

Query Endpoint Not Found Response Body

```
{
    "message": "Could not find metacard for id: <metacardId>"
}
```

An unsuccessful retrieval request returns a status of `404 NOT FOUND`.

11.6.1.5. Queries Endpoint Update Examples

Update Query Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries/<ID>
```

To update a specific query metacard through the queries endpoint, send a `PUT` request to the queries endpoint with an id.

Update Query Request Request Body

```
{
    "cql":"(\"anyText\" ILIKE 'foo bar')",
    "filterTree":"{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "federation":"enterprise",
    "sorts":[
        {
            "attribute":"modified",
            "direction":"descending"
        }
    ],
    "type":"advanced",
    "title":"New Search Title"
}
```

A successful update request returns a status of `200 OK`.

Update Query Request Response Body

```
{
    "id": "cd6b83db301544e4bb7ece39564261ca",
    "title": "New Search Title",
    "created": "Apr 18, 2019 11:09:35 AM",
    "modified": "Apr 18, 2019 11:09:35 AM",
    "owner": null,
    "cql": "(\"anyText\" ILIKE 'foo barararra')",
    "filterTree": "{\"type\":\"AND\",\"filters\":[{\"type\":\"ILIKE\",\"property\":
\"anyText\",\"value\":\"foo bar\"}]}",
    "enterprise": null,
    "sources": [],
    "sorts": [
        {
            "attribute": "modified",
            "direction": "descending"
        }
    ],
```

```
    "polling": null,
    "federation": "enterprise",
    "type": "advanced",
    "detailLevel": null,
    "schedules": [],
    "facets": []
}
```

An unsuccessful update request returns a status of `404 NOT FOUND`.

Update Query Unsuccessful Response Body

```
{
    "message": "Form is either restricted or not found."
}
```

11.6.1.6. Queries Endpoint Delete Examples

Delete Query Endpoint URL

```
https://<HOSTNAME>:<PORT>/search/catalog/internal/queries/<ID>
```

To delete a specific query metacard through the queries endpoint, send a `GET` request to the queries endpoint with an id.

A successful deletion request returns a status of `204 NO CONTENT`.

An unsuccessful deletion request returns a status of `404 NOT FOUND`.

Delete Query Not Found Response Body

```
{
    "message": "Form is either restricted or not found."
}
```

11.6.2. Prometheus Endpoint

The Prometheus endpoint provides various metrics about DDF such as the number of queries made to a specific source and counters of ingest operations.

An external Prometheus server must be set up and configured to scrape from DDF. The endpoint is exposed at:

Prometheus URL

```
https://<FQDN>:<PORT>/metrics
```

11.6.2.1. Visualising Metrics

A visualization software, such as Grafana, can be connected to Prometheus and configured to display available metrics.

11.6.3. 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.

The OpenSearch specification defines a file format to describe an OpenSearch endpoint. This file is XML-based and is used to programatically retrieve a site's endpoint, as well as the different parameter options a site holds. The parameters are defined via the OpenSearch and CDR IPT Specifications.

11.6.3.1. OpenSearch Contextual Queries

To use the OpenSearch endpoint for a query, send a `GET` request with the query options as parameters

OpenSearch Query URL

```
https://<FQDN>:<PORT>/services/catalog/query?<NAME>="<VALUE>"
```

Table 26. OpenSearch Parameter List

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`searchTerms`	`q`	URL-encoded, space-delimited list of search terms	Complex contextual search string.
`count`	`count`	Integer >= 0	Maximum # of results to retrieve. default: `10`

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`startIndex`	`start`	integer > 0	Index of first result to return. This value uses a one-based index for the results. default: `1`
`format`	`format`	Requires a transformer shortname as a string, possible values include, when available, `atom`, `html`, and `kml`. See Query Response transformers for more possible values.	Defines the format that the return type should be in. default: `atom`

Sample OpenSearch Textual Query

```
https://<FQDN>:<PORT>/services/catalog/query?q="Aliquam"&count=20
```

11.6.3.1.1. Complex OpenSearch Contextual Query Format

The OpenSearch Endpoint supports the following operators: `AND`, `OR`, and `NOT`. These operators are case sensitive. Implicit `AND`s are also supported.

Use parentheses to change the order of operations. Use quotes to group keywords into literal expressions.

See the OpenSearch specification for more syntax specifics.

OpenSearch Endpoint Complex Query Example

```
https://<FQDN>:<PORT>/services/catalog/query?q='cat OR dog'
```

11.6.3.2. OpenSearch Temporal Queries

Queries can also specify a start and end time to narrow results.

Table 27. OpenSearch Temporal Parameters

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`start`	`dtstart`	RFC-3399-defined value: `YYYY-MM-DDTHH:mm:ssZ` or `yyyy-MM-dd␣T␣HH:mm:ss.SSSZZ`	Specifies the beginning of the time slice of the search. Default value of "1970-01-01T00:00:00Z" is used when `dtend` is specified but `dtstart` is not specified.
`end`	`dtend`	RFC-3399-defined value: `YYYY-MM-DDTHH:mm:ssZ` or `yyyy-MM-dd␣T␣HH:mm:ss.SSSZZ`	Specifies the ending of the time slice of the search Current GMT date/time is used when `dtstart` is specified but `dtend` is not specified.

OpenSearch Temporal Query Example

```
https://<FQDN>:<PORT>/services/catalog/query?q='*'&dtstart=2026-04-
24T00:00:00Z&dtend=2026-04-24T18:00:00Z
```

NOTE

The start and end temporal criteria must be of the format specified above. Other formats are currently not supported. Example:

`2026-04-24T12:00:00.111-04:00`.

The start and end temporal elements are based on modified timestamps for a metacard.

11.6.3.3. OpenSearch Geospatial Queries

Query by location.

Use geospatial query parameters to create a geospatial `INTERSECTS` query, where `INTERSECTS` means geometries that are not `DISJOINT` to the given geospatial parameters.

Table 28. Opensearch Geospatial Parameters

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`lat`	`lat`	`EPSG:4326 (WGS84)` decimal degrees	Used in conjunction with the `lon` and `radius` parameters.

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`lon`	`lon`	`EPSG:4326 (WGS84)` decimal degrees	Used in conjunction with the `lat` and `radius` parameters.
`radius`	`radius`	`EPSG:4326 (WGS84)` meters along the Earth's surface > 0	Specifies the search distance in meters from the `lon`,`lat` point. Used in conjunction with the `lat` and `lon` parameters. default: `5000`
`polygon`	`polygon`	Comma-delimited list of lat/lon (`EPSG:4326 (WGS84)` decimal degrees) pairs, in clockwise order around the polygon, where the last point is the same as the first in order to close the polygon. (for example, `-80,-170,0, -170,80, -170,80,170,0,170, -80,170,-80,-170`)	According to the OpenSearch Geo Specification this is **deprecated**. Use the `geometry` parameter instead.
`box`	`bbox`	4 comma-delimited `EPSG:4326 (WGS84)` decimal degrees coordinates in the format West,South,East,North	

OpenSearch Element	HTTPS Parameter	Possible Values	Comments

`geometry`	`geometry`	WKT Geometries
		
		Examples:
		
		`POINT(10 20)` where 10 is the longitude and 20 is the latitude.
		
		`POLYGON ( ( 30 10, 10 20, 20 40, 40 40, 30 10 ) )`. 30 is longitude and 10 is latitude for the first point.
		
		`MULTIPOLYGON ( ( (40 40, 20 45, 45 30, 40 40) ), ( (20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20) ) )`
		
		`GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))`
			Make sure to repeat the starting point as the last point to close the polygon.

OpenSearch GeoSpatial Query Example

```
https://localhost:8993/services/catalog/query?q='*'&lon=44.792&lat=-6.171
```

11.6.3.4. Additional OpenSearch Query Parameters

The OpenSearch Endpoint can also use these additional parameters to refine queries

Table 29. OpenSearch Query Extensions

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
sort	sort	<sbfield>:<sborder> where <sbfield> is date or relevance <sborder> is asc or desc	<sborder> is optional but has a value of asc or desc (default is desc). However, when <sbfield> is relevance, <sborder> must be desc. Sorting by date sorts the results by the effective date. default: relevance:desc
maxResults	mr	Integer >= 0	Maximum # of results to return. If count is also specified, the count value takes precedence over the maxResults value. default: 1000
maxTimeout	mt	Integer > 0	Maximum timeout (milliseconds) for query to respond. default: 300000 (5 minutes)
dateOffset	dtoffset	Integer > 0	Specifies an offset (milliseconds), backwards from the current time, to search on the modified time field for entries.
type	type	Any valid datatype (such as Text)	Specifies the type of data to search for.
version	version	Comma-delimited list of strings (such as 20,30)	Version values for which to search.

Table 30. Federated Search

OpenSearch Element	HTTPS Parameter	Possible Values	Comments
`routeTo`	`src`	Comma-delimited list of site names to query. Varies depending on the names of the sites in the federation. `local` specifies to query the local site.	If `src` is not provided, the default behavior is to execute an enterprise search to the entire federation.

11.6.4. CSW Endpoint

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

The CSW endpoint supports metadata operations only.

For more information about the Catalogue Services for Web (CSW) standard.

The CSW Endpoint can be used for one or more of these operations on an instance of DDF:

- Ingest metadata into the DDF catalog.
- Read metacards from the catalog.
- Update metadata in the catalog.
- Publish and/or subscribe to catalog events.
- Delete metadata from the catalog.
- Get capabilities of the catalog and the URLs used to access.

NOTE

Sample Responses May Not Match Actual Responses

Actual responses may vary from these samples, depending on your configuration. Send a GET or POST request to obtain an accurate response.

11.6.4.1. CSW Endpoint Create Examples

Metacards are ingested into the catalog via the `Insert` sub-operation.

The schema of the record needs to conform to a schema of the information model that the catalog supports.

Send a `POST` request to the CSW endpoint URL.

CSW Endpoint Ingest URL

```
https://<FQDN>:<PORT>/services/csw
```

Include the metadata to ingest within a `csw:Insert` block in the body of the request.

Sample XML Transaction `Insert` Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    verboseResponse="true"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Insert typeName="csw:Record">
        <csw:Record
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dct="http://purl.org/dc/terms/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <dc:identifier></dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
            <dc:subject>Hydrography--Dictionaries</dc:subject>
            <dc:format>application/pdf</dc:format>
            <dc:date>2026-04-24</dc:date>
            <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.</dct:abstract>
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
                <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
                <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:Record>
    </csw:Insert>
</csw:Transaction>
```

To specify the document type being ingested and select the appropriate input transformer, use the `typeName` attribute in the `csw:Insert` element

```
<csw:Insert typeName="xml">
```

To receive a copy of the metacard in the response, specify `verboseResponse="true"` in the `csw:Transaction`. The `InsertResult` element of the response holds the metacard information added to the catalog.

```
<csw:Transaction service="CSW" version="2.0.2" verboseResponse="true" [...]
```

Sample XML Transformer Insert

```
<csw:Transaction service="CSW" version="2.0.2" verboseResponse="true" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2">
  <csw:Insert typeName="xml">
    <metacard xmlns="urn:catalog:metacard" xmlns:ns2="http://www.opengis.net/gml"
          xmlns:ns3="http://www.w3.org/1999/xlink" xmlns:ns4=
"http://www.w3.org/2001/SMIL20/"
          xmlns:ns5="http://www.w3.org/2001/SMIL20/Language">
        <type>ddf.metacard</type>
        <string name="title">
            <value>PlainXml near</value>
        </string>
    </metacard>
  </csw:Insert>
</csw:Transaction>
```

A successful ingest returns a status of `200 OK` and `csw:TransactionResponse`.

Sample XML Transaction `Insert` Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:ns3="http://www.w3.org/1999/xlink"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:ns5="http://www.w3.org/2001/SMIL20/"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:ows="http://www.opengis.net/ows"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                         version="2.0.2"
                         ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:TransactionSummary>
        <csw:totalInserted>1</csw:totalInserted>
        <csw:totalUpdated>0</csw:totalUpdated>
        <csw:totalDeleted>0</csw:totalDeleted>
    </csw:TransactionSummary>
    <csw:InsertResult>
        <csw:BriefRecord>
            <dc:identifier><METACARD ID</dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
            <ows:BoundingBox crs="EPSG:4326">
                <ows:LowerCorner>-6.171 44.792</ows:LowerCorner>
```

```
                <ows:UpperCorner>-2.228 51.126</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:BriefRecord>
    </csw:InsertResult>
</csw:TransactionResponse>
```

Table 31. Create Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	`ExceptionText` with description of error.	XML error. Check for formatting errors in record.
		Schema error. Verify metadata is compliant with defined schema.

11.6.4.2. CSW Endpoint Query Examples

To query through the CSW Endpoint, send a `POST` request to the CSW endpoint.

CSW Endpoint Query URL

```
https://<FQDN>:<PORT>/services/csw
```

Within the body of the request, include a `GetRecords` operation to define the query. Define the service and version to use (CSW, 2.0.2). The output format must be `application/xml`. Specify the output schema. (To get a list of supported schemas, send a Get Capabilities request to the CSW endpoint.)

GetRecords Syntax

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
```

Include the query within the `GetRecords` request. Optionally, set the `ElementSetName` to determine how much detail to return.

- Brief: the least possible detail.

- Summary: (Default)
- Full: All metadata elements for the records.

Within the `Constraint` element, define the query as an OSG or CQL filter.

```
<Query typeNames="Record">
    <ElementSetName>summary</ElementSetName>
    <Constraint version="1.1.0">
        <ogc:Filter>
            <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                <ogc:PropertyName>AnyText</ogc:PropertyName>
                <ogc:Literal>%</ogc:Literal>
            </ogc:PropertyIsLike>
        </ogc:Filter>
    </Constraint>
</Query>
```

```
<Query typeNames="Record">
    <ElementSetName>summary</ElementSetName>
    <Constraint version="2.0.0">
        <ogc:CqlText>
           "AnyText" = '%'
        </ogc:CqlText>
    </csw:Constraint>
</Query>
```

`GetRecords` *XML Request Example*

```
<?xml version="1.0" ?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <Query typeNames="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
```

```
                <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                    <ogc:PropertyName>AnyText</ogc:PropertyName>
                    <ogc:Literal>%</ogc:Literal>
                </ogc:PropertyIsLike>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

`GetRecords` *Sample Response (*`application/xml`*)*

```
<?xml version='1.0' encoding='UTF-8'?>
<csw:GetRecordsResponse xmlns:dct="http://purl.org/dc/terms/"
                        xmlns:xml="http://www.w3.org/XML/1998/namespace"
                        xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                        xmlns:ows="http://www.opengis.net/ows"
                        xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0.2">
  <csw:SearchStatus timestamp="2026-04-24"/>
  <csw:SearchResults numberOfRecordsMatched="1" numberOfRecordsReturned="1" nextRecord="
0" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">
    <csw:Record xmlns:ows="http://www.opengis.net/ows"
                xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                xmlns:dc="http://purl.org/dc/elements/1.1/"
                xmlns:dct="http://purl.org/dc/terms/"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <dc:identifier/>
        <dc:title>Aliquam fermentum purus quis arcu</dc:title>
        <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
        <dc:subject>Hydrography--Dictionaries</dc:subject>
        <dc:format>application/pdf</dc:format>
        <dc:date>2026-04-24</dc:date>
        <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.</dct:abstract>
        <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
            <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
            <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
        </ows:BoundingBox>
    </csw:Record>
  </csw:SearchResults>
</csw:GetRecordsResponse>
```

Querying a Specific Source with the CSW Endpoint

To query a `Specific Source`, specify a query for a `source-id`. To find a valid `source-id` , send a Get Capabilities request. Configured sources are listed in the `FederatedCatalogs` section of the response.

NOTE The `DistributedSearch` element must be specific with a `hopCount` greater than 1 to identify it as a federated query, otherwise the `source-id`'s is ignored.

Querying a Specific Source Sample Request

```
<?xml version="1.0" ?>
<csw:GetRecords resultType="results"
    outputFormat="application/xml"
    outputSchema="urn:catalog:metacard"
    startPosition="1"
    maxRecords="10"
    service="CSW"
    version="2.0.2"
    xmlns:ns2="http://www.opengis.net/ogc" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:ns4="http://www.w3.org/1999/xlink"
xmlns:ns3="http://www.opengis.net/gml" xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
xmlns:ns5="http://www.opengis.net/ows" xmlns:ns6="http://purl.org/dc/elements/1.1/"
xmlns:ns7="http://purl.org/dc/terms/" xmlns:ns8="http://www.w3.org/2001/SMIL20/">
  <csw:DistributedSearch hopCount="2" />
    <ns10:Query typeNames="csw:Record" xmlns="" xmlns:ns10=
"http://www.opengis.net/cat/csw/2.0.2">
        <ns10:ElementSetName>full</ns10:ElementSetName>
        <ns10:Constraint version="1.1.0">
            <ns2:Filter>
              <ns2:And>
                <ns2:PropertyIsEqualTo wildCard="*" singleChar="#" escapeChar="!">
                  <ns2:PropertyName>source-id</ns2:PropertyName>
                  <ns2:Literal>Source1</ns2:Literal>
                </ns2:PropertyIsEqualTo>_
                <ns2:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!">
                  <ns2:PropertyName>title</ns2:PropertyName>
                    <ns2:Literal>*</ns2:Literal>
                </ns2:PropertyIsLike>
              </ns2:And>
            </ns2:Filter>
        </ns10:Constraint>
    </ns10:Query>
</csw:GetRecords>
```

Querying for GMD Output Schema

To receive a response to a `GetRecords` query that conforms to the GMD specification, set the Namespace(xmlns),`outputschema`, and `typeName` elements for GML schema.

GML Output Schema Sample Request

```
<?xml version="1.0" ?>
```

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:gmd="http://www.isotc211.org/2005/gmd"
        xmlns:gml="http://www.opengis.net/gml"
        service="CSW"
        version="2.0.2"
        maxRecords="8"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.isotc211.org/2005/gmd"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <Query typeNames="gmd:MD_Metadata">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                    <ogc:PropertyName>apiso:Title</ogc:PropertyName>
                    <ogc:Literal>%</ogc:Literal>
                </ogc:PropertyIsLike>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

Querying by UTM Coordinates

UTM coordinates can be used when making a CSW GetRecords request using an `ogc:Filter`. UTM coordinates should use `EPSG:326XX` as the `srsName` where `XX` is the zone within the northern hemisphere. UTM coordinates should use `EPSG:327XX` as the `srsName` where `XX` is the zone within the southern hemisphere.

NOTE UTM coordinates are only supported with requests providing an `ogc:Filter`, but not with CQL as there isn't a way to specify the UTM `srsName` in CQL.

UTM Northern Hemisphere Zone 36 Sample Request

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:gml="http://www.opengis.net/gml"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
```

```
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <Query typeNames="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:Intersects>
                    <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                    <gml:Envelope srsName="EPSG:32636">
                        <gml:lowerCorner>171070 1106907</gml:lowerCorner>
                        <gml:upperCorner>225928 1106910</gml:upperCorner>
                    </gml:Envelope>
                </ogc:Intersects>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

Querying by Metacard ID

To locate a record by Metacard ID, send a `POST` request with a `GetRecordById` element specifying the ID.

`GetRecordById` *Request Example*

```
<GetRecordById xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  service="CSW"
  version="2.0.2"
  outputFormat="application/xml"
  outputSchema="http://www.opengis.net/cat/csw/2.0.2"
  xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
../../../csw/2.0.2/CSW-discovery.xsd">
 <ElementSetName>full</ElementSetName>
 <Id>_<METACARD-ID>_</Id>
</GetRecordById>
```

Table 32. CSW Record to Metacard Mapping

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dc:title	title	1-n	1-n	0-n
dc:creator				0-n

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dc:subject			0-n	0-n
dc:description				0-n
dc:publisher				0-n □□□□□□□□□□
dc:contributor				0-n
dc:date	modified			0-n
dc:type	metadata-content-type	0-1	0-1	0-n
dc:format			0-n	0-n
dc:identifier	id	1-n	1-n	0-n
dc:source	source-id			0-n
dc:language				0-n
dc:relation			0-n	0-n
dc:coverage				0-n
dc:rights				0-n
dct:abstract	description		0-n	0-n
dct:accessRights				0-n
dct:alternative	title			0-n
dct:audience				0-n
dct:available				0-n
dct:bibliographicCitation	id			0-n
dct:conformsTo				0-n
dct:created	created			0-n
dct:dateAccepted	effective			0-n
dct:Copyrighted	effective			0-n
dct:dateSubmitted	modified			0-n □
dct:educationLevel				0-n □
dct:extent				0-n
dct:hasFormat				0-n
dct:hasPart				0-n

CSW Record Field	Metacard Field	Brief Record	Summary Record	Record
dct:hasVersion				0-n
dct:isFormatOf				0-n
dct:isPartOf				0-n
dct:isReferencedBy				0-n
dct:isReplacedBy				0-n
dct:isRequiredBy				0-n
dct:issued	modified			0-n
dct:isVersionOf				0-n
dct:license				0-n
dct:mediator				0-n
dct:medium				0-n
dct:modified	modified		0-n	0-n
dct:provenance				0-n
dct:references				0-n
dct:replaces				0-n
dct:requires				0-n
dct:rightsHolder				0-n
dct:spatial	location		0-n	0-n
dct:tableOfContents				0-n
dct:temporal	effective + " - " + expiration			0-n
dct:valid	expiration			0-n
ows:BoundingBox		0-n	0-n	0-n

Table 33. Query Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	<ows:ExceptionText>ddf.catalog.util.impl.CatalogQueryException: ddf.catalog.federation.FederationException: SiteNames could not be resolved due to invalid site names, none of the sites were available, or the current subject doesn't have permission to access the sites.</ows:ExceptionText>	A query to a specific source has specified a source that is unavailable.
200 OK	<csw:SearchResults numberOfRecordsMatched="0" numberOfRecordsReturned="0" nextRecord="0"	No results found for query. Verify input.

11.6.4.3. CSW Endpoint Update Examples

The CSW Endpoint can edit the metadata attributes of a metacard.

Send a POST request to the CSW Endpoint URL:

CSW Endpoint Update URL

```
https://<FDQN>:<PORT>/services/csw
```

Replace the <METACARD-ID> value with the metacard id being updated, and edit any properties within the csw:Record.

CSW Update Record Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
    <csw:Update>
        <csw:Record
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dct="http://purl.org/dc/terms/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <dc:identifier><METACARD-ID></dc:identifier>
            <dc:title>Aliquam fermentum purus quis arcu</dc:title>
            <dc:type>http://purl.org/dc/dcmitype/Text</dc:type>
```

```
            <dc:subject>Hydrography--Dictionaries</dc:subject>
            <dc:format>application/pdf</dc:format>
            <dc:date>2026-04-24</dc:date>
            <dct:abstract>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.</dct:abstract>
            <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
                <ows:LowerCorner>44.792 -6.171</ows:LowerCorner>
                <ows:UpperCorner>51.126 -2.228</ows:UpperCorner>
            </ows:BoundingBox>
        </csw:Record>
    </csw:Update>
</csw:Transaction>
```

CSW Update Record Sample Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ows="http://www.opengis.net/ows"
                         xmlns:ns2="http://www.w3.org/1999/xlink"
                         xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:ns6="http://www.w3.org/2001/SMIL20/"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance" version=
"2.0.2"
                         ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:TransactionSummary>
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>1</csw:totalUpdated>
        <csw:totalDeleted>0</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

Updating Individual Attributes

Within the `csw:Transaction` element, use the `csw:RecordProperty` to update individual metacard attributes.

Use the `Name` element to specify the name of the record property to be updated and set the `Value` element to the value to update in the record. The values in the `Update` completely replace those that are already in the record.

```
<csw:RecordProperty>
```

```
    <csw:Name>title</csw:Name>
    <csw:Value>Updated Title</csw:Value>
</csw:RecordProperty>
```

Removing Attributes

To remove a non-required attribute, send the `csw:Name` without a `csw:Value`.

```
<csw:RecordProperty>
    <csw:Name>title</csw:Name>
</csw:RecordProperty>
```

Required attributes are set to a default value if no `Value` element is provided.

Table 34. `RecordProperty` Default Values

Property	Default Value
`metadata-content-type`	Resource
`created`	*current time*
`modified`	*current time*
`effective`	*current time*
`metadata-content-type-version`	*myVersion*
`metacard.created`	*current time*
`metacard.modified`	*current time*
`metacard-tags`	resource, VALID
`point-of-contact`	system@localhost
`title`	*current time*

Use a `csw:Constraint` to specify the metacard ID. The constraint can be an OGC Filter or a CQL query.

```
<csw:Constraint version="2.0.0">
    <ogc:Filter>
        <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>id</ogc:PropertyName>
            <ogc:Literal><METACARD-ID></ogc:Literal>
        </ogc:PropertyIsEqualTo>
    </ogc:Filter>
</csw:Constraint>
```

```
<csw:Constraint version="2.0.0">
```

```
    <ogc:CqlText>
        "id" = '<METACARD-ID>'
    </ogc:CqlText>
</csw:Constraint>
```

WARNING	These filters can search on any arbitrary query criteria, but take care to only affect desired records.

Sample XML Transaction `Update` Request with OGC filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Update>
        <csw:RecordProperty>
            <csw:Name>title</csw:Name>
            <csw:Value>Updated Title</csw:Value>
        </csw:RecordProperty>
            <csw:Constraint version="2.0.0">
                <ogc:Filter>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>id</ogc:PropertyName>
                        <ogc:Literal><METACARD-ID></ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                </ogc:Filter>
            </csw:Constraint>
    </csw:Update>
</csw:Transaction>
```

Sample XML Transaction `Update` Request with CQL filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction
    service="CSW"
    version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Update>
        <csw:RecordProperty>
            <csw:Name>title</csw:Name>
            <csw:Value>Updated Title</csw:Value>
        </csw:RecordProperty>
        <csw:RecordProperty>
```

```
        </csw:RecordProperty>
        <csw:Constraint version="2.0.0">
            <ogc:CqlText>
                "id" = '<METACARD-ID>'
            </ogc:CqlText>
        </csw:Constraint>
    </csw:Update>
</csw:Transaction>
```

Sample XML Transaction Update Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:ns3="http://www.w3.org/1999/xlink"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:ns5="http://www.w3.org/2001/SMIL20/"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:ows="http://www.opengis.net/ows"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                         ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd"
                         version="2.0.2">
    <csw:TransactionSummary>
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>1</csw:totalUpdated>
        <csw:totalDeleted>0</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

Table 35. Update Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	`<ows:ExceptionText>Unable to update record(s).</ows:ExceptionText>`	XML or CSW schema error. Verify input.
200 OK	`<csw:totalUpdated>0</csw:totalUpdated>`	No records were updated. Verify metacard id or search parameters.

11.6.4.4. CSW Endpoint Publication / Subscription Examples

The subscription `GetRecords` operation is very similar to the `GetRecords` operation used to search the catalog but it subscribes to a search and sends events to a `ResponseHandler` endpoint as metacards are

ingested matching the `GetRecords` request used in the subscription. The `ResponseHandler` must use the https protocol and receive a HEAD request to poll for availability and POST/PUT/DELETE requests for creation, updates, and deletions. The response to a `GetRecords` request on the subscription url is an acknowledgement containing the original `GetRecords` request and a `requestId`. The client is assigned a `requestId` (URN).

A Subscription listens for events from federated sources if the `DistributedSearch` element is present and the catalog is a member of a federation.

Adding a Subscription

Send a `POST` request to the CSW endpoint.

CSW Add Subscription Sample URL

```
https://<FQDN>:<PORT>/services/csw/subscription
```

Subscription `GetRecords` XML Request

```
<?xml version="1.0" ?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        service="CSW"
        version="2.0.2"
        maxRecords="4"
        startPosition="1"
        resultType="results"
        outputFormat="application/xml"
        outputSchema="http://www.opengis.net/cat/csw/2.0.2"
        xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
    <ResponseHandler>https://some.ddf/services/csw/subscription/event</ResponseHandler>
    <Query typeNames="Record">
        <ElementSetName>summary</ElementSetName>
        <Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                    <ogc:PropertyName>xml</ogc:PropertyName>
                    <ogc:Literal>%</ogc:Literal>
                </ogc:PropertyIsLike>
            </ogc:Filter>
        </Constraint>
    </Query>
</GetRecords>
```

Updating a Subscription

To update an existing subscription, send a `PUT` request with the `requestid` URN appended to the url.

CSW Endpoint Subscription Update URL

```
https://{FQDN}:{PORT}/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

Subscription `GetRecords` XML Response

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2026-04-24T18:49:45" xmlns=
"http://www.opengis.net/cat/csw/2.0.2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-
discovery.xsd">
  <EchoedRequest>
    <GetRecords
            requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
            service="CSW"
            version="2.0.2"
            maxRecords="4"
            startPosition="1"
            resultType="results"
            outputFormat="application/xml"
            outputSchema="urn:catalog:metacard">
        <ResponseHandler>
https://some.ddf/services/csw/subscription/event</ResponseHandler>
        <Query typeNames="Record">
            <ElementSetName>summary</ElementSetName>
            <Constraint version="1.1.0">
                <ogc:Filter>
                    <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                        <ogc:PropertyName>xml</ogc:PropertyName>
                        <ogc:Literal>%</ogc:Literal>
                    </ogc:PropertyIsLike>
                </ogc:Filter>
            </Constraint>
        </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

Subscription `GetRecords` Event Sample Response

```
<csw:GetRecordsResponse version="2.0.2" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows" xmlns:xs=
"http://www.w3.org/2001/XMLSchema"  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <csw:SearchStatus timestamp="2014-02-19T15:33:44.602-05:00"/>
    <csw:SearchResults numberOfRecordsMatched="1" numberOfRecordsReturned="1" nextRecord
="5" recordSchema="http://www.opengis.net/cat/csw/2.0.2" elementSet="summary">
      <csw:SummaryRecord>
        <dc:identifier>f45415884c11409497e22db8303fe8c6</dc:identifier>
        <dc:title>Product10</dc:title>
        <dc:type>pdf</dc:type>
        <dct:modified>2014-02-19T15:22:51.563-05:00</dct:modified>
        <ows:BoundingBox crs="urn:x-ogc:def:crs:EPSG:6.11:4326">
          <ows:LowerCorner>20.0 10.0</ows:LowerCorner>
          <ows:UpperCorner>20.0 10.0</ows:UpperCorner>
        </ows:BoundingBox>
      </csw:SummaryRecord>
    </csw:SearchResults>
  </csw:GetRecordsResponse>
```

Retrieving an Active Subscription

To retrieve an active subscription, send a `GET` request with the `requestid` URN appended to the url.

Retrieve.

```
https://<FQDN>:<PORT>/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

Subscription `HTTP GET` Sample Response

```
<?xml version="1.0" ?>
<Acknowledgement timeStamp="2026-04-24T18:49:45" xmlns=
"http://www.opengis.net/cat/csw/2.0.2"
                                                 xmlns:ogc="http://www.opengis.net/ogc"
                                                 xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
                                                 xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 ../../../csw/2.0.2/CSW-discovery.xsd">
  <EchoedRequest>
    <GetRecords
            requestId="urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f"
            service="CSW"
            version="2.0.2"
            maxRecords="4"
```

```
            startPosition="1"
            resultType="results"
            outputFormat="application/xml"
            outputSchema="urn:catalog:metacard">
        <ResponseHandler>
https://some.ddf/services/csw/subscription/event</ResponseHandler>
        <Query typeNames="Record">
            <ElementSetName>summary</ElementSetName>
            <Constraint version="1.1.0">
                <ogc:Filter>
                    <ogc:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
                        <ogc:PropertyName>xml</ogc:PropertyName>
                        <ogc:Literal>%</ogc:Literal>
                    </ogc:PropertyIsLike>
                </ogc:Filter>
            </Constraint>
        </Query>
    </GetRecords>
  </EchoedRequest>
  <RequestId>urn:uuid:4d5a5249-be03-4fe8-afea-6115021dd62f</ns:RequestId>
</Acknowledgement>
```

Deleting a Subscription

To delete a subscription, send a `DELETE` request with the `requestid` URN appended to the url.

Delete Subscription Sample URL

```
https://<FQDN>:<PORT>/services/csw/subscription/urn:uuid:4d5a5249-be03-4fe8-afea-
6115021dd62f
```

11.6.4.5. CSW Endpoint Delete Examples

To delete metacards via the CSW Endpoint, send a `POST` request with a `csw:Delete` to the CSW Endpoint URL.

```
https://<FQDN>:<PORT>/services/csw
```

Define the records to delete with the `csw:Constraint` field. The constraint can be either an OGC or CQL filter.

Sample XML Transaction `Delete` Request with OGC filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
```

```
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Delete typeName="csw:Record" handle="something">
        <csw:Constraint version="2.0.0">
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                   <ogc:PropertyName>id</ogc:PropertyName>
                   <ogc:Literal>_<METACARD-ID>_</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
        </csw:Constraint>
    </csw:Delete>
</csw:Transaction>
```

Sample XML Transaction `Delete` Request with CQL filter constraint

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:Transaction service="CSW" version="2.0.2"
    xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc">
    <csw:Delete typeName="csw:Record" handle="something">
        <csw:Constraint version="2.0.0">
            <ogc:CqlText>
                 "id" = '<METACARD-ID>'
            </ogc:CqlText>
        </csw:Constraint>
    </csw:Delete>
</csw:Transaction>
```

Sample XML Transaction Delete Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<csw:TransactionResponse xmlns:ows="http://www.opengis.net/ows"
                         xmlns:ns2="http://www.w3.org/1999/xlink"
                         xmlns:ogc="http://www.opengis.net/ogc"
                         xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:dct="http://purl.org/dc/terms/"
                         xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:ns8="http://www.w3.org/2001/SMIL20/"
                         xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
                         xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance"
                         version="2.0.2" ns10:schemaLocation="http://www.opengis.net/csw
/ogc/csw/2.0.2/CSW-publication.xsd">
    <csw:TransactionSummary>
```

```
        <csw:totalInserted>0</csw:totalInserted>
        <csw:totalUpdated>0</csw:totalUpdated>
        <csw:totalDeleted>1</csw:totalDeleted>
    </csw:TransactionSummary>
</csw:TransactionResponse>
```

Table 36. Delete Error Response Examples

Status Code	Error Message	Possible Causes
`200 OK`	`<csw:totalDeleted>0</csw:totalDeleted>`	No records matched filter criteria. Verify metacard ID.
`400 Bad Request`	`<ows:Exception>` with details of error.	XML or CSW formatting error. Verify request.

11.6.4.6. CSW Endpoint Get Capabilities Examples

The `GetCapabilities` operation describes the operations the catalog supports and the URLs used to access those operations. The CSW endpoint supports both `HTTP GET` and `HTTP POST` requests for the `GetCapabilities` operation. The response to either request is always a `csw:Capabilities` XML document. This XML document is defined by the CSW-Discovery XML Schema.

CSW Endpoint `GetCapabilities` URL for GET request

```
https://<FQDN>:<PORT>/services/csw?service=CSW&version=2.0.2&request=GetCapabilities
```

Alternatively, send a `POST` request to the root CSW endpoint URL.

CSW Endpoint `GetCapabilities` URL for GET request

```
$https://<FQDN>:<PORT>/services/csw
```

Include an XML message body with a `GetCapabilities` element.

`GetCapabilities` *Sample Request*

```
<?xml version="1.0" ?>
<csw:GetCapabilities
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
  service="CSW"
  version="2.0.2" >
</csw:GetCapabilities>
```

`GetCapabilities` *Sample Response (`application/xml`)*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<csw:Capabilities xmlns:ows="http://www.opengis.net/ows" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml=
"http://www.opengis.net/gml" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:ns6=
"http://www.w3.org/2001/SMIL20/" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dct=
"http://purl.org/dc/terms/" xmlns:ns9="http://www.w3.org/2001/SMIL20/Language"
xmlns:ns10="http://www.w3.org/2001/XMLSchema-instance" version="2.0.2"
ns10:schemaLocation="http://www.opengis.net/csw /ogc/csw/2.0.2/CSW-publication.xsd">
    <ows:ServiceIdentification>
        <ows:Title>Catalog Service for the Web</ows:Title>
        <ows:Abstract>DDF CSW Endpoint</ows:Abstract>
        <ows:ServiceType>CSW</ows:ServiceType>
        <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>DDF</ows:ProviderName>
        <ows:ProviderSite/>
        <ows:ServiceContact/>
    </ows:ServiceProvider>
    <ows:OperationsMetadata>
        <ows:Operation name="GetCapabilities">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="sections">
                <ows:Value>ServiceIdentification</ows:Value>
                <ows:Value>ServiceProvider</ows:Value>
                <ows:Value>OperationsMetadata</ows:Value>
                <ows:Value>Filter_Capabilities</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Operation name="DescribeRecord">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
```

```
            <ows:Parameter name="typeName">
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputFormat">
                <ows:Value>application/xml</ows:Value>
                <ows:Value>application/json</ows:Value>
                <ows:Value>application/atom+xml</ows:Value>
                <ows:Value>text/xml</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="schemaLanguage">
                <ows:Value>http://www.w3.org/XMLSchema</ows:Value>
                <ows:Value>http://www.w3.org/XML/Schema</ows:Value>
                <ows:Value>http://www.w3.org/2001/XMLSchema</ows:Value>
                <ows:Value>http://www.w3.org/TR/xmlschema-1/</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Operation name="GetRecords">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="ResultType">
                <ows:Value>hits</ows:Value>
                <ows:Value>results</ows:Value>
                <ows:Value>validate</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputFormat">
                <ows:Value>application/xml</ows:Value>
                <ows:Value>application/json</ows:Value>
                <ows:Value>application/atom+xml</ows:Value>
                <ows:Value>text/xml</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputSchema">
                <ows:Value>urn:catalog:metacard</ows:Value>
                <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
                <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="typeNames">
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
            </ows:Parameter>
```

```
            <ows:Parameter name="ConstraintLanguage">
                <ows:Value>Filter</ows:Value>
                <ows:Value>CQL_Text</ows:Value>
            </ows:Parameter>
            <ows:Constraint name="FederatedCatalogs">
                <ows:Value>Source1</ows:Value>
                <ows:Value>Source2</ows:Value>
            </ows:Constraint>
        </ows:Operation>
        <ows:Operation name="GetRecordById">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get ns2:href="https://<FQDN>:<PORT>/services/csw"/>
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="OutputSchema">
                <ows:Value>urn:catalog:metacard</ows:Value>
                <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
                <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
                <ows:Value>http://www.iana.org/assignments/media-types/application/octet-
stream</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="OutputFormat">
                <ows:Value>application/xml</ows:Value>
                <ows:Value>application/json</ows:Value>
                <ows:Value>application/atom+xml</ows:Value>
                <ows:Value>text/xml</ows:Value>
                <ows:Value>application/octet-stream</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ResultType">
                <ows:Value>hits</ows:Value>
                <ows:Value>results</ows:Value>
                <ows:Value>validate</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ElementSetName">
                <ows:Value>brief</ows:Value>
                <ows:Value>summary</ows:Value>
                <ows:Value>full</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Operation name="Transaction">
            <ows:DCP>
                <ows:HTTP>
```

```
                    <ows:Post ns2:href="https://<FQDN>:<PORT>/services/csw">
                        <ows:Constraint name="PostEncoding">
                            <ows:Value>XML</ows:Value>
                        </ows:Constraint>
                    </ows:Post>
                </ows:HTTP>
            </ows:DCP>
            <ows:Parameter name="typeNames">
                <ows:Value>xml</ows:Value>
                <ows:Value>appxml</ows:Value>
                <ows:Value>csw:Record</ows:Value>
                <ows:Value>gmd:MD_Metadata</ows:Value>
                <ows:Value>tika</ows:Value>
            </ows:Parameter>
            <ows:Parameter name="ConstraintLanguage">
                <ows:Value>Filter</ows:Value>
                <ows:Value>CQL_Text</ows:Value>
            </ows:Parameter>
        </ows:Operation>
        <ows:Parameter name="service">
            <ows:Value>CSW</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="version">
            <ows:Value>2.0.2</ows:Value>
        </ows:Parameter>
    </ows:OperationsMetadata>
    <ogc:Filter_Capabilities>
        <ogc:Spatial_Capabilities>
            <ogc:GeometryOperands>
                <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
                <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
                <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
            </ogc:GeometryOperands>
            <ogc:SpatialOperators>
                <ogc:SpatialOperator name="BBOX"/>
                <ogc:SpatialOperator name="Beyond"/>
                <ogc:SpatialOperator name="Contains"/>
                <ogc:SpatialOperator name="Crosses"/>
                <ogc:SpatialOperator name="Disjoint"/>
                <ogc:SpatialOperator name="DWithin"/>
                <ogc:SpatialOperator name="Intersects"/>
                <ogc:SpatialOperator name="Overlaps"/>
                <ogc:SpatialOperator name="Touches"/>
                <ogc:SpatialOperator name="Within"/>
            </ogc:SpatialOperators>
        </ogc:Spatial_Capabilities>
        <ogc:Scalar_Capabilities>
            <ogc:LogicalOperators/>
```

```
            <ogc:ComparisonOperators>
                <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
                <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
            </ogc:ComparisonOperators>
        </ogc:Scalar_Capabilities>
        <ogc:Id_Capabilities>
            <ogc:EID/>
        </ogc:Id_Capabilities>
    </ogc:Filter_Capabilities>
</csw:Capabilities>
```

11.6.5. Catalog REST Endpoint

The Catalog REST Endpoint allows clients to perform operations on the Catalog using REST, a simple architectural style that performs communication using HTTP.

Bulk operations are not supported: for all RESTful CRUD commands, only one metacard ID is supported in the URL.

The Catalog REST Endpoint can be used for one or more of these operations on an instance of DDF:

- Ingest metacards or resources into the DDF catalog.
- Retrieve metacards or resources from the catalog.
- Update metacards or resources in the catalog.
- Delete resources and metadata from the catalog.
- Get information about configured sources.

This example metacard can be used to test the integration with DDF.

Example Metacard

```
<?xml version="1.0" encoding="UTF-8"?>
<metacard xmlns="urn:catalog:metacard" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil="http://www.w3.org/2001/SMIL20/"
xmlns:smillang="http://www.w3.org/2001/SMIL20/Language" gml:id=
"3a59483ba44e403a9f0044580343007e">
  <type>ddf.metacard</type>
```

```
  <string name="title">
    <value>Test REST Metacard</value>
  </string>
  <string name="description">
    <value>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla scelerisque
cursus mi.</value>
  </string>
</metacard>
```

11.6.5.1. Catalog REST Create Operation Examples

The REST endpoint can be used to upload resources as attachments.

Send a `POST` request with the input to be ingested contained in the HTTP request body to the endpoint.

Create Request URL

```
https://<FQDN>:<PORT>/services/catalog/
```

Example Create Request

```
POST /services/catalog?transform=xml HTTP/1.1
Host: <FQDN>:<PORT>
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
Cache-Control: no-cache

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="parse.resource"; filename=""
Content-Type:


------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="parse.metadata"; filename=""
Content-Type:


------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

The `create` and `update` methods both support the multipart mime format. If only a single attachment exists, it is interpreted as a resource to be parsed, which results in a metacard and resource being stored in the system.

If multiple attachments exist, then the REST endpoint assumes that one attachment is the actual resource (attachment should be named `parse.resource`) and the other attachments are overrides of metacard attributes (attachment names should follow metacard attribute names). In the case of the metadata attribute, it is possible to also have the system transform that metadata and use the results of

that to override the metacard that would be generated from the resource (attachment should be named `parse.metadata`).

Create Success

If the ingest is successful, a status of `201 Created` is returned, along with the Metacard ID in the header of the response.

NOTE	*Request with Non-XML Data* If a request with non-XML data is sent to the Catalog REST endpoint, the metacard is created but the resource is stored in the `metadata` field. This could affect discoverability.

If content or metadata is not ingested successfully, check for these error messages.

Table 37. Create Error Responses

Status Code	Error Message	Possible Causes
`400 Bad Request`	`<pre>Error while storing entry in catalog: </pre>`	*Malformed XML Response*: If the XML being ingested has formatting errors.
		Request with Unknown Schema: If ingest is attempted with a schema that is unknown, unsupported, or not configured by the endpoint, DDF creates a generic resource metacard with the provided XML as content for the `metadata` XML field in the metacard.

11.6.5.2. Catalog REST Read Operation Examples

The `read` operation can be used to retrieve metadata in different formats.

1. Send a `GET` request to the endpoint.
2. Optionally add a `transform` query parameter to the end of the URL with the transformer to be used (such as `transform=kml`). By default, the response body includes the XML representation of the Metacard.

Read Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>
```

If successful, a status of `200 OK` is returned, along with the content of the metacard requested.

Read Success Response Example

```
<metacard xmlns="urn:catalog:metacard" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:smil="http://www.w3.org/2001/SMIL20/"
```

```
xmlns:smillang="http://www.w3.org/2001/SMIL20/Language" gml:id="<METACARD_ID>">
    <type>ddf.metacard</type>
    <source>ddf.distribution</source>
    <string name="title">
        <value>Test REST Metacard</value>
    </string>
    <string name="point-of-contact">
        <value>email@example.com</value>
    </string>
    <dateTime name="metacard.created">
        <value>2026-04-24</value>
    </dateTime>
    <dateTime name="effective">
        <value>2026-04-24</value>
    </dateTime>
    <dateTime name="modified">
        <value>2026-04-24</value>
    </dateTime>
    <dateTime name="created">
        <value>2026-04-24</value>
    </dateTime>
    <string name="description">
        <value>Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla scelerisque
cursus mi.</value>
    </string>
    <string name="metacard-tags">
        <value>resource</value>
        <value>VALID</value>
    </string>
    <dateTime name="metacard.modified">
        <value>2026-04-24</value>
    </dateTime>
</metacard>
```

- To receive metadata in an alternate format, add a transformer to the request URL.

Metacard Transform Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>?transform=<TRANSFORMER_ID>
```

Metacard Transform Response (`transform=geojson`)

```
{
    "geometry": null,
    "type": "Feature",
    "properties": {
        "effective": "2026-04-24",
```

```
        "point-of-contact": "email@example.com",
        "created": "2026-04-24",
        "metacard.modified": "2026-04-24",
        "metacard-tags": [
            "resource",
            "VALID"
        ],
        "modified": "2026-04-24",
        "description": "Vestibulum quis ipsum sit amet metus imperdiet vehicula. Nulla
scelerisque cursus mi.",
        "id": "3a59483ba44e403a9f0044580343007e",
        "metacard-type": "ddf.metacard",
        "title": "Test REST Metacard",
        "source-id": "ddf.distribution",
        "metacard.created": "2026-04-24"
    }
}
```


To retrieve a metacard from a specific federated source, add `sources/<SOURCE_ID>` to the URL.

Federated Read Request URL

```
https://<FQDN>:<PORT>/services/catalog/sources/<sourceId>/<metacardId>?transform=<TRANSFO
RMER_ID>
```

To retrieve the resource associated with a metacard, use the `resource` transformer with the `GET` request.

Retrieve Resource Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>?transform=resource
```

See Metacard Transformers for details on metacard transformers.

Read Error Response Examples

If the metacard or resource is not returned successfully, check for these errors.

Table 38. Read Error Responses

Status Code	Error Message	Possible Causes
`404 Not Found`	`<pre>Unable to retrieve requested metacard.</pre>`	Invalid Metacard ID

Status Code	Error Message	Possible Causes
500 Server Error	<pre>Unknown error occurred while processing request.</pre>	Transformer is invalid, unsupported, or not configured.
	<pre>Unable to transform Metacard. Try different transformer: </pre>	Metacard does not have an associated resource (is metadata only).
	<pre>Resource file is not available</pre>	The resource file/data is not accessible or doesn't exist
	<pre>READ failed due to unexpected exception: </pre>	Invalid source ID, or source unavailable.

11.6.5.3. Catalog Rest Update Operation Examples

To update the metadata for a metacard, send a PUT request with the ID of the Metacard to be updated appended to the end of the URL and the updated metadata is contained in the HTTP body.

Optionally, specify the transformer to use when parsing an override of a metadata attribute.

Update Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>?transform=<input transformer>
```

Table 39. Update Error Response Examples

Status Code	Error Message	Possible Causes
400 Bad Request	<pre>Error cataloging updated metadata: </pre>	Invalid metacard ID.
500 Server Error	<pre>Error cataloging updated metadata: </pre>	Invalid transformer ID.

11.6.5.4. Catalog REST Delete Operation Examples

To delete a metacard, send a DELETE request with the metacard ID to be deleted appended to the end of the URL.

Delete Request URL

```
https://<FQDN>:<PORT>/services/catalog/<metacardId>
```

Table 40. Delete Error Response Examples

Status Code	Error Message	Possible Causes
`400 Bad Request`	`<pre>Error deleting entry from catalog: </pre>`	Invalid metacard ID.

11.6.5.5. Catalog REST Sources Operation Examples

To retrieve information about federated sources, including `sourceId`, `availability`, `contentTypes`,and `version`, send a `GET` request to the endpoint.

Sources Response URL

```
https://<FQDN>:<PORT>//sources/
```

Sources Response Example

```
[
   {
      "id" : "DDF-OS",
      "available" : true,
      "contentTypes" :
         [
         ],
      "version" : "2.30.1"
   },
   {
      "id" : "ddf.distribution",
      "available" : true,
      "contentTypes" :
         [
         ],
      "version" : "2.30.1"
   }
]
```

Table 41. Sources Error Responses

Status Code	Error Message	Possible Causes
403	Problem accessing /ErrorServlet. Reason: <pre>Forbidden</pre>	Connection error or service unavailable.

Developing

Developers build or extend the functionality of the applications.

DDF includes several extension points where external developers can add functionality to support individual use cases.

DDF is written in Java and uses many open source libraries. DDF uses OSGi to provide modularity, lifecycle management, and dynamic services. OSGi services can be installed and uninstalled while DDF is running. DDF development typically means developing new OSGi bundles and deploying them to the running DDF. A complete description of OSGi is outside the scope of this documentation. For more information about OSGi, see the OSGi Alliance website.

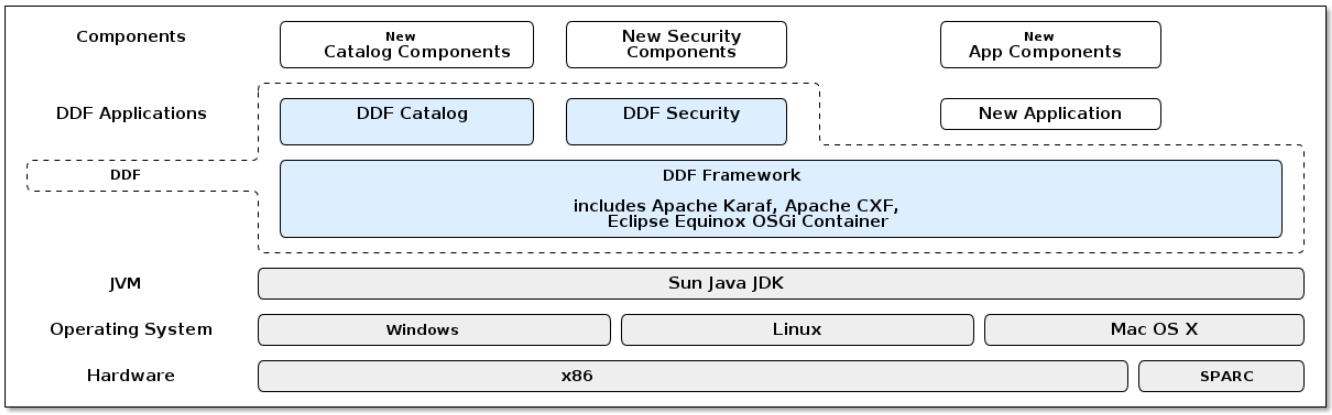


Figure 7. Architecture Diagram

IMPORTANT	If developing for a Highly Available Cluster of DDF, see High Availability Guidance.

12. Catalog Framework API

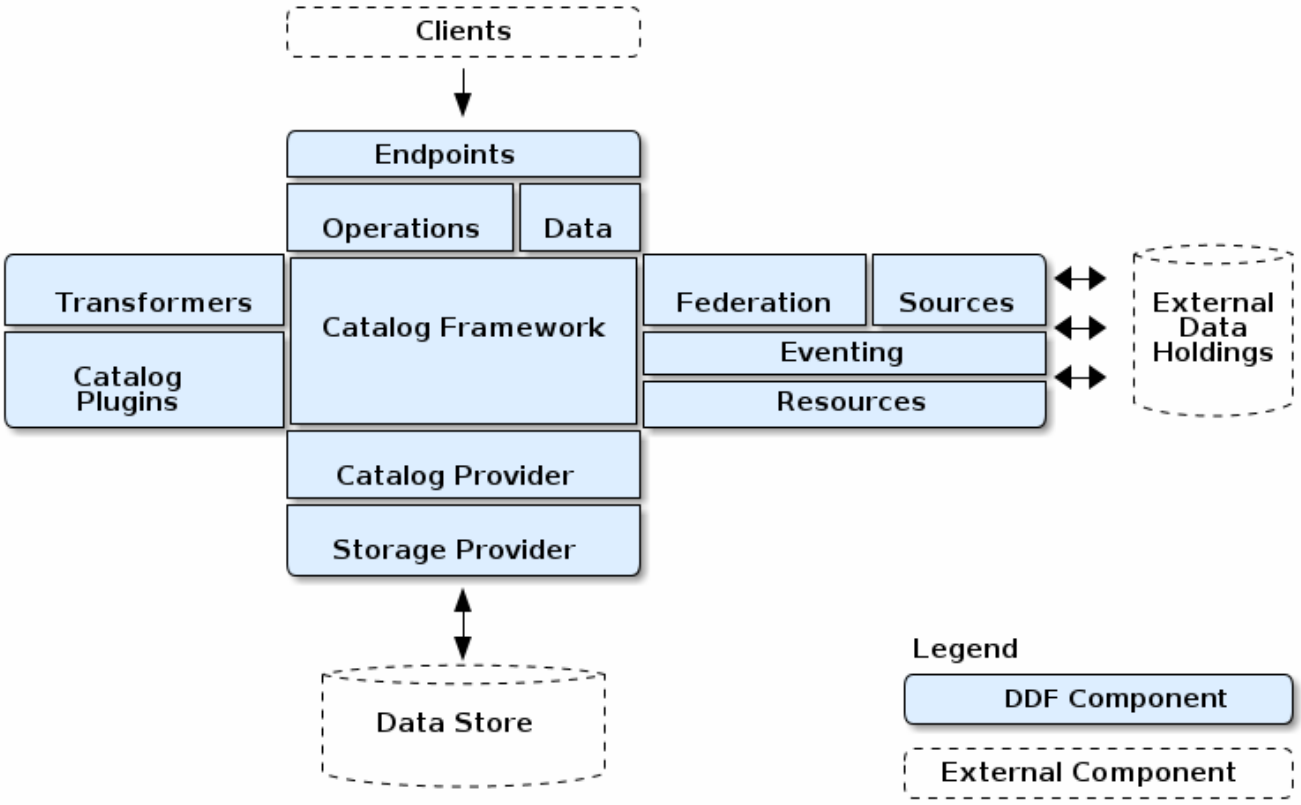


Figure 8. Catalog Architecture

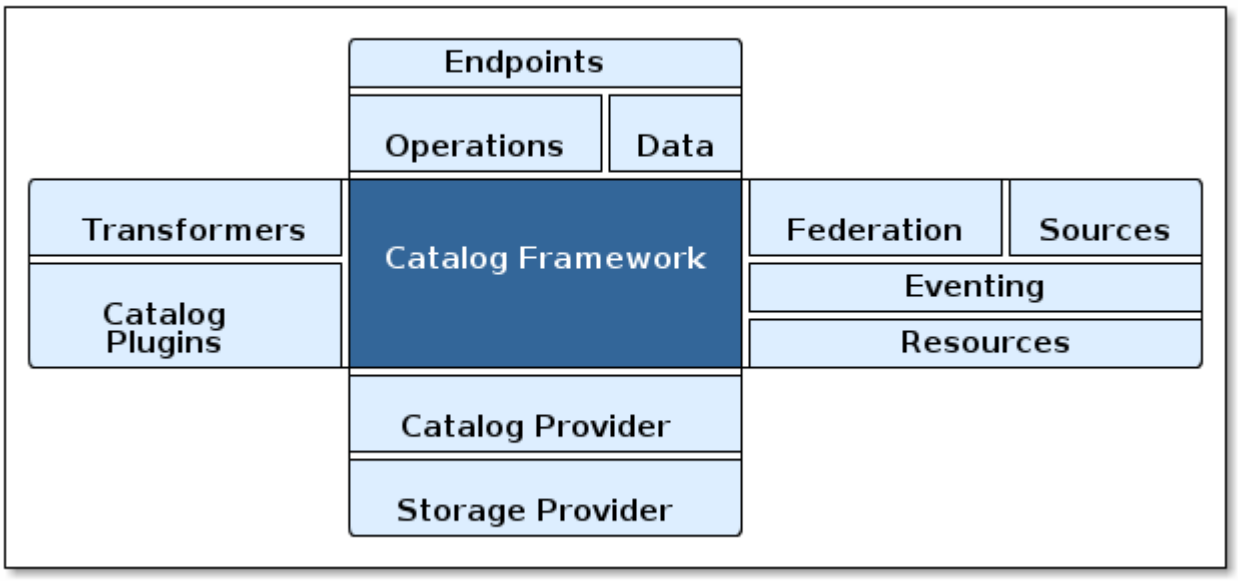


Figure 9. Catalog Framework Architecture

The `CatalogFramework` is the routing mechanism between catalog components that provides integration points for the Catalog Plugins. An endpoint invokes the active Catalog Framework, which calls any

configured Pre-query or Pre-ingest plug-ins. The selected federation strategy calls the active Catalog Provider and any connected or federated sources. Then, any Post-query or Post-ingest plug-ins are invoked. Finally, the appropriate response is returned to the calling endpoint.

The Catalog Framework wires all Catalog components together.

It is responsible for routing Catalog requests and responses to the appropriate target.

Endpoints send Catalog requests to the Catalog Framework. The Catalog Framework then invokes Catalog Plugins, Transformers, and Resource Components as needed before sending requests to the intended destination, such as one or more Sources.

The Catalog Framework decouples clients from service implementations and provides integration points for Catalog Plugins and convenience methods for Endpoint developers.

13. Catalog API Design

The Catalog is composed of several components and an API that connects them together. The Catalog API is central to DDF's architectural qualities of extensibility and flexibility. The Catalog API consists of Java interfaces that define Catalog functionality and specify interactions between components.

These interfaces provide the ability for components to interact without a dependency on a particular underlying implementation, thus allowing the possibility of alternate implementations that can maintain interoperability and share developed components. As such, new capabilities can be developed independently, in a modular fashion, using the Catalog API interfaces and reused by other DDF installations.

13.1. Ensuring Compatibility

The Catalog API evolves, but great care is taken to retain backwards compatibility with developed components. Compatibility is reflected in version numbers.

13.2. Catalog Framework Sequence Diagrams

Because the Catalog Framework plays a central role to Catalog functionality, it interacts with many different Catalog components. To illustrate these relationships, high-level sequence diagrams with notional class names are provided below. These examples are for illustrative purposes only and do not necessarily represent every step in each procedure.

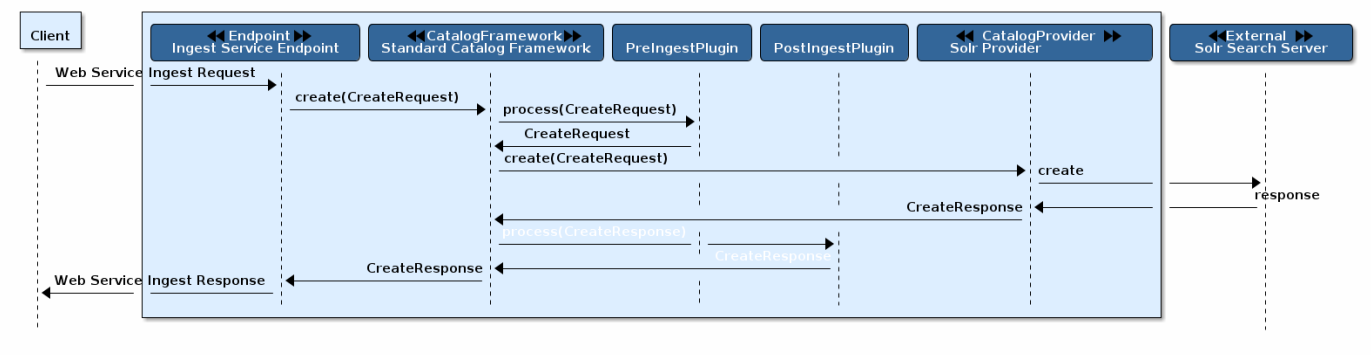


Figure 10. Ingest Request Data Flow

The Ingest Service Endpoint, the Catalog Framework, and the Catalog Provider are key components of the Reference Implementation. The Endpoint bundle implements a Web service that allows clients to create, update, and delete metacards. The Endpoint calls the `CatalogFramework` to execute the operations of its specification. The `CatalogFramework` routes the request through optional `PreIngest` and `PostIngest` Catalog Plugins, which may modify the ingest request/response before/after the Catalog Provider executes the ingest request and provides the response. Note that a `CatalogProvider` must be present for any ingest requests to be successfully processed, otherwise a fault is returned.

This process is similar for updating catalog entries, with update requests calling the `update(UpdateRequest)` methods on the Endpoint, `CatalogFramework`, and Catalog Provider. Similarly, for deletion of catalog entries, the delete requests call the `delete(DeleteRequest)` methods on the `Endpoint`, `CatalogFramework`, and `CatalogProvider`.

13.2.1. Error Handling

Any ingest attempts that fail inside the Catalog Framework (whether the failure comes from the Catalog Framework itself, pre-ingest plugin failures, or issues with the Catalog Provider) are logged to a separate log file for ease of error handling. The file is located at `<DDF_HOME>/data/log/ingest_error.log` and it logs the Metacards that fail, their ID and Title name, and the stack trace associated with their failure. By default, successful ingest attempts are not logged. However, that functionality can be achieved by setting the log level of the `ingestLogger` to DEBUG (note that enabling DEBUG can cause a non-trivial performance hit).

TIP

To turn off logging failed ingest attempts into a separate file, execute the following via the command line console

```
log:set
 ERROR ingestLogger
```

13.2.2. Query

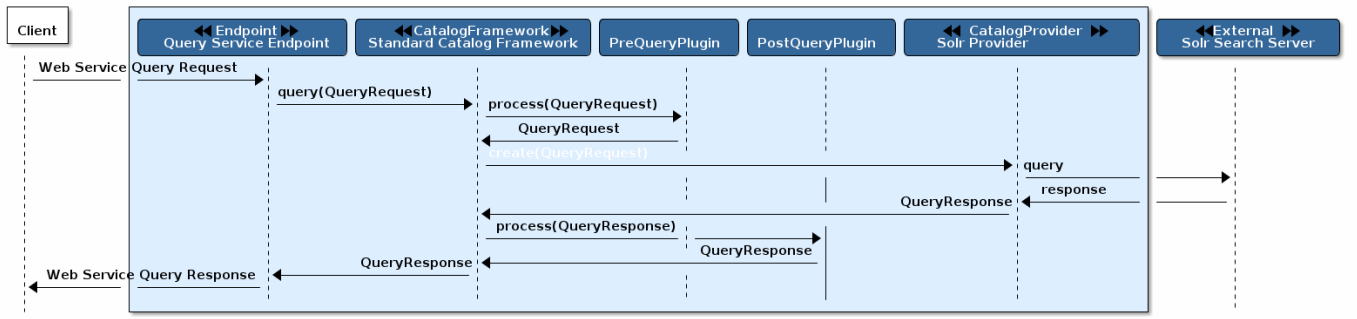


Figure 11. Query Request Data Flow

The Query Service Endpoint, the Catalog Framework, and the `CatalogProvider` are key components for processing a query request as well. The Endpoint bundle contains a Web service that exposes the interface to query for `Metacards`. The Endpoint calls the `CatalogFramework` to execute the operations of its specification. The `CatalogFramework` relies on the `CatalogProvider` to execute the actual query. Optional PreQuery and PostQuery Catalog Plugins may be invoked by the `CatalogFramework` to modify the query request/response prior to the Catalog Provider processing the query request and providing the query response. If a `CatalogProvider` is not configured, and no other remote Sources are configured, a fault is returned. It is possible to have only remote Sources configured and no local `CatalogProvider` configured and be able to execute queries to specific remote Sources by specifying the site names in the query request.

13.2.3. Product Caching

The Catalog Framework optionally provides caching of products, so future requests to retrieve the same product are serviced much quicker. If caching is enabled, each time a retrieve product request is received, the Catalog Framework looks in its cache (default location `<DDF_HOME>/data/product-cache`) to see if the product has been cached locally. If it has, the product is retrieved from the local site and returned to the client, providing a much quicker turnaround because remote product retrieval and network traffic was avoided. If the requested product is not in the cache, the product is retrieved from the Source (local or remote) and cached locally while returning the product to the client. The caching to a local file of the product and the streaming of the product to the client are done simultaneously so that the client does not have to wait for the caching to complete before receiving the product. If errors are detected during the caching, caching of the product is abandoned, and the product is returned to the client.

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

If the admin has enabled the **Always Cache When Canceled** option, caching of the product occurs

even if the client cancels the product retrieval so that future requests are serviced quickly. Otherwise, caching is canceled if the user cancels the product download.

13.2.4. Product Download Status

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

13.2.5. Catalog API

The Catalog API is an OSGi bundle (`catalog-core-api`) that contains the Java interfaces for the Catalog components and implementation classes for the Catalog Framework, Operations, and Data components.

13.2.5.1. Catalog API Search Interfaces

The Catalog API includes two different search interfaces.

Search UI Application Search Interface

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

SSH Search Interface

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

13.2.5.2. Catalog Search Result Objects

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

Distance

Populated when a point-radius spatial search occurs. Numerical value that indicates the result's distance from the center point of the search.

Units

Populated when a point-radius spatial search occurs. Indicates the units (kilometer, mile, etc.) for the distance field.

Relevance

Populated when a contextual search occurs. Numerical value that indicates how relevant the text in the result is to the text originally searched for.

13.2.5.3. Search Programmatic Flow

Searching the catalog involves three basic steps:

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

13.2.5.4. Sort Policies

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

There are three policies available.

Table 42. Sort Policies

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

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

13.2.5.5. Product Retrieval

The DDF is used to catalog resources. A Resource is a URI-addressable entity that is represented by a Metacard. Resources may exist either locally or on a remote data store.

Examples of Resources

- NITF image
- MPEG video
- Live video stream
- Audio recording
- Document

Product Retrieval Services

- SOAP Web services
- DDF JSON
- DDF REST

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

Product Caching

Product Caching is enabled by default. Existing DDF clients are able to leverage product caching due to the product cache being implemented in the DDF.

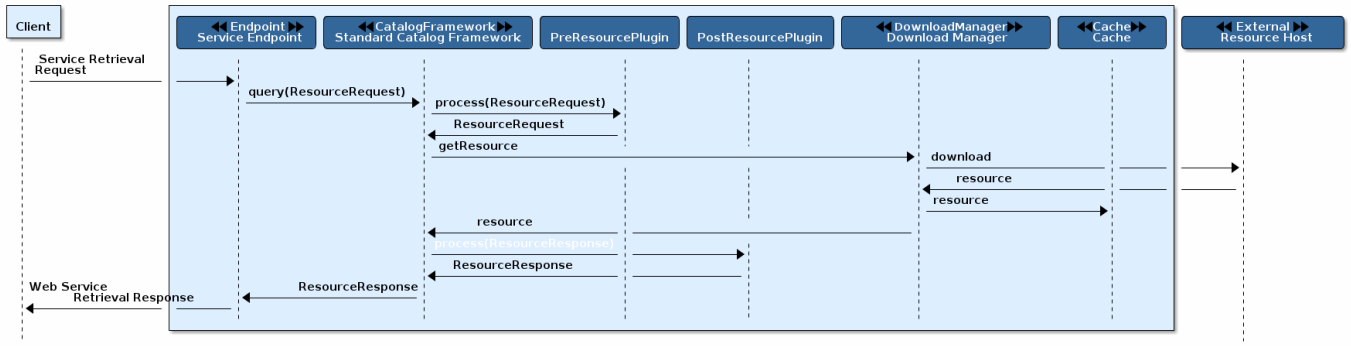


Figure 12. Product Retrieval Request

13.2.5.6. Notifications and Activities

DDF can send/receive notifications of "Activities" occurring in the system.

Currently, the notifications provide information about resource retrieval only.

Activity events include the status and progress of actions that are being performed by the user, such as searches and downloads.

13.3. Included Catalog Frameworks, Associated Components, and Configurations

These catalog frameworks are available in a standard DDF installation:

Standard Catalog Framework

Reference implementation of a Catalog Framework that implements all requirements of the Catalog API.

Catalog Framework Camel Component

Supports creating, updating, and deleting metacards using the Catalog Framework from a Camel route.

13.3.1. Standard Catalog Framework

The Standard Catalog Framework provides the reference implementation of a Catalog Framework that implements all requirements of the Catalog API. `CatalogFrameworkImpl` is the implementation of the DDF Standard Catalog Framework.

The Standard Catalog Framework is the core class of DDF. It provides the methods for create, update, delete, and resource retrieval (CRUD) operations on the `Sources`. By contrast, the Fanout Catalog Framework only allows for query and resource retrieval operations, no catalog modifications, and all queries are enterprise-wide.

Use this framework if:

- access to a catalog provider is required to create, update, and delete catalog entries.
- queries to specific sites are required.
- queries to only the local provider are required.

It is possible to have only remote Sources configured with no local `CatalogProvider` configured and be able to execute queries to specific remote sources by specifying the site names in the query request.

The Standard Catalog Framework also maintains a list of `ResourceReaders` for resource retrieval operations. A resource reader is matched to the scheme (the protocol, such as `file://`) in the URI of the resource specified in the request to be retrieved.

Site information about the catalog provider and/or any federated sources can be retrieved using the Standard Catalog Framework. Site information includes the source's name, version, availability, and the list of unique content types currently stored in the source (such as NITF). If no local catalog provider is configured, the site information returned includes site info for the catalog framework with no content types included.

13.3.1.1. Installing the Standard Catalog Framework

The Standard Catalog Framework is bundled as the `catalog-core-standardframework` feature and can be installed and uninstalled using the normal processes described in Configuration.

13.3.1.2. Configuring the Standard Catalog Framework

These are the configurable properties on the Standard Catalog Framework.

See Catalog Standard Framework configurations for all possible configurations.

Table 43. Standard Catalog Framework Exported Services

Registered Interface	Service Property	Value
`ddf.catalog.federation.FederationStrategy`	shortname	`sorted`
`org.osgi.service.event.EventHandler`	event.topics	`ddf/catalog/event/CREATED`, `ddf/catalog/event/UPDATED`, `ddf/catalog/event/DELETED`
`ddf.catalog.CatalogFramework`		
`ddf.catalog.event.EventProcessor`		
`ddf.catalog.plugin.PostIngestPlugin`		

Table 44. Standard Catalog Framwork Imported Services

Registered Interface	Availability	Multiple
`ddf.catalog.plugin.PostFederatedQueryPlugin`	optional	`true`
`ddf.catalog.plugin.PostIngestPlugin`	optional	`true`
`ddf.catalog.plugin.PostQueryPlugin`	optional	`true`
`ddf.catalog.plugin.PostResourcePlugin`	optional	`true`
`ddf.catalog.plugin.PreDeliveryPlugin`	optional	`true`
`ddf.catalog.plugin.PreFederatedQueryPlugin`	optional	`true`
`ddf.catalog.plugin.PreIngestPlugin`	optional	`true`
`ddf.catalog.plugin.PreQueryPlugin`	optional	`true`
`ddf.catalog.plugin.PreResourcePlugin`	optional	`true`
`ddf.catalog.plugin.PreSubscriptionPlugin`	optional	`true`

Registered Interface	Availability	Multiple
ddf.catalog.plugin.PolicyPlugin	optional	true
ddf.catalog.plugin.AccessPlugin	optional	true
ddf.catalog.resource.ResourceReader	optional	true
ddf.catalog.source.CatalogProvider	optional	true
ddf.catalog.source.ConnectedSource	optional	true
ddf.catalog.source.FederatedSource	optional	true
ddf.cache.CacheManager		false
org.osgi.service.event.EventAdmin		false

13.3.1.3. Known Issues with Standard Catalog Framework

None.

13.3.2. Catalog Framework Camel Component

The Catalog Framework Camel Component supports creating, updating, and deleting metacards using the Catalog Framework from a Camel route.

URI Format

```
catalog:framework
```

Table 45. Catalog Framework Producer Message Headers

Header	Description
operation	the operation to perform using the Catalog Framework (possible values are CREATE | UPDATE | DELETE)

13.3.2.1. Sending Messages to Catalog Framework Endpoint

Catalog Framework Producer

In Producer mode, the component provides the ability to supply different inputs and have the Catalog Framework perform different operations based upon the header values.

For the CREATE and UPDATE operation, the message body can contain a list of metacards or a single metacard object.

For the DELETE operation, the message body can contain a list of strings or a single string object. The string objects represent the IDs of metacards to be deleted. The exchange's "in" message is set with the affected metacards. In the case of a CREATE, it is updated with the created metacards. In the case of the UPDATE, it is updated with the updated metacards and with the DELETE it contains the deleted

metacards.

Table 46. Catalog Framework Camel Component Operations

Header	Message Body (Input)	Exchange Modification (Output)
operation = CREATE	List<Metacard> or Metacard	exchange.getIn().getBody() updated with List of Metacards created
operation = UPDATE	List<Metacard> or Metacard	exchange.getIn().getBody() updated with List of Metacards updated
operation = DELETE	List<String> or String (representing metacard IDs)	exchange.getIn().getBody() updated with List of Metacards deleted

NOTE If there is an exception thrown while the route is being executed, a `FrameworkProducerException` is thrown causing the route to fail with a CamelExecutionException.

Example Route

This example demonstrates:

1. Reading in some sample data from the file system.
2. Using a Java bean to convert the data into a metacard.
3. Setting a header value on the Exchange.
4. Sending the Metacard to the Catalog Framework component for ingesting.

```
<route>
 <from uri="file:data/sampleData?noop=true"/>
    <bean ref="sampleDataToMetacardConverter" method="covertToMetacard"/>\
   <setHeader name="operation">
  <constant>CREATE</constant>
 </setHeader>
    <to uri="catalog:framework"/>
</route>
```

14. Transformers

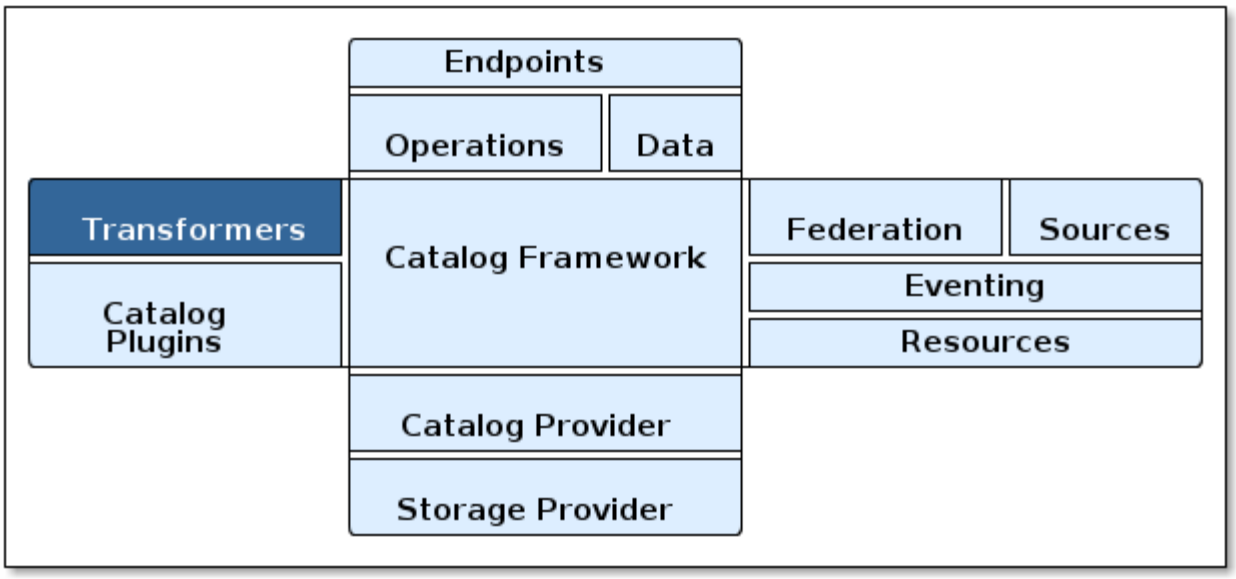


Figure 13. Transformers

Transformers transform data to and from various formats. Transformers are categorized by when they are invoked and used. The existing types are Input transformers, Metacard transformers, and Query Response transformers. Additionally, XSLT transformers are provided to aid in developing custom, lightweight Metacard and Query Response transformers.

Transformers are utility objects used to transform a set of standard DDF components into a desired format, such as into PDF, GeoJSON, XML, or any other format. For instance, a transformer can be used to convert a set of query results into an easy-to-read GeoJSON format (GeoJSON Transformer) or convert a set of results into a RSS feed that can be easily published to a URL for RSS feed subscription. Transformers can be registered in the OSGi Service Registry so that any other developer can access them based on their standard interface and self-assigned identifier, referred to as its "shortname." Transformers are often used by endpoints for data conversion in a system standard way. Multiple endpoints can use the same transformer, a different transformer, or their own published transformer.

WARNING	The current transformers only work for UTF-8 characters and do not support Non-Western Characters (for example, Hebrew). It is recommend not to use international character sets, as they may not be displayed properly.

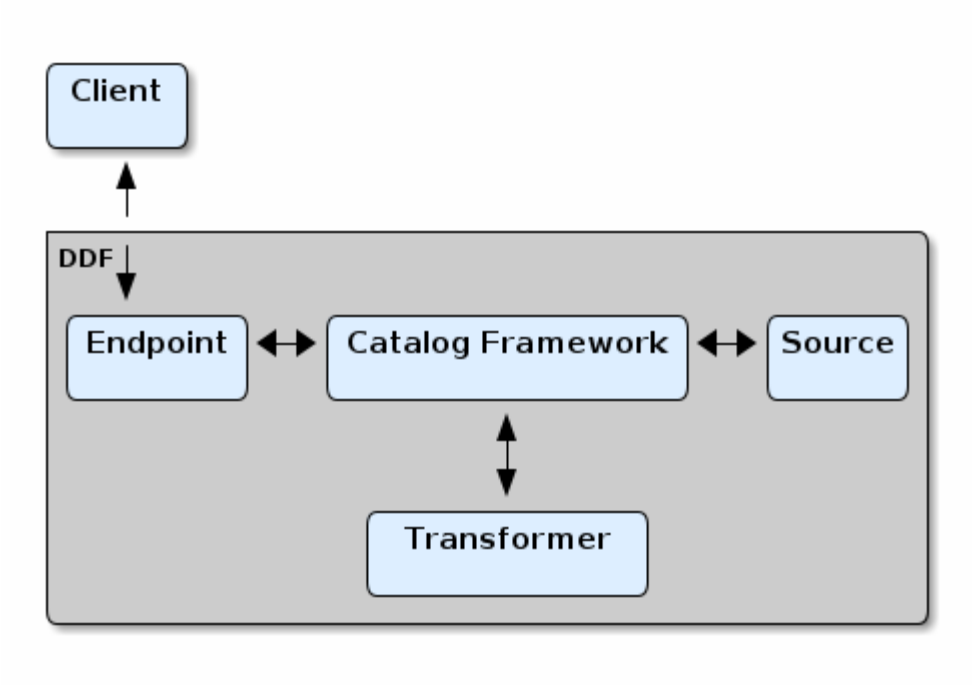


Figure 14. Communication Diagram

Transformers are used to alter the format of a resource or its metadata to or from the catalog's metacard format.

Types of Transformers

Input Transformers

Input Transformers create metacards from input. Once converted to a Metacard, the data can be used in a variety of ways, such as in an `UpdateRequest`, `CreateResponse`, or within Catalog Endpoints or Sources. For instance, an input transformer could be used to receive and translate XML into a Metacard so that it can be placed within a `CreateRequest` to be ingested within the Catalog. Input transformers should be registered within the Service Registry with the interface `ddf.catalog.transform.InputTransformer` to notify Catalog components of any new transformers.

Metacard Transformers

Metacard Transformers translate a metacard from catalog metadata to a specific data format.

Query Response Transformers

Query Response transformers convert query responses into other data formats.

14.1. Available Input Transformers

The following input transformers are available in a standard installation of DDF:

PPTX Input Transformer

Translates Microsoft PowerPoint (OOXML only) documents into Catalog Metacards.

XML Input Transformer

Translates an XML document into a Catalog Metacard.

PDF Input Transformer

Translates a PDF document into a Catalog Metacard.

Video Input Transformer

Creates Catalog metacards from certain video file types.

GeoJSON Input Transformer

Translates GeoJSON into a Catalog metacard.

Tika Input Transformer

Translates Microsoft Word, Microsoft Excel, Microsoft PowerPoint, OpenOffice Writer, and PDF documents into Catalog records.

14.2. Available Metacard Transformers

The following metacard transformers are available in a standard installation of DDF:

Metadata Metacard Transformer

returns the `Metacard.METADATA` attribute when given a metacard.

Thumbnail Metacard Transformer

Retrieves the thumbnail bytes of a Metacard by returning the `Metacard.THUMBNAIL` attribute value.

KML Style Mapper

Maps a KML Style URL to a metacard based on that metacard's attributes.

XML Metacard Transformer

Translates a metacard into an XML-formatted document.

Resource Metacard Transformer

Retrieves the resource bytes of a metacard by returning the resource associated with the metacard.

GeoJSON Metacard Transformer

Translates a metacard into GeoJSON.

KML Metacard Transformer

Translates a metacard into a KML-formatted document.

14.3. Available Query Response Transformers

The following query response transformers are available in a standard installation of DDF:

GeoJSON Query Response Transformer

Translates a query response into a GeoJSON-formatted document.

Atom Query Response Transformer

Transforms a query response into an Atom 1.0 feed.

XML Query Response Transformer

Translates a query response into an XML-formatted document.

CSW Query Response Transformer

Transforms a query response into a CSW-formatted document.

Query Response Transformer Consumer

Translates a query response into a Catalog Metacard.

KML Query Response Transformer

Translates a query response into a KML-formatted document.

14.4. Transformers Details

Availability and configuration details of available transformers.

14.4.1. Metadata Metacard Transformer

The Metadata Metacard Transformer returns the `Metacard.METADATA` attribute when given a metacard. The MIME Type returned is `text/xml`.

14.4.1.1. Installing the Metadata Metacard Transformer

The Metadata Metacard Transformer is installed by default in a standard installation with the Catalog application.

14.4.1.2. Configuring the Metadata Metacard Transformer

The Metadata Metacard Transformer has no configurable properties.

14.4.1.3. Using the Metadata Metacard Transformer

The Metadata Metacard Transformer can be used programmatically by requesting a metacard transformer with the id `metadata`. It can also be used within the REST Endpoint by providing the transform option as `metadata`.

Example REST GET method with the Metadata Metacard Transformer

```
http://{FQDN}:{PORT}/services/catalog/0123456789abcdef0123456789abcdef?transform=metadata
```

14.4.2. PPTX Input Transformer

The PPTX Input Transformer translates Microsoft PowerPoint (OOXML only) documents into Catalog Metacards, using Apache Tika for basic metadata and Apache POI for thumbnail creation. The PPTX Input Transformer ingests PPTX documents into the DDF Content Repository and the Metadata Catalog, and adds a thumbnail of the first page in the PPTX document.

The PPTX Input Transformer takes precedence over the Tika Input Transformer for PPTX documents.

Table 47. PPTX Input Transformer Usage

Schema	Mime-types
N/A	`application/vnd.openxmlformats-officedocument.presentationml.presentation`

14.4.2.1. Installing the PPTX Input Transformer

This transformer is installed by default with a standard installation in the Catalog application.

14.4.2.2. Configuring the PPTX Input Transformer

The PPTX Input Transformer has no configurable properties. "'

14.4.3. GeoJSON Query Response Transformer

The GeoJSON Query Response Transformer translates a query response into a GeoJSON-formatted document.

14.4.3.1. Installing the GeoJSON Query Response Transformer

The GeoJSON Query Response Transformer is installed by default with a standard installation in the Catalog application.

14.4.3.2. Configuring the GeoJSON Query Response Transformer

The GeoJSON Query Response Transformer has no configurable properties.

14.4.4. Thumbnail Metacard Transformer

The Thumbnail Metacard Transformer retrieves the thumbnail bytes of a Metacard by returning the `Metacard.THUMBNAIL` attribute value.

14.4.4.1. Installing the Thumbnail Metacard Transformer

This transformer is installed by default with a standard installation in the Catalog application.

14.4.4.2. Configuring the Thumbnail Metacard Transformer

The Thumbnail Metacard Transformer has no configurable properties.

14.4.4.3. Using the Thumbnail Metacard Transformer

Endpoints or other components can retrieve an instance of the Thumbnail Metacard Transformer using its id `thumbnail`.

Sample Blueprint Reference Snippet

```
<reference id="metacardTransformer" interface="ddf.catalog.transform.MetacardTransformer"
filter="(id=thumbnail)"/>
```

The Thumbnail Metacard Transformer returns a `BinaryContent` object of the `Metacard.THUMBNAIL` bytes and a MIME Type of `image/jpeg`.

14.4.5. KML Style Mapper

The KML Style Mapper provides the ability for the `KMLTransformer` to map a KML Style URL to a metacard based on that metacard's attributes. For example, if a user wanted all JPEGs to be blue, the KML Style Mapper provides the ability to do so. This would also allow an administrator to configure metacards from each source to be different colors.

The configured style URLs are expected to be HTTP URLs. For more information on style URL's, refer to the KML Reference.

The KML Style Mapper supports all basic and extended metacard attributes. When a style mapping is configured, the resulting transformed KML contain a `<styleUrl>` tag pointing to that style, rather than the default KML style supplied by the `KMLTransformer`.

14.4.5.1. Installing the KML Style Mapper

The KML Style Mapper is installed by default with a standard installation in the Spatial Application in the `spatial-kml-transformer` feature.

14.4.5.2. Configuring the KML Style Mapper

The properties below describe how to configure a style mapping. The configuration name is `Spatial KML Style Map Entry`.

See KML Style Mapper configurations for all possible configurations.

KML Style Mapper Example Values

```
<xmlns="http://www.opengis.net/kml/2.2"
  xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xmlns:ns3="http://www.w3.org/2005/Atom">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
    <description>&lt;!DOCTYPE html&gt;
&lt;html&gt;
  &lt;head&gt;
    &lt;meta content="text/html; charset=windows-1252" http-equiv="content-type"&gt;
    &lt;style media="screen" type="text/css"&gt;
      .label {
        font-weight: bold
      }
      .linkTable {
width: 100% }
      .thumbnailDiv {
        text-align: center
} img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    &lt;/style&gt;
  &lt;/head&gt;
  &lt;body&gt;
        &lt;div class="thumbnailDiv"&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48
fab939da528e?transform=resource"&gt;&lt;img alt="Thumnail"
src=" CA=="&gt;&lt;/a&gt;&lt;/div&gt;
    &lt;table&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Source:&lt;/td&gt;
        &lt;td&gt;ddf.distribution&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Created:&lt;/td&gt;
        &lt;td&gt;Wed Oct 30 09:46:29 MDT 2013&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Effective:&lt;/td&gt;
        &lt;td&gt;2014-01-07T14:58:16-0700&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
    &lt;table class="linkTable"&gt;
      &lt;tr&gt;
        &lt;td&gt;&lt;a
```

```
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48
fab939da528e?transform=html"&gt;View Details...&lt;/a&gt;&lt;/td&gt;
        &lt;td&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48
fab939da528e?transform=resource"&gt;Download...&lt;/a&gt;&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
  &lt;/body&gt;
&lt;/html&gt;
</description>
    <TimeSpan>
      <begin>2014-01-07T21:58:16</begin>
    </TimeSpan>
 <styleUrl>http://example.com/kml/style#sampleStyle</styleUrl>
    <MultiGeometry>
      <Point>
        <coordinates>102.0,2.0</coordinates>
      </Point>
      <MultiGeometry>
        <Polygon>
          <outerBoundaryIs>
            <LinearRing>
              <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0
102.0,2.0</coordinates>
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
        <Polygon>
100.8,0.2
<outerBoundaryIs>
  <LinearRing>
    <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2
      100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
  </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</MultiGeometry>
</MultiGeometry>
</Placemark>
</kml>
```

14.4.6. Atom Query Response Transformer

The Atom Query Response Transformer transforms a query response into an Atom 1.0 feed. The Atom transformer maps a `QueryResponse` object as described in the Query Result Mapping.

14.4.6.1. Installing the Atom Query Response Transformer

The Atom Query Response Transformer is installed by default with a standard installation.

14.4.6.2. Configuring the Atom Query Response Transformer

The Atom Query Response Transformer has no configurable properties.

14.4.6.3. Using the Atom Query Response Transformer

Use this transformer when Atom is the preferred medium of communicating information, such as for feed readers or federation. An integrator could use this with an endpoint to transform query responses into an Atom feed.

For example, clients can use the OpenSearch Endpoint. The client can query with the format option set to the shortname, `atom`.

Sample OpenSearch Query with Atom Specified as Return Format

```
http://{FQDN}:{PORT}/services/catalog/query?q=ddf?format=atom
```

Developers could use this transformer to programmatically transform `QueryResponse` objects on the fly.

Sample Atom Feed from `QueryResponse` object

```
 <feed xmlns="http://www.w3.org/2005/Atom" xmlns:os="http://a9.com/-
/spec/opensearch/1.1/">
    <title type="text">Query Response</title>
    <updated>2017-01-31T23:22:37.298Z</updated>
    <id>urn:uuid:a27352c9-f935-45f0-9b8c-5803095164bb</id>
    <link href="#" rel="self" />
    <author>
        <name>Organization Name</name>
    </author>
    <generator version="2.1.0.20130129-1341">ddf123</generator>
    <os:totalResults>1</os:totalResults>
    <os:itemsPerPage>10</os:itemsPerPage>
    <os:startIndex>1</os:startIndex>
    <entry xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/"
xmlns:fs="http://a9.com/-/opensearch/extensions/federation/1.0/"
        xmlns:georss="http://www.georss.org/georss">
        <fs:resultSource fs:sourceId="ddf123" />
        <relevance:score>0.19</relevance:score>
        <id>urn:catalog:id:ee7a161e01754b9db1872bfe39d1ea09</id>
        <title type="text">F-15 lands in Libya; Crew Picked Up</title>
        <updated>2013-01-31T23:22:31.648Z</updated>
        <published>2013-01-31T23:22:31.648Z</published>
        <link href=
```

```
"http://123.45.67.123:8181/services/catalog/ddf123/ee7a161e01754b9db1872bfe39d1ea09" rel
="alternate" title="View Complete Metacard" />
        <category term="Resource" />
        <georss:where xmlns:gml="http://www.opengis.net/gml">
            <gml:Point>
                <gml:pos>32.8751900768792 13.1874561309814</gml:pos>
            </gml:Point>
        </georss:where>
        <content type="application/xml">
            <ns3:metacard xmlns:ns3="urn:catalog:metacard" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ns1="http://www.opengis.net/gml"
                xmlns:ns4="http://www.w3.org/2001/SMIL20/" xmlns:ns5=
"http://www.w3.org/2001/SMIL20/Language" ns1:id="4535c53fc8bc4404a1d32a5ce7a29585">
                <ns3:type>ddf.metacard</ns3:type>
                <ns3:source>ddf.distribution</ns3:source>
                <ns3:geometry name="location">
                    <ns3:value>
                        <ns1:Point>
                            <ns1:pos>32.8751900768792 13.1874561309814</ns1:pos>
                        </ns1:Point>
                    </ns3:value>
                </ns3:geometry>
                <ns3:dateTime name="created">
                    <ns3:value>2013-01-31T16:22:31.648-07:00</ns3:value>
                </ns3:dateTime>
                <ns3:dateTime name="modified">
                    <ns3:value>2013-01-31T16:22:31.648-07:00</ns3:value>
                </ns3:dateTime>
                <ns3:stringxml name="metadata">
                    <ns3:value>
                        <ns6:xml xmlns:ns6="urn:sample:namespace" xmlns=
"urn:sample:namespace">Example description.</ns6:xml>
                    </ns3:value>
                </ns3:stringxml>
                <ns3:string name="metadata-content-type-version">
                    <ns3:value>myVersion</ns3:value>
                </ns3:string>
                <ns3:string name="metadata-content-type">
                    <ns3:value>myType</ns3:value>
                </ns3:string>
                <ns3:string name="title">
                    <ns3:value>Example title</ns3:value>
                </ns3:string>
            </ns3:metacard>
        </content>
    </entry>
</feed>
```

Table 48. Atom Query Response Transformer Result Mapping

XPath to Atom XML	Value
`/feed/title`	"Query Response"
`/feed/updated`	ISO 8601 dateTime of when the feed was generated
`/feed/id`	Generated UUID URN
`/feed/author/name`	Platform Global Configuration organization
`/feed/generator`	Platform Global Configuration site name
`/feed/generator/@version`	Platform Global Configuration version
`/feed/os:totalResults`	SourceResponse Number of Hits
`/feed/os:itemsPerPage`	Request's Page Size
`/feed/os:startIndex`	Request's Start Index
`/feed/entry/fs:resultSource/@fs:sourceId`	Source Id from which the Result came. `Metacard.getSourceId()`
`/feed/entry/relevance:score`	Result's relevance score if applicable. `Result.getRelevanceScore()`
`/feed/entry/id`	`urn:catalog:id:<Metacard.ID>`
`/feed/entry/title`	`Metacard.TITLE`
`/feed/entry/updated`	ISO 8601 dateTime of `Metacard.MODIFIED`
`/feed/entry/published`	ISO 8601 dateTime of `Metacard.CREATED`
`/feed/entry/link[@rel='related']`	URL to retrieve underlying resource (if applicable and link is available)
`/feed/entry/link[@rel='alternate']`	Link to alternate view of the Metacard (if a link is available)
`/feed/entry/category`	`Metacard.CONTENT_TYPE`
`/feed/entry//georss:where`	GeoRSS GML of every Metacard attribute with format `AttributeFormat.GEOMETRY`
`/feed/entry/content`	Metacard XML generated by `DDF.catalog.transform.MetacardTransformer` with `shortname=xml`. If no transformer found, `/feed/entry/content/@type` is text and `Metacard.ID` is displayed `<content type="text">4e1f38d1913b4e93ac622e6c1b258f89</content>`

14.4.7. XML Metacard Transformer

The XML metacard transformer is responsible for translating a metacard into an XML-formatted

document. The metacard element that is generated is an extension of `gml:AbstractFeatureType`, which makes the output of this transformer GML 3.1.1 compatible.

14.4.7.1. Installing the XML Metacard Transformer

This transformer comes installed by default with a standard installation in the Catalog application.

To install or uninstall manually, use the `catalog-transformer-xml` feature.

14.4.7.2. Configuring the XML Metacard Transformer

The XML Metacard Transformer has no configurable properties.

14.4.7.3. Using the XML Metacard Transformer

Using the REST Endpoint for example, request a metacard with the transform option set to the XML shortname.

XML Metacard Transformer URL

```
https://{FQDN}:{PORT}/services/catalog/ac0c6917d5ee45bfb3c2bf8cd2ebaa67?transform=xml
```

Table 49. Metacard to XML Mappings

Metacard Variables	XML Element
`id`	`metacard/@gml:id`
`metacardType`	`metacard/type`
`sourceId`	`metacard/source`
`all other attributes`	`metacard/<AttributeType>[name='<AttributeName>']/value` `For instance, the value for the metacard attribute named "title" would be found at metacard/string[@name='title']/value`

XML Adapted Attributes (AttributeTypes)

- `boolean`
- `base64Binary`
- `dateTime`
- `double`
- `float`
- `geometry`
- `int`
- `long`

- `object`
- `short`
- `string`
- `stringxml`

14.4.8. XML Query Response Transformer

The XML Query Response Transformer is responsible for translating a query response into an XML-formatted document. The metacard element generated is an extension of `gml:AbstractFeatureCollectionType`, which makes the output of this transformer GML 3.1.1 compatible.

14.4.8.1. Installing the XML Query Response Transformer

This transformer is installed by default with a standard installation in the Catalog application. To uninstall, uninstall the `catalog-transformer-xml` feature.

14.4.8.2. Configuring the XML Query Response Transformer

To configure the XML Query Response Transformer:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the XML Query Response Transformer.

See XML Query Response Transformer configurations for all possible configurations.

14.4.8.3. Using the XML Query Response Transformer

Using the OpenSearch Endpoint, for example, query with the format option set to the XML shortname `xml`.

XML Query Response Transformer Query Example

```
http://{FQDN}:{PORT}/services/catalog/query?q=input?format=xml
```

XML Query Response Transformer Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:metacards xmlns:ns1="http://www.opengis.net/gml" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ns3="urn:catalog:metacard" xmlns:ns4=
"http://www.w3.org/2001/SMIL20/" xmlns:ns5="http://www.w3.org/2001/SMIL20/Language">
    <ns3:metacard ns1:id="000ba4dd7d974e258845a84966d766eb">
```

```
        <ns3:type>ddf.metacard</ns3:type>
        <ns3:source>southwestCatalog1</ns3:source>
        <ns3:dateTime name="created">
          <ns3:value>2013-04-10T15:30:05.702-07:00</ns3:value>
        </ns3:dateTime>
        <ns3:string name="title">
            <ns3:value>Input 1</ns3:value>
        </ns3:string>
    </ns3:metacard>
    <ns3:metacard ns1:id="00c0eb4ba9b74f8b988ef7060e18a6a7">
        <ns3:type>ddf.metacard</ns3:type>
        <ns3:source>southwestCatalog1</ns3:source>
        <ns3:dateTime name="created">
          <ns3:value>2013-04-10T15:30:05.702-07:00</ns3:value>
        </ns3:dateTime>
        <ns3:string name="title">
            <ns3:value>Input 2</ns3:value>
        </ns3:string>
    </ns3:metacard>
</ns3:metacards>
```

14.4.9. Resource Metacard Transformer

The Resource Metacard Transformer retrieves a resource associated with a metacard.

14.4.9.1. Installing the Resource Metacard Transformer

The Resource Metacard Transformer is installed by default in a standard installation with the Catalog application as the feature `catalog-transformer-resource`.

14.4.9.2. Configuring the Resource Metacard Transformer

The Resource Metacard Transformer has no configurable properties.

14.4.9.3. Using the Resource Metacard Transformer

Endpoints or other components can retrieve an instance of the Resource Metacard Transformer using its `id` resource.

Sample Resource Metacard Transformer Blueprint Reference Snippet

```
<reference id="metacardTransformer" interface="ddf.catalog.transform.MetacardTransformer"
filter="(id=resource)"/>
```

14.4.10. GeoJSON Metacard Transformer

GeoJSON Metacard Transformer translates a metacard into GeoJSON.

14.4.10.1. Installing the GeoJSON Metacard Transformer

The GeoJSON Metacard Transformer is not installed by default with a standard installation.

To install:

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `catalog-transformer-json` feature.

14.4.10.2. Configuring the GeoJSON Metacard Transformer

The GeoJSON Metacard Transformer has no configurable properties.

14.4.10.3. Using the GeoJSON Metacard Transformer

The GeoJSON Metacard Transformer can be used programmatically by requesting a `MetacardTransformer` with the id `geojson`. It can also be used within the REST Endpoint by providing the transform option as `geojson`.

Example REST GET Method with the GeoJSON Metacard Transformer

```
https://{FQDN}:{PORT}/services/catalog/0123456789abcdef0123456789abcdef?transform=geojson
```

Example REST GET Output from the GeoJSON Metacard Transformer

```
{
    "properties":{
        "title":"myTitle",
        "thumbnail":"CA==",
        "resource-uri":"http:\/\/example.com",
        "created":"2012-08-31T23:55:19.518+0000",
        "metadata-content-type-version":"myVersion",
        "metadata-content-type":"myType",
        "metadata":"<xml>text<\/xml>",
        "modified":"2012-08-31T23:55:19.518+0000",
        "metacard-type": "ddf.metacard"
    },
    "type":"Feature",
    "geometry":{
        "type":"LineString",
```

```
        "coordinates":[
            [
                30.0,
                10.0
            ],
            [
                10.0,
                30.0
            ],
            [
                40.0,
                40.0
            ]
        ]
    }
}
```

14.4.11. KML Metacard Transformer

The KML Metacard Transformer is responsible for translating a metacard into a KML-formatted document. The KML contains an HTML description that displays in the pop-up bubble in Google Earth. The HTML contains links to the full metadata view as well as the resource.

14.4.11.1. Installing the KML Metacard Transformer

The KML Metacard Transformer is installed by default with a standard installation in the Spatial Application.

14.4.11.2. Configuring the KML Metacard Transformer

The KML Metacard Transformer has no configurable properties.

14.4.11.3. Using the KML Metacard Transformer

Using the REST Endpoint for example, request a metacard with the transform option set to the KML shortname.

KML Metacard Transformer Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns:ns2="http://www.google.com/kml/ext/2.2" xmlns="http://www.opengis.net/kml/2.2"
xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:ns3=
"http://www.w3.org/2005/Atom">
  <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
    <name>MultiPolygon</name>
```

```
    <description>&lt;!DOCTYPE html&gt;
    &lt;html&gt;
      &lt;head&gt;
        &lt;meta content="text/html; charset=windows-1252" http-equiv="content-type"&gt;
        &lt;style media="screen" type="text/css"&gt;
          .label {
            font-weight: bold
          }
          .linkTable {
width: 100% }
          .thumbnailDiv {
            text-align: center
          }
          img {
            max-width: 100px;
            max-height: 100px;
            border-style:none
          }
    &lt;/style&gt;
  &lt;/head&gt;
  &lt;body&gt;
        &lt;div class="thumbnailDiv"&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48
fab939da528e?transform=resource"&gt;&lt;img alt="Thumnail"
src=" CA=="&gt;&lt;/a&gt;&lt;/div&gt;
    &lt;table&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Source:&lt;/td&gt;
        &lt;td&gt;ddf.distribution&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Created:&lt;/td&gt;
        &lt;td&gt;Wed Oct 30 09:46:29 MDT 2013&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
      &lt;td class="label"&gt;Effective:&lt;/td&gt;
        &lt;td&gt;2014-01-07T14:58:16-0700&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
    &lt;table class="linkTable"&gt;
      &lt;tr&gt;
        &lt;td&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48
fab939da528e?transform=html"&gt;View Details...&lt;/a&gt;&lt;/td&gt;
        &lt;td&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f48
fab939da528e?transform=resource"&gt;Download...&lt;/a&gt;&lt;/td&gt;
      &lt;/tr&gt;
```

```
    &lt;/table&gt;
  &lt;/body&gt;
&lt;/html&gt;
</description>
    <TimeSpan>
      <begin>2014-01-07T21:58:16</begin>
    </TimeSpan>
    <Style id="bluenormal">
      <LabelStyle>
        <scale>0.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>33ff0000</color>
        <width>3.0</width>
      </LineStyle>
      <PolyStyle>
        <color>33ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
<text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;&lt;td
width="400"&gt;$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
      </BalloonStyle>
    </Style>
    <Style id="bluehighlight">
      <LabelStyle>
        <scale>1.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>99ff0000</color>
        <width>6.0</width>
      </LineStyle>
      <PolyStyle>
        <color>99ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
        <text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
&lt;td width="400"&gt;$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
      </BalloonStyle>
    </Style>
    <StyleMap id="default">
      <Pair>
        <key>normal</key>
        <styleUrl>#bluenormal</styleUrl>
      </Pair>
```

```
      <Pair>
        <key>highlight</key>
        <styleUrl>#bluehighlight</styleUrl>
      </Pair>
    </StyleMap>
    <MultiGeometry>
      <Point>
        <coordinates>102.0,2.0</coordinates>
      </Point>
      <MultiGeometry>
        <Polygon>
          <outerBoundaryIs>
            <LinearRing>
              <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0 102.0,2.0</
coordinates>
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
        <Polygon>
100.8,0.2
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2 100.8
,0.8 100.2,0.8 100.2,0.2</coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</MultiGeometry>
</Placemark>
</kml>
```

14.4.12. XML Input Transformer

The XML Input Transformer is responsible for translating an XML document into a Catalog Metacard.

Table 50. XML Input Transformer Usage

Schema	Mime-types
urn:catalog:metacard	`text/xml`

14.4.12.1. Installing the XML Input Transformer

The XML Input Transformer is installed by default with a standard installation in the Catalog application.

14.4.12.2. Configuring the XML Input Transformer

The XML Input Transformer has no configurable properties.

14.4.13. PDF Input Transformer

The PDF Input Transformer is responsible for translating a PDF document into a Catalog Metacard.

Table 51. PDF Input Transformer Usage

Schema	Mime-types
N/A	`application/pdf`

14.4.13.1. Installing the PDF Input Transformer

The PDF Transformer is installed by default with a standard installation in the Catalog application.

14.4.13.2. Configuring the PDF Input Transformer

To configure the PDF Input Transformer:

1. Navigate to the **Catalog** application.
2. Select the **Configuration** tab.
3. Select the **PDF Input Transformer**.

These configurations are available for the PDF Input Transformer:

See PDF Input Transformer configurations for all possible configurations.

14.4.14. CSW Query Response Transformer

The transforms a query response into a CSW-formatted document.

14.4.14.1. Installing the CSW Query Response Transformer

The CSW Query Response Transformer is installed by default with a standard installation in the Spatial application.

14.4.14.2. Configuring the CSW Query Response Transformer

The CSW Query Response Transformer has no configurable properties.

14.4.15. Video Input Transformer

The Video Input transformer Creates Catalog metacards from certain video file types. Currently, it is

handles MPEG-2 transport streams as well as MPEG-4, AVI, MOV, and WMV videos. This input transformer uses Apache Tika to extract basic metadata from the video files and applies more sophisticated methods to extract more meaningful metadata from these types of video.

Table 52. Video Input Transformer Usage

Schema	Mime-types
N/A	• video/avi • video/msvideo • video/vnd.avi • video/x-msvideo • video/mp4 • video/MP2T • video/mpeg • video/quicktime • video/wmv • video/x-ms-wmv

14.4.15.1. Installing the Video Input Transformer

This transformer is installed by default with a standard installation in the Catalog application.

14.4.15.2. Configuring the Video Input Transformer

The Video Input Transformer has no configurable properties.

14.4.16. GeoJSON Input Transformer

The GeoJSON input transformer is responsible for translating GeoJSON into a Catalog metacard.

Table 53. GeoJSON Input Transformer Usage

Schema	Mime-types
N/A	`application/json`

14.4.16.1. Installing the GeoJSON Input Transformer

The GeoJSON Input Transformer is installed by default with a standard installation.

14.4.16.2. Configuring the GeoJSON Input Transformer

The GeoJSON Input Transformer has no configurable properties.

14.4.16.3. Using the GeoJSON Input Transformer

Using the REST Endpoint, for example, HTTP POST a GeoJSON metacard to the Catalog. Once the REST Endpoint receives the GeoJSON Metacard, it is converted to a Catalog metacard.

Example HTTP POST of a Local `metacard.json` File Using the Curl Command

```
curl -X POST -i -H "Content-Type: application/json" -d "@metacard.json"
https://{FQDN}:{PORT}/services/catalog
```

14.4.16.4. Conversion to a Metacard

A GeoJSON object consists of a single JSON object. This can be a geometry, a feature, or a `FeatureCollection`. The GeoJSON input transformer only converts "feature" objects into metacards because feature objects include geometry information and a list of properties. A geometry object alone does not contain enough information to create a metacard. Additionally, the input transformer currently does not handle `FeatureCollection`s.

IMPORTANT

Cannot create Metacard from this limited GeoJSON

```
{ "type": "LineString",
 "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
 }
```

The following sample *does* create a valid metacard:

Sample Parseable GeoJson (Point)

```
{
    "properties": {
        "title": "myTitle",
        "thumbnail": "CA==",
        "resource-uri": "http://example.com",
        "created": "2012-09-01T00:09:19.368+0000",
        "metadata-content-type-version": "myVersion",
        "metadata-content-type": "myType",
        "metadata": "<xml></xml>",
        "modified": "2012-09-01T00:09:19.368+0000"
    },
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [
            30.0,
            10.0
        ]
```

```
    }
}
```

In the current implementation, `Metacard.LOCATION` is not taken from the properties list as WKT, but instead interpreted from the `geometry` JSON object. The geometry object is formatted according to the GeoJSON standard. Dates are in the ISO 8601 standard. White space is ignored, as in most cases with JSON. Binary data is accepted as Base64. XML must be properly escaped, such as what is proper for normal JSON.

Currently, only **Required Attributes** are recognized in the properties.

14.4.16.4.1. Metacard Extensibility

GeoJSON supports custom, extensible properties on the incoming GeoJSON using DDF's extensible metacard support. To have those customized attributes understood by the system, a corresponding `MetacardType` must be registered with the `MetacardTypeRegistry`. That `MetacardType` must be specified by name in the metacard-type property of the incoming GeoJSON. If a `MetacardType` is specified on the GeoJSON input, the customized properties can be processed, cataloged, and indexed.

Sample GeoJSON input

```
{
    "properties": {
        "title": "myTitle",
        "thumbnail": "CA==",
        "resource-uri": "http://example.com",
        "created": "2012-09-01T00:09:19.368+0000",
        "metadata-content-type-version": "myVersion",
        "metadata-content-type": "myType",
        "metadata": "<xml></xml>",
        "modified": "2012-09-01T00:09:19.368+0000",
        "min-frequency": "10000000",
        "max-frequency": "20000000",
        "metacard-type": "ddf.metacard.custom.type"
 },
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [
            30.0,
            10.0
        ]
    }
}
```

When the GeoJSON Input Transformer gets GeoJSON with the `MetacardType` specified, it performs a

lookup in the `MetacardTypeRegistry` to obtain the specified `MetacardType` in order to understand how to parse the GeoJSON. If no `MetacardType` is specified, the GeoJSON Input Transformer assumes the default `MetacardType`. If an unregistered `MetacardType` is specified, an exception is returned to the client indicating that the `MetacardType` was not found.

14.4.16.5. Usage Limitations of the GeoJSON Input Transformer

The GeoJSON Input Transformer does not handle multiple geometries.

14.4.17. Query Response Transformer Consumer

The Query Response Transformer Consumer is responsible for translating a query response into a Catalog Metacard.

14.4.17.1. Installing the Query Response Transformer Consumer

The Query Response Transformer Consumer is installed by default with a standard installation in the Catalog application.

14.4.17.2. Configuring the Query Response Transformer Consumer

The Query Response Transformer Consumer has no configurable properties.

14.4.18. KML Query Response Transformer

The KML Query Response Transformer translates a query response into a KML-formatted document. The KML contains an HTML description for each metacard that displays in the pop-up bubble in Google Earth. The HTML contains links to the full metadata view as well as the resource.

14.4.18.1. Installing the KML Query Response Transformer

The `spatial-kml-transformer` feature is installed by default in the Spatial Application.

14.4.18.2. Configuring the KML Query Response Transformer

The KML Query Response Transformer has no configurable properties.

14.4.18.3. Using the KML Query Response Transformer

Using the OpenSearch Endpoint, for example, query with the format option set to the KML shortname: `kml`.

KML Query Response Transformer URL

```
http://{FQDN}:{PORT}/services/catalog/query?q=schematypesearch&format=kml
```

KML Query Response Transformer Example Output

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns:ns2="http://www.google.com/kml/ext/2.2" xmlns="http://www.opengis.net/kml/2.2"
xmlns:ns4="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:ns3=
"http://www.w3.org/2005/Atom">
  <Document id="f0884d8c-cf9b-44a1-bb5a-d3c6fb9a96b6">
    <name>Results (1)</name>
    <open xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">false</open>
    <Style id="bluenormal">
      <LabelStyle>
        <scale>0.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>33ff0000</color>
        <width>3.0</width>
      </LineStyle>
      <PolyStyle>
        <color>33ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
        <text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
&lt;td width="400"&gt;$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
      </BalloonStyle>
    </Style>
    <Style id="bluehighlight">
      <LabelStyle>
        <scale>1.0</scale>
      </LabelStyle>
      <LineStyle>
        <color>99ff0000</color>
        <width>6.0</width>
      </LineStyle>
      <PolyStyle>
        <color>99ff0000</color>
        <fill xsi:type="xs:boolean" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">true</fill>
      </PolyStyle>
      <BalloonStyle>
        <text>&lt;h3&gt;&lt;b&gt;$[name]&lt;/b&gt;&lt;/h3&gt;&lt;table&gt;&lt;tr&gt;
&lt;td width="400"&gt;$[description]&lt;/td&gt;&lt;/tr&gt;&lt;/table&gt;</text>
      </BalloonStyle>
    </Style>
    <StyleMap id="default">
      <Pair>
```

```
        <key>normal</key>
        <styleUrl>#bluenormal</styleUrl>
      </Pair>
      <Pair>
        <key>highlight</key>
        <styleUrl>#bluehighlight</styleUrl>
      </Pair>
    </StyleMap>
    <Placemark id="Placemark-0103c77e66d9428d8f48fab939da528e">
      <name>MultiPolygon</name>
      <description>&lt;!DOCTYPE html&gt;
&lt;html&gt;
  &lt;head&gt;
    &lt;meta content="text/html; charset=windows-1252" http-equiv="content-type"&gt;
    &lt;style media="screen" type="text/css"&gt;
      .label {
        font-weight: bold
      }
      .linkTable {
width: 100% }
      .thumbnailDiv {
        text-align: center
} img {
        max-width: 100px;
        max-height: 100px;
        border-style:none
      }
    &lt;/style&gt;
  &lt;/head&gt;
  &lt;body&gt;
        &lt;div class="thumbnailDiv"&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f
48fab939da528e?transform=resource"&gt;&lt;img alt="Thumnail"
src=" CA=="&gt;&lt;/a&gt;&lt;/div&gt;
    &lt;table&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Source:&lt;/td&gt;
        &lt;td&gt;ddf.distribution&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Created:&lt;/td&gt;
        &lt;td&gt;Wed Oct 30 09:46:29 MDT 2013&lt;/td&gt;
      &lt;/tr&gt;
      &lt;tr&gt;
        &lt;td class="label"&gt;Effective:&lt;/td&gt;
        &lt;td&gt;2014-01-07T14:48:47-0700&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
```

```
    &lt;table class="linkTable"&gt;
      &lt;tr&gt;
        &lt;td&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f
48fab939da528e?transform=html"&gt;View Details...&lt;/a&gt;&lt;/td&gt;
        &lt;td&gt;&lt;a
href="http://{FQDN}:{PORT}/services/catalog/sources/ddf.distribution/0103c77e66d9428d8f
48fab939da528e?transform=resource"&gt;Download...&lt;/a&gt;&lt;/td&gt;
      &lt;/tr&gt;
    &lt;/table&gt;
  &lt;/body&gt;
&lt;/html&gt;
</description>
      <TimeSpan>
        <begin>2014-01-07T21:48:47</begin>
      </TimeSpan>
      <styleUrl>#default</styleUrl>
      <MultiGeometry>
        <Point>
          <coordinates>102.0,2.0</coordinates>
        </Point>
        <MultiGeometry>
          <Polygon>
            <outerBoundaryIs>
              <LinearRing>
                <coordinates>102.0,2.0 103.0,2.0 103.0,3.0 102.0,3.0
102.0,2.0</coordinates>
              </LinearRing>
100.8,0.2
  </outerBoundaryIs>
</Polygon>
<Polygon>
  <outerBoundaryIs>
    <LinearRing>
      <coordinates>100.0,0.0 101.0,0.0 101.0,1.0 100.0,1.0 100.0,0.0 100.2,0.2
        100.8,0.8 100.2,0.8 100.2,0.2</coordinates>
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
      </MultiGeometry>
    </MultiGeometry>
  </Placemark>
</Document>
</kml>
```

14.4.19. Tika Input Transformer

The Tika Input Transformer is the default input transformer responsible for translating Microsoft Word, Microsoft Excel, Microsoft PowerPoint, OpenOffice Writer, and PDF documents into Catalog records. This input transformer utilizes Apache Tika to provide basic support for these mime types. The metadata common to all these document types, such as creation date, author, last modified date, etc., is extracted and used to create the catalog record. The Tika Input Transformer's main purpose is to ingest these types of content into the Metadata Catalog.

The Tika input transformer is most basic input transformer and the last to be invoked. This allows any registered input transformers that are more specific to a document type to be invoked instead of this rudimentary input transformer.

Table 54. Tika Input Transformer Usage

Schema	Mime-types
N/A	This basic transformer can ingest many file types. See All Formats Supported.

14.4.19.1. Installing the Tika Input Transformer

This transformer is installed by default with a standard installation in the Catalog.

14.4.19.2. Configuring the Tika Input Transformer

The properties below describe how to configure the Tika input transformer.

See Tika Input Transformer configurations for all possible configurations.

14.5. Mime Type Mapper

The Mime Type Mapper is the entry point in DDF for resolving file extensions to mime types, and vice versa.

`MimeTypeMappers` are used by the `ResourceReader` to determine the file extension for a given mime type in aid of retrieving a resource. `MimeTypeMappers` are also used by the `FileSystemProvider` in the Catalog Framework to read a file from the content file repository.

The `MimeTypeMapper` maintains a list of all of the `MimeTypeResolvers` in DDF.

The `MimeTypeMapper` accesses each `MimeTypeResolver` according to its priority until the provided file extension is successfully mapped to its corresponding mime type. If no mapping is found for the file extension, `null` is returned for the mime type. Similarly, the `MimeTypeMapper` accesses each `MimeTypeResolver` according to its priority until the provided mime type is successfully mapped to its corresponding file extension. If no mapping is found for the mime type, `null` is returned for the file extension.

For files with no file extension, the MimeTypeMapper attempts to determine the mime type from the contents of the file. If it is unsuccessful, the file is ingested as a binary file.

DDF Mime Type Mapper

Core implementation of the DDF Mime API.

14.5.1. DDF Mime Type Mapper

The DDF Mime Type Mapper is the core implementation of the DDF Mime API. It provides access to all `MimeTypeResolvers` within DDF, which provide mapping of mime types to file extensions and file extensions to mime types.

14.5.1.1. Installing the DDF Mime Type Mapper

The DDF Mime Type Mapper is installed by default with a standard installation in the Platform application.

14.5.1.2. Configuring DDF Mime Type Mapper

The DDF Mime Type Mapper has no configurable properties.

14.6. Mime Type Resolver

A Mime Type Resolver is a DDF service that can map a file extension to its corresponding mime type and, conversely, can map a mime type to its file extension.

`MimeTypeResolvers` are assigned a priority (0-100, with the higher the number indicating the higher priority). This priority is used to sort all of the `MimeTypeResolvers` in the order they should be checked to map a file extension to a mime type (or vice versa). This priority also allows custom `MimeTypeResolvers` to be invoked before default `MimeTypeResolvers` by setting custom resolver's priority higher than the default.

`MimeTypeResolvers` are not typically invoked directly. Rather, the `MimeTypeMapper` maintains a list of `MimeTypeResolvers` (sorted by their priority) that it invokes to resolve a mime type to its file extension (or to resolve a file extension to its mime type).

Tika Mime Type Resolver

Provides support for resolving over 1300 mime types.

Custom Mime Type Resolver

The Custom Mime Type Resolver is a `MimeTypeResolver` that defines the custom mime types that DDF will support.

14.6.1. Tika Mime Type Resolver

The `TikaMimeTypeResolver` is a `MimeTypeResolver` that is implemented using the Apache Tika open source product.

Using the Apache Tika content analysis toolkit, the `TikaMimeTypeResolver` provides support for resolving over 1300 mime types, but not all mime types yield the same quality metadata.

The `TikaMimeTypeResolver` is assigned a default priority of `-1` to insure that it is always invoked last by the `MimeTypeMapper`. This insures that any custom `MimeTypeResolvers` that may be installed is invoked before the `TikaMimeTypeResolver`.

The `TikaMimeTypeResolver` provides the bulk of the default mime type support for DDF.

14.6.1.1. Installing the Tika Mime Type Resolver

The `TikaMimeTypeResolver` is bundled as the `mime-tika-resolver` feature in the `mime-tika-app` application.

This feature is installed by default.

14.6.1.2. Configuring the Tika Mime Type Resolver

The Tika Mime Type Resolver has no configurable properties.

14.6.2. Custom Mime Type Resolver

These are mime types not supported by the default `TikaMimeTypeResolver`.

Table 55. Custom Mime Type Resolver Default Supported Mime Types

File Extension	Mime Type
`nitf`	`image/nitf`
`ntf`	`image/nitf`
`json`	`json=application/json;id=geojson`

As a `MimeTypeResolver`, the Custom Mime Type Resolver provides methods to map the file extension to the corresponding mime type, and vice versa.

14.6.2.1. Installing the Custom Mime Type Resolver

One Custom Mime Type Resolver is configured and installed for the `image/nitf` mime type. This custom resolver is bundled in the `mime-core-app` application and is part of the `mime-core` feature.

Additional Custom Mime Type Resolvers can be added for other custom mime types.

14.6.2.2. Configuring the Custom Mime Type Resolver

The configurable properties for the Custom Mime Type Resolver are accessed from the **MIME Custom Types** configuration in the Admin Console.

- Navigate to the Admin Console.
- Select the **Platform** application.
- Select **Configuration**.
- Select **MIME Custom Types**.

Managed Service Factory PID

- `Ddf_Custom_Mime_Type_Resolver`

See Custom Mime Type Resolver configurations for all possible configurations.

15. Catalog Plugins

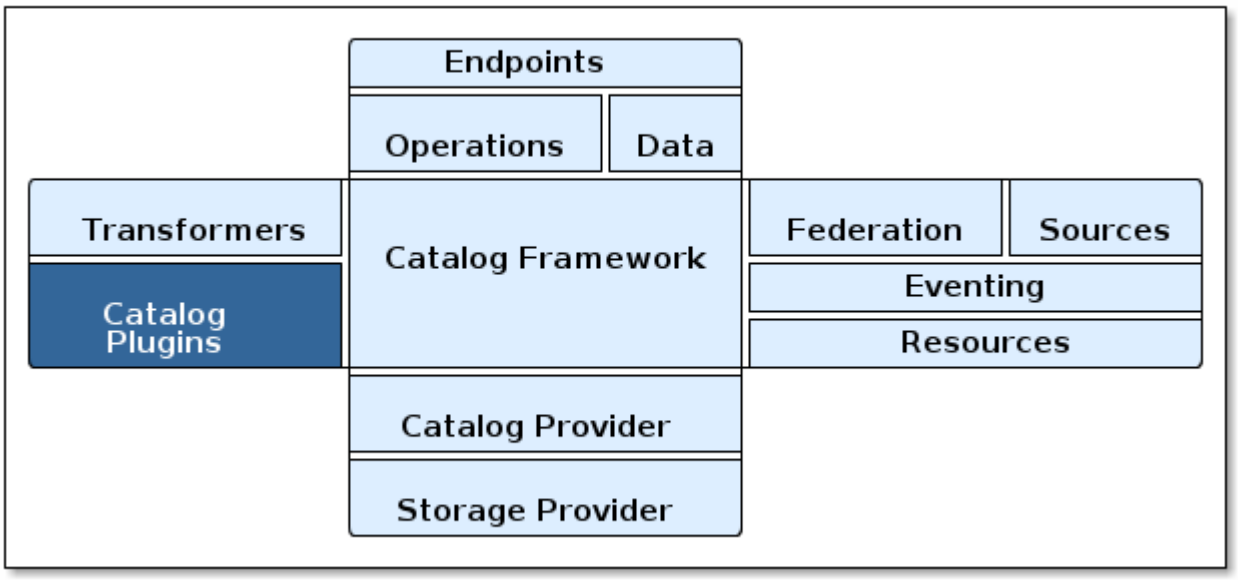


Figure 15. Catalog Architecture Catalog Plugins

Catalog Plugins are additional tools to use to add additional business logic at certain points, depending on the type of plugin.

The Catalog Framework calls Catalog Plugins to process requests and responses as they enter and leave the Framework.

15.1. Types of Plugins

Plugins can be designed to run before or after certain processes. They are often used for validation, optimization, or logging. Many plugins are designed to be called at more than one time. See Catalog Plugin Compatibility.

Pre-Authorization Plugins

Perform any changes needed before security rules are applied.

Policy Plugins

Allows or denies access to the Catalog operation or response.

Access Plugins

Used to build policy information for requests.

Pre-Ingest Plugins

Perform any changes to a metacard prior to ingest.

Post-Ingest Plugins

Perform actions after ingest is completed.

Post-Process Plugins

Performs additional processing after ingest.

Pre-Query Plugins

Perform any changes to a query before execution.

Pre-Federated-Query Plugins

Perform any changes to a federated query before execution.

Post-Query Plugins

Perform any changes to a response after query completes.

Post-Federated-Query Plugins

Perform any changes to a response after federated query completes.

Pre-Resource Plugins

Perform any changes to a request associated with a metacard prior to download.

Post-Resource Plugins

Perform any changes to a resource after download.

Pre-Create Storage Plugins

Perform any changes before creating a resource.

Post-Create Storage Plugins

Perform any changes after creating a resource.

Pre-Update Storage Plugins

Perform any changes before updating a resource.

Post-Update Storage Plugins

Perform any changes after updating a resource.

Pre-Subscription Plugins

Perform any changes before creating a subscription.

Pre-Delivery Plugins

Perform any changes before delivering a subscribed event.

Plugins are called in a specific order during different operations. Custom Plugins can be added to the chain for special use cases.



Figure 16. Query Request Plugin Call Order

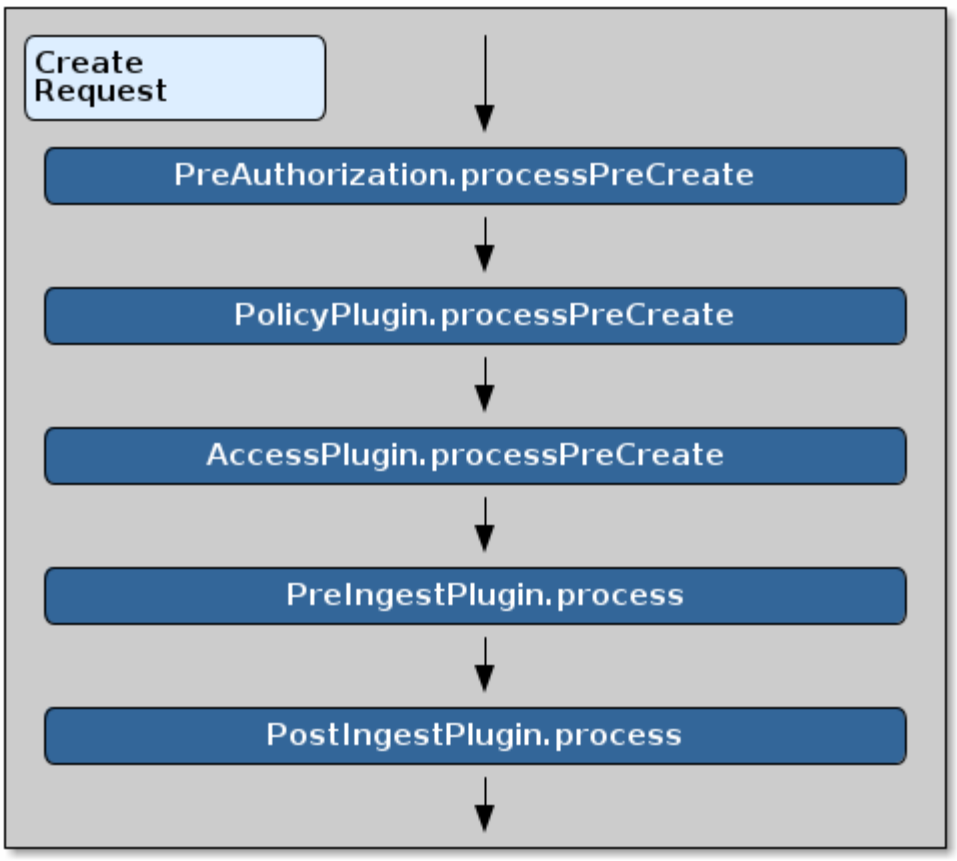


Figure 17. Create Request Plugin Call Order

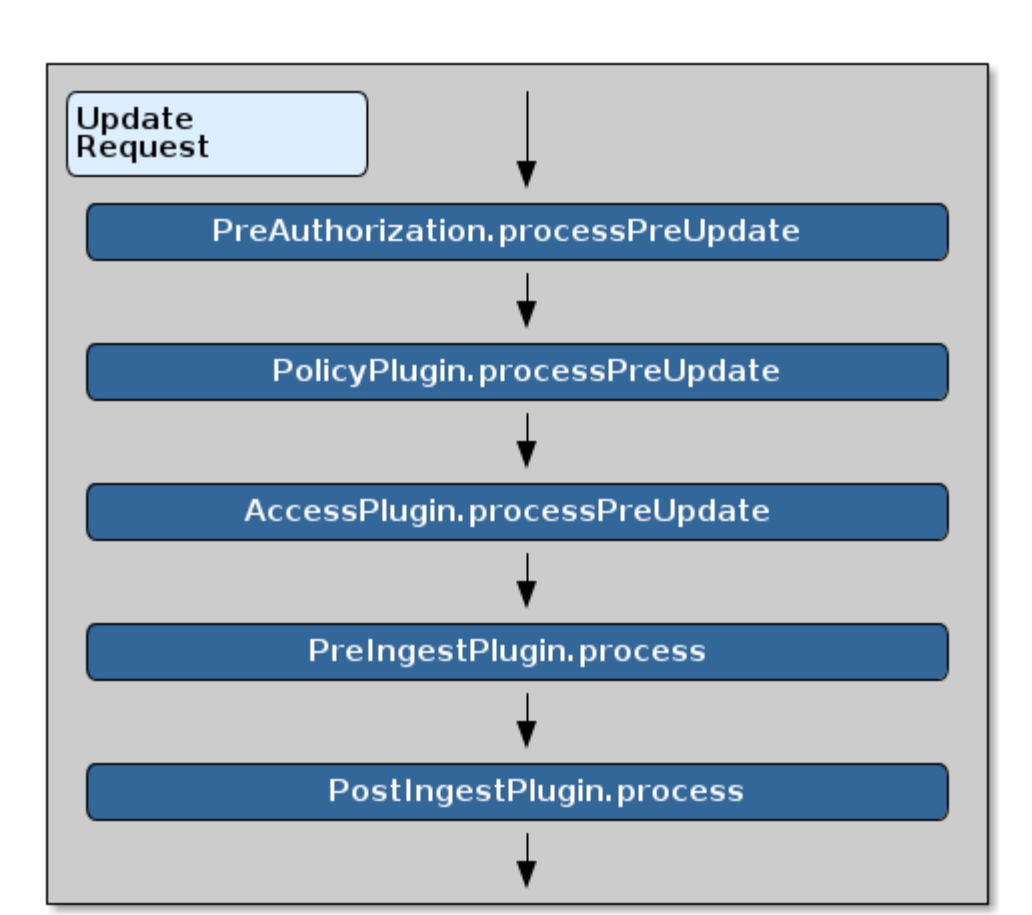


Figure 18. Update Request Plugin Call Order

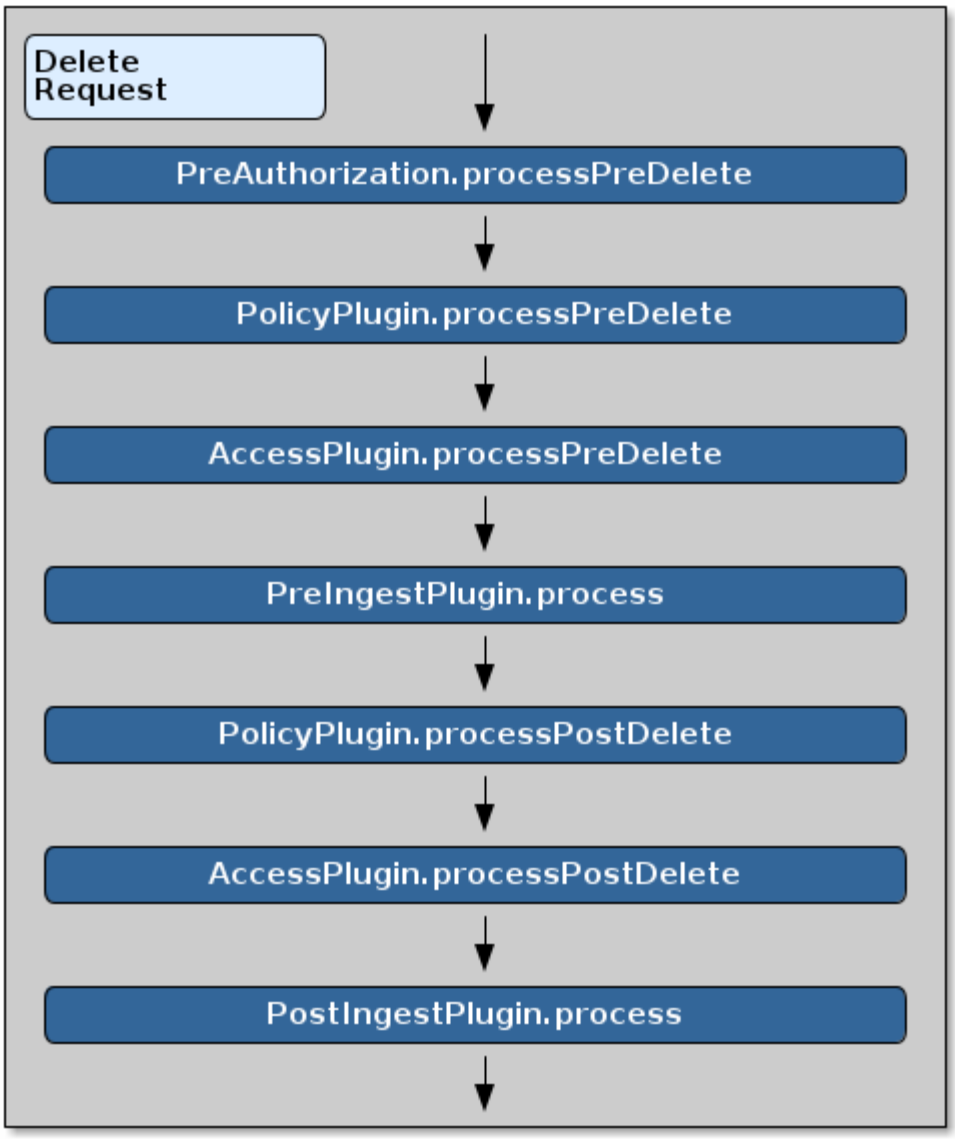


Figure 19. Delete Request Plugin Call Order



Figure 20. Resource Request Plugin Call Order

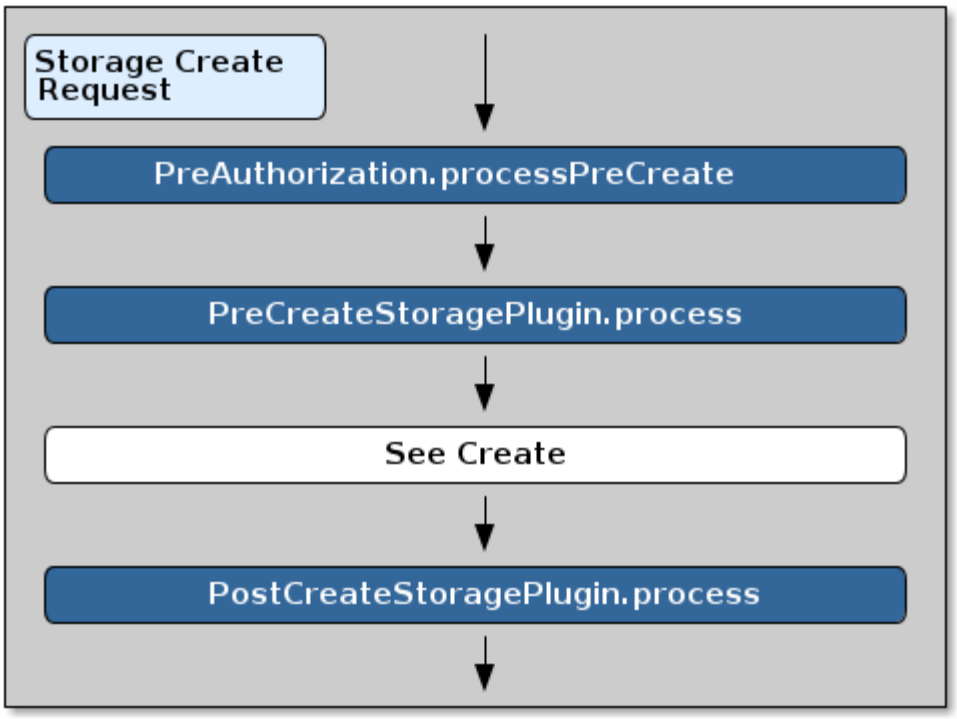


Figure 21. Storage Create Request Plugin Call Order

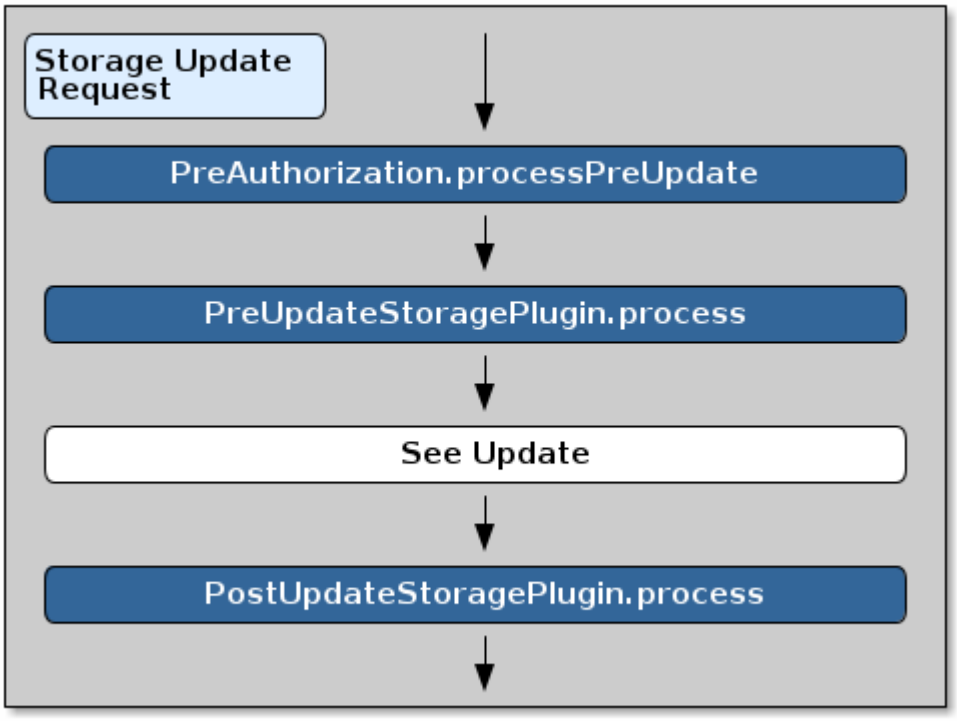


Figure 22. Storage Update Request Plugin Call Order

Table 56. Catalog Plugin Compatibility

Plugin	Pre-Authorization Plugins	Policy Plugins	Access Plugins	Pre-Ingest Plugins	Post-Ingest Plugins	Pre-Query Plugins	Post-Query Plugins	Post-Process Plugins
Event Processor					x			
Processing Post-Ingest Plugin					x			
Catalog Backup Plugin					x			
Security Audit Plugin			x					
Point of Contact Policy Plugin		x						
Metacard Validity Filter Plugin		x						
XML Attribute Security Policy Plugin		x						
Catalog Policy Plugin		x						
Security Plugin			x					
Metacard Resource Size Plugin							x	
Metacard Groomer				x				

Plugin	Pre-Authorization Plugins	Policy Plugins	Access Plugins	Pre-Ingest Plugins	Post-Ingest Plugins	Pre-Query Plugins	Post-Query Plugins	Post-Process Plugins
Workspace Access Plugin			x					
Client Info Plugin	x							
Resource URI Policy Plugin		x						
Metacard Ingest Network Plugin	x							
Workspace Sharing Policy Plugin		x						
Metacard Validity Marker				x				
Filter Plugin			x					
Metacard Backup S3 Storage Provider					x			
GeoCoder Plugin				x				
JPEG2000 Thumbnail Converter							x	
Content URI Access Plugin			x					

Plugin	Pre-Authorization Plugins	Policy Plugins	Access Plugins	Pre-Ingest Plugins	Post-Ingest Plugins	Pre-Query Plugins	Post-Query Plugins	Post-Process Plugins
Operation Plugin			x					
Workspace Pre-Ingest Plugin				x				
Expiration Date Pre-Ingest Plugin				x				
Security Logging Plugin				x	x	x	x	
Historian Policy Plugin		x						
Metacard Backup File Storage Provider					x			
Metacard Attribute Security Policy Plugin		x						

Table 57. Catalog Plugin Compatibility, Cont.

Plugin	Pre-Federated-Query Plugins	Post-Federated-Query Plugins	Pre-Resource Plugins	Post-Resource Plugins	Pre-Create Storage Plugins	Post-Create Storage Plugins	Pre-Update Storage Plugins	Post-Update Storage Plugins	Pre-Subscription Plugins	Pre-Delivery Plugins
Checksum Plugin					x		x			

Plugin	Pre-Federated-Query Plugins	Post-Federated-Query Plugins	Pre-Resource Plugins	Post-Resource Plugins	Pre-Create Storage Plugins	Post-Create Storage Plugins	Pre-Update Storage Plugins	Post-Update Storage Plugins	Pre-Subscription Plugins	Pre-Delivery Plugins
Video Thumbnail Plugin						x		x		
Security Logging Plugin	x!	x	x	x	x	x	x	x		

15.1.1. Pre-Authorization Plugins

Pre-delivery plugins are invoked before any security rules are applied. This is an opportunity to take any action before authorization, including but not limited to:

- logging.
- adding network-specific information.
- adding user-identifying information.

15.1.1.1. Available Pre-Authorization Plugins

Client Info Plugin

Injects request-specific network information into a request.

Metacard Ingest Network Plugin

Adds attributes for network info from ingest request.

15.1.2. Policy Plugins

Policy plugins are invoked to set up the policy for a request/response. This provides an opportunity to attach custom requirements on operations or individual metacards. All the 'requirements' from each Policy plugin are combined into a single policy that is included in the request/response. Access plugins are used to act on this combined policy.

15.1.2.1. Available Policy Plugins

Catalog Policy Plugin

Configures user attributes required for catalog operations.

Historian Policy Plugin

Protects metacard history from being edited by users without the history role.

Metacard Attribute Security Policy Plugin

Collects attributes into a security field for the metacard.

Metacard Validity Filter Plugin

Determines whether to filter metacards with validation errors or warnings.

Point of Contact Policy Plugin

Adds a policy if Point of Contact is updated.

Resource URI Policy Plugin

Configures required user attributes for setting or altering a resource URI.

Workspace Sharing Policy Plugin

Collects attributes for a workspace to identify the appropriate policy to allow sharing.

XML Attribute Security Policy Plugin

Finds security attributes contained in a metacard's metadata.

15.1.3. Access Plugins

Access plugins are invoked directly after the Policy plugins have been successfully executed. This is an opportunity to either stop processing or modify the request/response based on policy information.

15.1.3.1. Available Access Plugins

Content URI Access Plugin

Prevents a Metacard's resource URI from being overridden by an incoming UpdateRequest.

Filter Plugin

Performs filtering on query responses as they pass through the framework.

Operation Plugin

Validates a user or subject's security attributes.

Security Audit Plugin

Audits specific metacard attributes.

Security Plugin

Identifies the subject for an operation.

Workspace Access Plugin

Prevents non-owner users from changing workspace permissions.

15.1.4. Pre-Ingest Plugins

Pre-ingest plugins are invoked before an ingest operation is sent to the catalog. They are not run on a query. This is an opportunity to take any action on the ingest request, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

15.1.4.1. Available Pre-Ingest Plugins

Expiration Date Pre-Ingest Plugin

Adds or updates expiration dates for the resource.

GeoCoder Plugin

Populates the `Location.COUNTRY_CODE` attribute if the Metacard has an associated location.

Metacard Groomer

Modifies metacards when created or updated.

Metacard Validity Marker

Modifies metacards when created or ingested according to metacard validator services.

Security Logging Plugin

Logs operations to the security log.

Workspace Pre-Ingest Plugin

Verifies that a workspace has an associated email to enable sharing.

15.1.5. Post-Ingest Plugins

Post-ingest plugins are invoked after data has been created, updated, or deleted in a Catalog Provider.

15.1.5.1. Available Post-Ingest Plugins

Catalog Backup Plugin

Enables backup of the catalog and its metacards.

Event Processor

Creates, updates, and deletes subscriptions.

Metacard Backup File Storage Provider

Stores backed-up metacards.

Metacard Backup S3 Storage Provider

Stores backed-up metacards in a specified S3 bucket and key.

Processing Post-Ingest Plugin

Submits catalog Create, Update, or Delete requests to the Processing Framework.

Security Logging Plugin

Logs operations to the security log.

15.1.6. Post-Process Plugins

NOTE	This code is experimental. While this interface is functional and tested, it may change or be removed in a future version of the library.

Post-Process Plugins are invoked after a metacard has been created, updated, or deleted and committed to the Catalog. They are the last plugins to run and are triggered by a **Post-Ingest Plugin**. Post-Process plugins are well-suited for asynchronous tasks. See the Asynchronous Processing Framework for more information about how **Post-Process Plugins** are used.

15.1.6.1. Available Post-Process Plugins

None.

15.1.7. Pre-Query Plugins

Pre-query plugins are invoked before a query operation is sent to any of the Sources. This is an opportunity to take any action on the query, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

15.1.7.1. Available Pre-Query Plugins

Security Logging Plugin

Logs operations to the security log.

15.1.8. Pre-Federated-Query Plugins

Pre-federated-query plugins are invoked before a federated query operation is sent to any of the Sources. This is an opportunity to take any action on the query, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

15.1.8.1. Available Pre-Federated-Query Plugins

Security Logging Plugin

Logs operations to the security log.

Tags Filter Plugin

Updates queries without filters.

15.1.9. Post-Query Plugins

Post-query plugins are invoked after a query has been executed successfully, but before the response is returned to the endpoint. This is an opportunity to take any action on the query response, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.
- deduplication.

15.1.9.1. Available Post-Query Plugins

JPEG2000 Thumbnail Converter

Creates thumbnails for jpeg2000 images.

Metacard Resource Size Plugin

Updates the resource size attribute of a metacard.

Security Logging Plugin

Logs operations to the security log.

15.1.10. Post-Federated-Query Plugins

Post-federated-query plugins are invoked after a federated query has been executed successfully, but before the response is returned to the endpoint. This is an opportunity to take any action on the query response, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.
- deduplication.

15.1.10.1. Available Post-Federated-Query Plugins

Security Logging Plugin

Logs operations to the security log.

15.1.11. Pre-Resource Plugins

Pre-Resource plugins are invoked before a request to retrieve a resource is sent to a Source. This is an opportunity to take any action on the request, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

15.1.11.1. Available Pre-Resource Plugins

Security Logging Plugin

Logs operations to the security log.

15.1.12. Post-Resource Plugins

Post-resource plugins are invoked after a resource has been retrieved, but before it is returned to the endpoint. This is an opportunity to take any action on the response, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.

15.1.12.1. Available Post-Resource Plugins

Security Logging Plugin

Logs operations to the security log.

15.1.13. Pre-Create Storage Plugins

Pre-Create storage plugins are invoked immediately before an item is created in the content repository.

15.1.13.1. Available Pre-Create Storage Plugins

Checksum Plugin

Creates a unique checksum for ingested resources.

Security Logging Plugin

Logs operations to the security log.

15.1.14. Post-Create Storage Plugins

Post-Create storage plugins are invoked immediately after an item is created in the content repository.

15.1.14.1. Available Post-Create Storage Plugins

Security Logging Plugin

Logs operations to the security log.

Video Thumbnail Plugin

Generates thumbnails for video files.

15.1.15. Pre-Update Storage Plugins

Pre-Update storage plugins are invoked immediately before an item is updated in the content repository.

15.1.15.1. Available Pre-Update Storage Plugins

Checksum Plugin

Creates a unique checksum for ingested resources.

Security Logging Plugin

Logs operations to the security log.

15.1.16. Post-Update Storage Plugins

Post-Update storage plugins are invoked immediately after an item is updated in the content repository.

15.1.16.1. Available Post-Update Storage Plugins

Security Logging Plugin

Logs operations to the security log.

Video Thumbnail Plugin

Generates thumbnails for video files.

15.1.17. Pre-Subscription Plugins

Pre-subscription plugins are invoked before a Subscription is activated by an Event Processor. This is an opportunity to take any action on the Subscription, including but not limited to:

- validation.
- logging.
- auditing.
- optimization.
- security filtering.

15.1.17.1. Available Pre-Subscription Plugins

None.

15.1.18. Pre-Delivery Plugins

Pre-delivery plugins are invoked before a Delivery Method is invoked on a Subscription. This is an opportunity to take any action before event delivery, including but not limited to:

- logging.
- auditing.
- security filtering/redaction.

15.1.18.1. Available Pre-Delivery Plugins

None.

15.2. Catalog Plugin Details

Installation and configuration details listed by plugin name.

15.2.1. Catalog Backup Plugin

The Catalog Backup Plugin is used to enable data backup of the catalog and the metacards it contains.

WARNING	*Catalog Backup Plugin Considerations* Using this plugin may impact performance negatively.

15.2.1.1. Installing the Catalog Backup Plugin

The Catalog Backup Plugin is installed by default with a standard installation in the Catalog application.

15.2.1.2. Configuring the Catalog Backup Plugin

To configure the Catalog Backup Plugin:

1. Navigate to the **Admin Console**.
2. Select **Catalog** application.
3. Select **Configuration** tab.
4. Select **Backup Post-Ingest Plugin**.

See Catalog Backup Plugin configurations for all possible configurations.

15.2.1.3. Usage Limitations of the Catalog Backup Plugin

- May affect performance.
- Must be installed prior to ingesting any content.
- Once enabled, disabling *may* cause incomplete backups.

15.2.2. Catalog Policy Plugin

The Catalog Policy Plugin configures the attributes required for users to perform Create, Read, Update, and Delete operations on the catalog.

15.2.2.1. Installing the Catalog Policy Plugin

The Catalog Policy Plugin is installed by default with a standard installation in the Catalog application.

15.2.2.2. Configuring the Catalog Policy Plugin

To configure the Catalog Policy Plugin:

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

See Catalog Policy Plugin configurations for all possible configurations.

15.2.3. Checksum Plugin

The Checksum plugin creates a unique checksum for resources input into the system to identify updated content.

15.2.3.1. Installing the Checksum Plugin

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

15.2.3.2. Configuring the Checksum Plugin

The Checksum Plugin has no configurable properties.

15.2.4. Client Info Plugin

The Client Info plugin injects request-specific network information into request properties, such as Remote IP Address, Remote Host Name, Servlet Scheme, and Servlet Context.

15.2.4.1. Related Components to the Client Info Plugin

- Client info filter
- Metacard Ingest Network Plugin

15.2.4.2. Installing the Client Info Plugin

The Client Info Plugin is installed by default with a standard installation in the Catalog application.

15.2.4.3. Configuring the Client Info Plugin

The Client Info Plugin has no configurable properties.

15.2.5. Content URI Access Plugin

The prevents a Metacard's resource URI from being overridden by an incoming UpdateRequest.

15.2.5.1. Installing the Content URI Access Plugin

The Content URI Access Plugin is installed by default with a standard installation in the Catalog application.

15.2.5.2. Configuring the Content URI Access Plugin

The Content URI Access Plugin has no configurable properties.

15.2.6. Event Processor

The Event Processor creates, updates, and deletes subscriptions for event notification. These subscriptions optionally specify a filter criteria so that only events of interest to the subscriber are posted for notification.

As metacards are created, updated, and deleted, the Catalog's Event Processor is invoked (as a post-ingest plugin) for each of these events. The Event Processor applies the filter criteria for each registered subscription to each of these ingest events to determine if they match the criteria.

For more information on creating subscriptions, see Creating a Subscription.

15.2.6.1. Installing the Event Processor

The Event Processor is installed by default with a standard installation in the Catalog application.

15.2.6.2. Configuring the Event Processor

The Event Processor has no configurable properties.

15.2.6.3. Usage Limitations of the Event Processor

The Standard Event processor currently broadcasts federated events and should not. It should only broadcast events that were generated locally, all other events should be dropped. See DDF-3151 for status.

15.2.7. Expiration Date Pre-Ingest Plugin

The Expiration Date plugin adds or updates expiration dates which can be used later for archiving old data.

15.2.7.1. Installing the Expiration Date Pre-Ingest Plugin

The Expiration Date Pre-Ingest Plugin is not installed by default with a standard installation. To install:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **Expiration Data Pre-Ingest Plugin**.

15.2.7.2. Configuring the Expiration Date Pre-Ingest Plugin

To configure the Expiration Date Pre-Ingest Plugin:

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **Expiration Date Pre-Ingest Plugin**.

See Expiration Date Plugin configurations for all possible configurations.

15.2.8. Filter Plugin

The Filter Plugin performs filtering on query responses as they pass through the framework.

Each metacard result can contain security attributes that are pulled from the metadata record after being processed by a `PolicyPlugin` that populates this attribute. The security attribute is a Map containing a set of keys that map to lists of values. The metacard is then processed by a filter plugin that creates a `KeyValueCollectionPermission` from the metacard's security attribute. This permission is then checked against the user subject to determine if the subject has the correct claims to view that metacard. The decision to filter the metacard eventually relies on the installed Policy Decision Point (PDP). The PDP that is being used returns a decision, and the metacard is either filtered or allowed to pass through.

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

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

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

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

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

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

In the above example, the user's claims are represented very simply and are similar to how they would actually appear in a SAML 2 assertion. Each of these user (or subject) claims is converted to a `KeyValuePermission` object. These permission objects are implied against the permission object generated from the metacard record. In this particular case, the metacard might be allowed if the policy is configured appropriately because all of the permissions line up correctly.

15.2.8.1. Installing the Filter Plugin

The Filter Plugin is installed by default with a standard installation in the Catalog application.

15.2.8.2. Configuring the Filter Plugin

The Filter Plugin has no configurable properties.

15.2.9. GeoCoder Plugin

The GeoCoder plugin is a pre-ingest plugin that is responsible for populating the Metacard's `Location.COUNTRY_CODE` attribute if the Metacard has an associated location. If there is a valid country code for the Metacard, it is in ISO 3166-1 alpha-3 format. If the metacard's country code is already populated, the plugin does **not** override it. The GeoCoder relies on either the WebService or Offline Gazetteer to retrieve country code information.

WARNING	For a polygon or polygons, this plugin takes the center point of the bounding box

> to assign the country code.

15.2.9.1. Installing the GeoCoder Plugin

The GeoCoder Plugin is installed by default with the Spatial application, when the WebService or Offline Gazetteer is started.

15.2.9.2. Configuring the GeoCoder Plugin

To configure the GeoCoder Plugin:

1. Navigate to the **Admin Console**.
2. Select **Spatial** application.
3. Select **Configuration** tab.
4. Select **GeoCoder Plugin**.

These are the available configurations:

See GeoCoder Plugin configurations for all possible configurations.

15.2.10. Historian Policy Plugin

The Historian Policy Plugin protects metacard history from being edited or deleted by users without the history role (a `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role` of `system-history`).

15.2.10.1. Installing the Historian Policy Plugin

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

15.2.10.2. Configuring the Historian Policy Plugin

The Historian Policy Plugin has no configurable properties.

15.2.11. JPEG2000 Thumbnail Converter

The JPEG2000 Thumbnail converter creates thumbnails from images ingested in jpeg2000 format.

15.2.11.1. Installing the JPEG2000 Thumbnail Converter

The JPEG2000 Thumbnail Converter is installed by default with a standard installation in the Catalog application.

15.2.11.2. Configuring the JPEG2000 Thumbnail Converter

The JPEG2000 Thumbnail Converter has no configurable properties.

15.2.12. 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. For example, if a metacard has two attributes, `sourceattribute1` and `sourceattribute2`, the values of the two attributes could be combined into a new attribute, `destinationattribute1`. The `sourceattribute1` and `sourceattribute2` are the *source attributes* and `destinationattribute1` is the *destination attribute*.

There are two way to combine the values of source attributes. The first, and most common, is to take all of the attribute values and put them together. This is called the union. For example, if the source attributes `sourceattribute1` and `sourceattribute2` had the values:

```
sourceattribute1 = MASK, VESSEL
```

```
sourceattribute2 = WIRE, SACK, MASK
```

...the **union** would result in the new attribute `destinationattribute1`:

```
destinationattribute1 = MASK, VESSEL, WIRE, SACK
```

The other way to combine attributes is use the values common to all of the attributes. This is called the intersection. Using the previous example, the **intersection** of `sourceattribute1` and `sourceattribute2` would create the new attribute `destinationattribute1`

```
destinationattribute1 = MASK
```

because only `MASK` is common to all of the source attributes.

The policy plugin could also be used to rename attributes. If there is only one source attribute, and the combination policy is union, then the attribute's values are effectively renamed to the destination attribute.

15.2.12.1. Installing the Metacard Attribute Security Policy Plugin

The Metacard Attribute Security Policy Plugin is installed by default with a standard installation in the Catalog application.

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

15.2.13. Metacard Backup File Storage Provider

The Metacard Backup File Storage Provider is a storage provider that stores backed-up metacards in a

specified file system location.

15.2.13.1. Installing the Metacard Backup File Storage Provider

To install the Metacard Backup File Storage Provider

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `catalog-metacard-backup-filestorage` feature.

15.2.13.2. Configuring the Metacard Backup File Storage Provider

To configure the Metacard Backup File Storage Provider

1. Navigate to the **Admin Console**.
2. Select Catalog application.
3. Select **Configuration** tab.
4. Select **Metacard Backup File Storage Provider**.

See Metacard Backup File Storage Provider configurations for all possible configurations.

15.2.14. Metacard Backup S3 Storage Provider

The Metacard Backup S3 Storage Provider is a storage provider that stores backed up metacards in the specified S3 bucket and key.

15.2.14.1. Installing the Metacard S3 File Storage Provider

To install the Metacard Backup File Storage Provider

1. Navigate to the **System** tab.
2. Select the **Features** tab.
3. Install the `catalog-metacard-backup-s3storage` feature.

15.2.14.2. Configuring the Metacard S3 File Storage Provider

To configure the Metacard Backup S3 Storage Provider:

1. Navigate to the **Admin Console**.
2. Select Catalog application.
3. Select **Configuration** tab.

4. Select **Metacard Backup S3 Storage Provider**.

See Metacard Backup S3 Storage Provider configurations for all possible configurations.

15.2.15. Metacard Groomer

The Metacard Groomer Pre-Ingest plugin makes modifications to `CreateRequest` and `UpdateRequest` metacards.

Use this pre-ingest plugin as a convenience to apply basic rules for your metacards.

This plugin makes the following modifications when metacards are in a `CreateRequest`:

- Overwrites the `Metacard.ID` field with a generated, unique, 32 character hexadecimal value if missing or if the resource URI is not a catalog resource URI.
- Sets `Metacard.CREATED` to the current time stamp if not already set.
- Sets `Metacard.MODIFIED` to the current time stamp if not already set.
- Sets `Core.METACARD_CREATED` to the current time stamp if not present.
- Sets `Core.METACARD_MODIFIED` to the current time stamp.

In an `UpdateRequest`, the same operations are performed as a `CreateRequest`, except:

- If no value is provided for `Metacard.ID` in the new metacard, it is set using the `UpdateRequest` ID if applicable.

15.2.15.1. Installing the Metacard Groomer

The Metacard Groomer is included in the `catalog-core-plugins` feature. It is not recommended to uninstall this feature.

15.2.15.2. Configuring the Metacard Groomer

The Metacard Groomer has no configurable properties.

15.2.16. Metacard Ingest Network Plugin

The Metacard Ingest Network Plugin allows the conditional insertion of new attributes on metacards during ingest based on network information from the ingest request; including IP address and hostname.

For the extent of this section, a 'rule' refers to a configured, single instance of this plugin.

15.2.16.1. Related Components to the Metacard Ingest Network Plugin

- Client Info Plugin

15.2.16.2. Installing the Metacard Ingest Network Plugin

The Metacard Ingest Network Plugin is installed by default during a standard installation in the Catalog application.

15.2.16.3. Configuring the Metacard Ingest Network Plugin

To configure the Metacard Ingest Network Plugin:

- Navigate to the **Admin Console**.
- Select the Catalog application.
- Select the **Configuration** tab.
- Select the label *Metacard Ingest Network Plugin* to setup a network rule.

See Metacard Ingest Network Plugin configurations for all possible configurations.

Multiple instances of the plugin can be configured by clicking on its configuration title within the configuration tab of the Catalog app. Each instance represents a conditional statement, or a 'rule', that gets evaluated for each ingest request. For any request that meets the configured criteria of a rule, that rule attempts to transform its list of key-value pairs to become new attributes on all metacards in that request.

The rule is divided into two fields: `"Criteria"` and `"Expected Value"`. The `"Criteria"` field features a drop-down list containing the four elements for which equality can be tested:

- IP Address of where the ingest request came from
- Host Name of where the ingest request came from
- Scheme that the ingest request arrived on, for example, *http* vs *https*
- Context Path that the ingest request arrived on, for example, */services/catalog*

In order for a rule to evaluate to true and the attributes be applied, the value in the "Expected Value" field must be an exact match to the actual value of the selected criteria. For example, if the selected criteria is `"IP Address"` with an expected value of `"192.168.0.1"`, the rule only evaluates to true for ingest requests coming from `"192.168.0.1"` and nowhere else.

IMPORTANT	*Check for IPv6* Verify your system's IP configuration. Rules using "IP Address" may need to be written in IPv6 format.

The key-value pairs within each rule should take the following form: "key = value" where the "key" is

the name of the attribute and the "value" is the value assigned to that attribute. Whitespace is ignored unless it is within the key or value. Multi-valued attributes can be expressed in comma-separated format if necessary.

Examples of Valid Attribute Assignments

```
contact.contributor-name = John Doe
contact.contributor-email = john.doe@example.net
language = English
language = English, French, German
security.access-groups = SJ202, SR 101, JS2201
```

15.2.16.3.1. Useful Attributes

The following table provides some useful attributes that may commonly be set by this plugin:

Table 58. Useful Attributes

Attribute Name	Expected Format	Multi-Valued
expiration	ISO DateTime	no
description	Any String	no
metacard.owner	Any String	no
language	Any String	yes
security.access-groups	Any String	yes
security.access-individuals	Any String	yes

15.2.16.4. Usage Limitations of the Metacard Ingest Network Plugin

- This plugin only works for ingest (create requests) performed over a network; data ingested via command line does not get processed by this plugin.
- Any attribute that is already set on the metacard is not overwritten by the plugin.
- The order of execution is not guaranteed. For any rule configuration where two or more rules add different values for the same attribute, it is undetermined which provides the final value for that attribute.

15.2.17. Metacard Resource Size Plugin

The Metacard Resource Size post-query plugin updates the resource size attribute of each metacard in the query results if there is a cached file for the resource and it has a size greater than zero; otherwise, the resource size is unmodified and the original result is returned.

Use this post-query plugin as a convenience to return query results with accurate resource sizes for

cached products.

15.2.17.1. Installing the Metacard Resource Size Plugin

The Metacard Resource Size Plugin is installed by default with a standard installation.

15.2.17.2. Configuring the Metacard Resource Size Plugin

The Metacard Resource Size Plugin has no configurable properties.

15.2.18. Metacard Validity Filter Plugin

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

15.2.18.1. Related Components to the Metacard Validity Filter Plugin

- Metacard Validity Marker.

15.2.18.2. Installing the Metacard Validity Filter Plugin

The Metacard Validity Filter Plugin is installed by default with a standard installation in the Catalog application.

15.2.19. Metacard Validity Marker

The Metacard Validity Marker Pre-Ingest plugin modifies the metacards contained in create and update requests.

The plugin runs each metacard in the `CreateRequest` and `UpdateRequest` against each registered `MetacardValidator` service.

NOTE	This plugin can make it seem like ingested resources are not successfully ingested if a user does not have permissions to access invalid metacards. If an ingest did not fail, there are no errors in the ingest log, but the expected results do not show up after a query, verify either that the ingested data is valid or that the Metacard Validity Filter Plugin is configured to show warnings and/or errors.

15.2.19.1. Related Components to the Metacard Validity Marker

- Metacard Validity Filter Plugin.

15.2.19.2. Installing Metacard Validity Marker

This plugin is installed by default with a standard installation in the Catalog application.

15.2.19.3. Configuring Metacard Validity Marker

See Metacard Validity Marker Plugin configurations for all possible configurations.

15.2.19.4. Using Metacard Validity Marker

Use this pre-ingest plugin to validate metacards against metacard validators, which can check schemas, schematron, or any other logic.

15.2.20. Operation Plugin

The Operation plugin validates the subject's security attributes to ensure they are adequate to perform the operation.

15.2.20.1. Installing the Operation Plugin

The Operation Plugin is installed by default with a standard installation in the Catalog application.

15.2.20.2. Configuring the Operation Plugin

The Operation Plugin has no configurable properties.

15.2.21. Point of Contact Policy Plugin

The Point of Contact Policy Plugin is a PreUpdate plugin that checks if the point-of-contact attribute has changed. If it does, then it adds a policy to that metacard's policy map that cannot be implied. This denies such an update request, which essentially makes the point-of-contact attribute read-only.

15.2.21.1. Related Components to Point of Contact Policy Plugin

Point of Contact Update Plugin

15.2.21.2. Installing the Point of Contact Policy Plugin

The Point of Contact Policy Plugin is installed by default with a standard installation in the Catalog application.

15.2.21.3. Configuring the Point of Contact Policy Plugin

The Point of Contact Policy Plugin has no configurable properties.

15.2.22. Processing Post-Ingest Plugin

The Processing Post Ingest Plugin is responsible for submitting catalog Create, Update, and Delete (CUD) requests to the Processing Framework.

15.2.22.1. Related Components to Processing Post-Ingest Plugin

None.

15.2.22.2. Installing the Processing Post-Ingest Plugin

The Processing Post-Ingest Plugin is not installed by default with a standard installation, but is installed by default when the in-memory Processing Framework is installed.

15.2.22.3. Configuring the Processing Post-Ingest Plugin

The Processing Post-Ingest Plugin has no configurable properties.

15.2.23. Resource URI Policy Plugin

The Resource URI Policy Plugin configures the attributes required for users to set the resource URI when creating a metacard or alter the resource URI when updating an existing metacard in the catalog.

15.2.23.1. Installing the Resource URI Policy Plugin

The Resource URI Policy Plugin is installed by default with a standard installation in the Catalog application.

15.2.23.2. Configuring the Resource URI Policy Plugin

To configure the Resource URI Policy Plugin:

1. Navigate to the **Admin Console**.
2. Select Catalog application.
3. Select **Configuration** tab.
4. Select **Resource URI Policy Plugin**.

See Resource URI Policy Plugin configurations for all possible configurations.

15.2.24. Security Audit Plugin

The Security Audit Plugin is used to allow the auditing of specific metacard attributes. Any time a

metacard attribute listed in the configuration is updated, a log entry is generated in the security log.

15.2.24.1. Installing the Security Audit Plugin

The Security Audit Plugin is installed by default with a standard installation in the Catalog application.

15.2.25. Security Logging Plugin

The Security Logging Plugin logs operations to the security log.

15.2.25.1. Installing Security Logging Plugin

The Security Logging Plugin is installed by default in a standard installation in the Security application.

15.2.25.2. Enhancing the Security Log

The security log contains attributes related to the subject acting on the system. To add additional attributes related to the subject to the logs, append the attribute's key to the comma separated values assigned to `security.logger.extra_attributes` in `/etc/custom.system.properties`.

15.2.26. Security Plugin

The Security Plugin identifies the subject for an operation.

15.2.26.1. Installing the Security Plugin

The Security Plugin is installed by default with a standard installation in the Catalog application.

15.2.26.2. Configuring the Security Plugin

The Security Plugin has no configurable properties.

15.2.27. Tags Filter Plugin

The Tags Filter Plugin updates queries without filters for tags, and adds a default tag of `resource`. For backwards compatibility, a filter is also added to include metacards without any tags attribute.

15.2.27.1. Related Components to Tags Filter Plugin

None.

15.2.27.2. Installing the Tags Filter Plugin

The Tags Filter Plugin is installed by default with a standard installation in the Catalog application.

15.2.27.3. Configuring the Tags Filter Plugin

The Tags Filter Plugin has no configurable properties.

15.2.28. Video Thumbnail Plugin

The Video Thumbnail Plugin provides the ability to generate thumbnails for video files stored in the Content Repository.

It is an implementation of both the `PostCreateStoragePlugin` and `PostUpdateStoragePlugin` interfaces. When installed, it is invoked by the Catalog Framework immediately after a content item has been created or updated by the Storage Provider.

This plugin uses a custom 32-bit LGPL build of FFmpeg (a video processing program) to generate thumbnails. When this plugin is installed, it places the FFmpeg executable appropriate for the current operating system in `<DDF_HOME>/bin_third_party/ffmpeg`. When invoked, this plugin runs the FFmpeg binary in a separate process to generate the thumbnail. The `<DDF_HOME>/bin_third_party/ffmpeg` directory is deleted when the plugin is uninstalled.

NOTE	Prebuilt FFmpeg binaries are provided for Linux, Mac, and Windows only.

15.2.28.1. Installing the Video Thumbnail Plugin

The Video Thumbnail Plugin is installed by default with a standard installation in the Catalog application.

15.2.28.2. Configuring the Video Thumbnail Plugin

To configure the Video Thumbnail Plugin:

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

See Video Thumbnail Plugin configurations for all possible configurations.

15.2.29. Workspace Access Plugin

The Workspace Access Plugin prevents non-owner users from changing workspace permissions.

15.2.29.1. Related Components to The Workspace Access Plugin

- Workspace Sharing Policy Plugin.
- Workspace Pre-Ingest Plugin.
- Workspace Extension.

15.2.29.2. Installing the Workspace Access Plugin

The Workspace Access Plugin is installed by default with a standard installation in the Catalog application.

15.2.29.3. Configuring the Workspace Access Plugin

The Workspace Access Plugin has no configurable properties.

15.2.30. Workspace Pre-Ingest Plugin

The Workspace Pre-Ingest Plugin verifies that a workspace has an associated email to enable sharing and assigns that email as `"owner"`.

15.2.30.1. Related Components to The Workspace Pre-Ingest Plugin

- Workspace Sharing Policy Plugin.
- Workspace Access Plugin.
- Workspace Extension.

15.2.30.2. Installing the Workspace Pre-Ingest Plugin

The Workspace Pre-Ingest Plugin is installed by default with a standard installation in the Catalog application.

15.2.30.3. Configuring the Workspace Pre-Ingest Plugin

The Workspace Pre-Ingest Plugin has no configurable properties.

15.2.31. Workspace Sharing Policy Plugin

The Workspace Sharing Policy Plugin collects attributes for a workspace to identify the appropriate

policy to apply to allow sharing.

15.2.31.1. Related Components to The Workspace Sharing Policy Plugin

- Workspace Access Plugin.
- Workspace Pre-Ingest Plugin.
- Workspace Extension.

15.2.31.2. Installing the Workspace Sharing Policy Plugin

The Workspace Sharing Policy Plugin is installed by default with a standard installation in the Catalog application.

15.2.31.3. Configuring the Workspace Sharing Policy Plugin

The Workspace Sharing Policy Plugin has no configurable properties.

15.2.32. XML Attribute Security Policy Plugin

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

15.2.32.1. Installing the XML Attribute Security Policy Plugin

The XML Attribute Security Policy Plugin is installed by default with a standard installation in the Security application.

---

16. Data

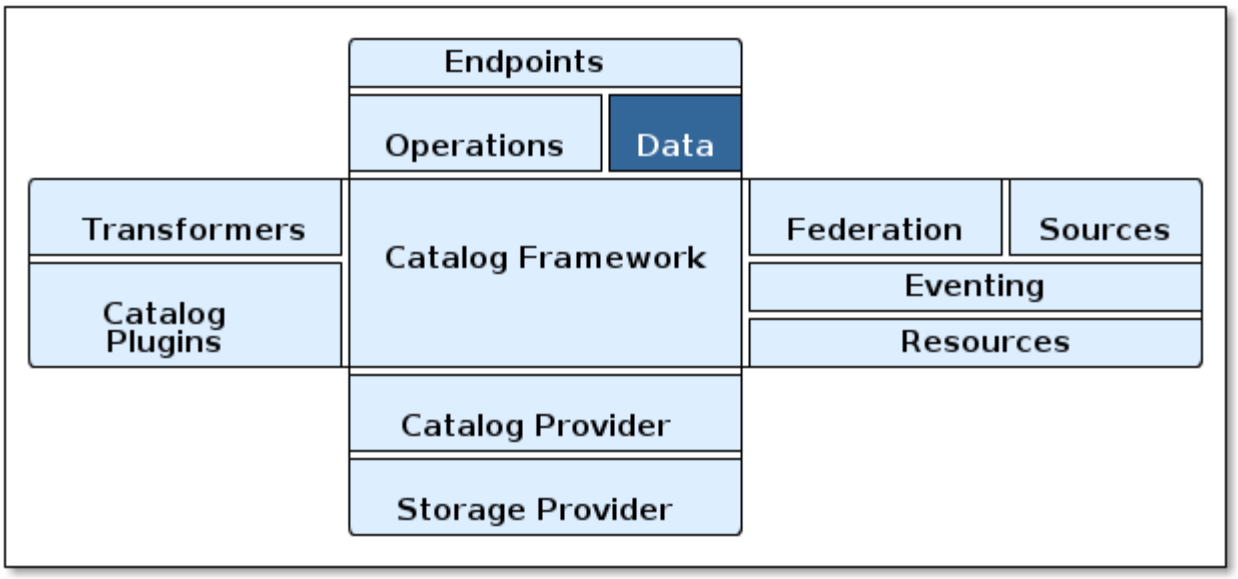


Figure 23. Catalog Architecture Diagram Data

The Catalog stores and translates Metadata, which can be transformed into many data formats, shared, and queried. The primary form of this metadata is the metacard. A `Metacard` is a container for metadata. `CatalogProviders` accept `Metacards` as input for ingest, and `Sources` search for metadata and return matching `Results` that include `Metacards`.

16.1. Metacards

A metacard is a single instance of metadata in the Catalog (an instance of a metacard type) which generally contains general information about the resource, such as the title of the resource, the resource's geo-location, the date the resource was created and/or modified, the owner or producer, and/or the security classification.

16.1.1. Metacard Type

A metacard type indicates the attributes available for a particular metacard. It is a model used to define the attributes of a metacard, much like a schema.

A metacard type indicates the attributes available for a particular type of data. For example, an image may have different attributes than a PDF document, so each could be defined to have their own metacard type.

16.1.1.1. Default Metacard Type and Attributes

Most metacards within the system are created using the default metacard type or a metacard type based on the default type. The default metacard type of the system can be programmatically retrieved by calling `ddf.catalog.data.impl.MetacardImpl.BASIC_METACARD`. The name of the default `MetacardType` can be retrieved from `ddf.catalog.data.MetacardType.DEFAULT_METACARD_TYPE_NAME`.

The default metacard type has the following required attributes. Though the following attributes are required on all metacard types, setting their values is optional except for ID.

Core Attributes

NOTE	It is highly recommended when referencing a default attribute name to use the `ddf.catalog.data.types.*` interface constants whenever possible. Mapping to a normalized taxonomy allows for higher quality transformations between different formats and for improved federation. This neutral profile facilitates improved search and discovery across disparate data types.

WARNING	Every Source should at the very least return an ID attribute according to Catalog API. Other fields may or may not be applicable, but a unique ID must be returned by a source.

16.1.1.2. Extensible Metacards

Metacard extensibility is achieved by creating a new `MetacardType` that supports attributes in addition to the required attributes listed above.

Required attributes must be the base of all extensible metacard types.

WARNING	Not all Catalog Providers support extensible metacards. Nevertheless, each Catalog Provider should at least have support for the default `MetacardType`. That is, it should be able to store and query on the attributes and attribute formats specified by the default metacard type. Catalog providers are neither expected nor required to store attributes that are not in a given metacard's type. Consult the documentation of the Catalog Provider in use for more information on its support of extensible metacards.

Often, the `BASIC_METACARD MetacardType` does not provide all the functionality or attributes necessary for a specific task. For performance or convenience purposes, it may be necessary to create custom attributes even if others are not aware of those attributes. One example could be if a user wanted to optimize a search for a date field that did not fit the definition of `CREATED`, `MODIFIED`, `EXPIRATION`, or `EFFECTIVE`. The user could create an additional `java.util.Date` attribute in order to query the attribute separately.

`Metacard` objects are extensible because they allow clients to store and retrieve standard and custom

key/value Attributes from the Metacard. All Metacards must return a MetacardType object that includes an AttributeDescriptor for each Attribute, indicating it's key and value type. AttributeType support is limited to those types defined by the Catalog.

New MetacardType implementations can be made by implementing the MetacardType interface.

16.1.2. Metacard Type Registry

WARNING	The MetacardTypeRegistry is experimental. While this component has been tested and is functional, it may change as more information is gathered about what is needed and as it is used in more scenarios.

The MetacardTypeRegistry allows DDF components, primarily catalog providers and sources, to make available the MetacardTypes that they support. It maintains a list of all supported MetacardTypes in the CatalogFramework, so that other components such as Endpoints, Plugins, and Transformers can make use of those MetacardTypes. The MetacardType is essential for a component in the CatalogFramework to understand how it should interpret a metacard by knowing what attributes are available in that metacard.

For example, an endpoint receiving incoming metadata can perform a lookup in the MetacardTypeRegistry to find a corresponding MetacardType. The discovered MetacardType is then used to help the endpoint populate a metacard based on the specified attributes in the MetacardType. By doing this, all the incoming metadata elements can then be available for processing, cataloging, and searching by the rest of the CatalogFramework.

MetacardTypes should be registered with the MetacardTypeRegistry. The MetacardTypeRegistry makes those MetacardTypes available to other DDF CatalogFramework components. Other components that need to know how to interpret metadata or metacards should look up the appropriate MetacardType from the registry. By having these MetacardTypes available to the CatalogFramework, these components can be aware of the custom attributes.

The MetacardTypeRegistry is accessible as an OSGi service. The following blueprint snippet shows how to inject that service into another component:

MetacardTypeRegistry Service Injection

```
<bean id="sampleComponent" class="ddf.catalog.SampleComponent">
    <argument ref="metacardTypeRegistry" />
</bean>

<!-- Access MetacardTypeRegistry -->
<reference id="metacardTypeRegistry" interface="ddf.catalog.data.MetacardTypeRegistry"/>
```

The reference to this service can then be used to register new MetacardTypes or to lookup existing ones.

New MetacardTypes are typically registered by CatalogProviders or sources indicating they know how to

persist, index, and query attributes from that type. Endpoints or `InputTransformers` typically use the lookup functionality to access a `MetacardType` based on a parameter in the incoming metadata. Once the appropriate `MetacardType` is discovered and obtained from the registry, the component knows how to translate incoming raw metadata into a DDF Metacard.

16.1.3. Attributes

An attribute is a single field of a metacard, an instance of an attribute type. Attributes are typically indexed for searching by a source or catalog provider.

16.1.3.1. Attribute Types

An attribute type indicates the attribute format of the value stored as an attribute. It is a model for an attribute.

16.1.3.1.1. Attribute Format

An enumeration of attribute formats are available in the catalog. Only these attribute formats may be used.

Table 59. Attribute Formats

AttributeFormat	Description
`BINARY`	Attributes of this attribute format must have a value that is a Java `byte[]` and `AttributeType.getBinding()` should return `Class<Array>of` byte.
`BOOLEAN`	Attributes of this attribute format must have a value that is a Java boolean.
`DATE`	Attributes of this attribute format must have a value that is a Java date.
`DOUBLE`	Attributes of this attribute format must have a value that is a Java double.
`FLOAT`	Attributes of this attribute format must have a value that is a Java float.
`GEOMETRY`	Attributes of this attribute format must have a value that is a WKT-formatted Java string.
`INTEGER`	Attributes of this attribute format must have a value that is a Java integer.
`LONG`	Attributes of this attribute format must have a value that is a Java long.
`OBJECT`	Attributes of this attribute format must have a value that implements the serializable interface.

AttributeFormat	Description
`SHORT`	Attributes of this attribute format must have a value that is a Java short.
`STRING`	Attributes of this attribute format must have a value that is a Java string and treated as plain text.
`XML`	Attributes of this attribute format must have a value that is a XML-formatted Java string.

16.1.3.1.2. Attribute Naming Conventions

Catalog taxonomy elements follow the naming convention of `group-or-namespace.specific-term`, except for extension fields outside of the core taxonomy. These follow the naming convention of `ext.group-or-namespace.specific-term` and must be namespaced. Nesting is not permitted.

16.1.3.2. Result

A single "hit" included in a query response.

A result object consists of the following:

- a metacard.
- a relevance score if included.
- distance in meters if included.

16.1.4. Creating Metacards

The quickest way to create a `Metacard` is to extend or construct the `MetacardImpl` object. `MetacardImpl` is the most commonly used and extended `Metacard` implementation in the system because it provides a convenient way for developers to retrieve and set `Attributes` without having to create a new `MetacardType` (see below). `MetacardImpl` uses `BASIC_METACARD` as its `MetacardType`.

16.1.4.1. Limitations

A given developer does not have all the information necessary to programmatically interact with any arbitrary source. Developers hoping to query custom fields from extensible `Metacards` of other sources cannot easily accomplish that task with the current API. A developer cannot question a source for all its *queryable* fields. A developer only knows about the `MetacardTypes` which that individual developer has used or created previously.

The only exception to this limitation is the `Metacard.ID` field, which is required in every `Metacard` that is stored in a source. A developer can always request `Metacards` from a source for which that developer has the `Metacard.ID` value. The developer could also perform a wildcard search on the `Metacard.ID` field if the source allows.

16.1.4.2. Processing Metacards

As `Metacard` objects are created, updated, and read throughout the Catalog, care should be taken by all catalog components to interrogate the `MetacardType` to ensure that additional `Attributes` are processed accordingly.

16.1.4.3. Basic Types

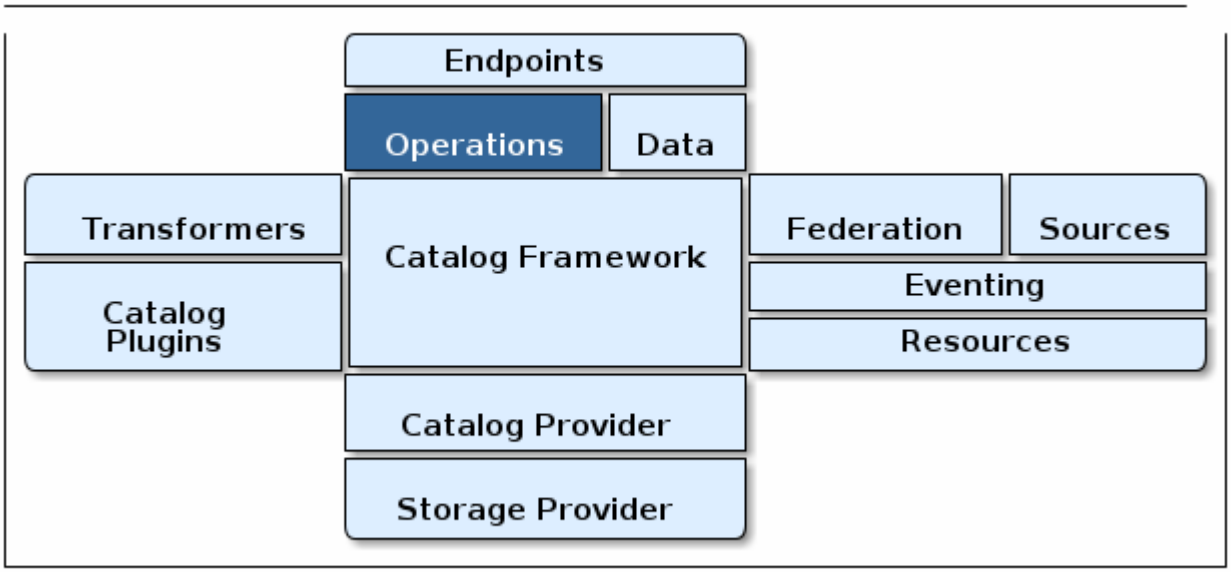
The Catalog includes definitions of several basic types all found in the `ddf.catalog.data.BasicTypes` class.

Table 60. Basic Types

Name	Type	Description
`BASIC_METACARD`	MetacardType	Represents all required Metacard Attributes.
`BINARY_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.BINARY`.
`BOOLEAN_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.BOOLEAN`.
`DATE_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.DATE`.
`DOUBLE_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.DOUBLE`.
`FLOAT_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.FLOAT`.
`GEO_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.GEOMETRY`.
`INTEGER_TYPE`	AttributeType	A Constant for an `AttributeType` with `AttributeType.AttributeFormat.INTEGER`.

Name	Type	Description
LONG_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.LONG.
OBJECT_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.OBJECT.
SHORT_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.SHORT.
STRING_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.STRING.
XML_TYPE	AttributeType	A Constant for an AttributeType with AttributeType.AttributeFormat.XML.

17. Operations



The Catalog provides the capability to query, create, update, and delete metacards; retrieve resources; and retrieve information about the sources in the enterprise.

Each of these operations follow a request/response paradigm. The request is the input to the operation and contains all of the input parameters needed by the Catalog Framework's operation to communicate with the Sources. The response is the output from the execution of the operation that is returned to the client, which contains all of the data returned by the sources. For each operation there is an associated request/response pair. For example, the `QueryRequest` and `QueryResponse` pair for the Catalog Framework's query operation.

All of the request and response objects are extensible in that they can contain additional key/value properties on each request/response. This allows additional capability to be added without changing the Catalog API, helping to maintain backwards compatibility.

18. Resources

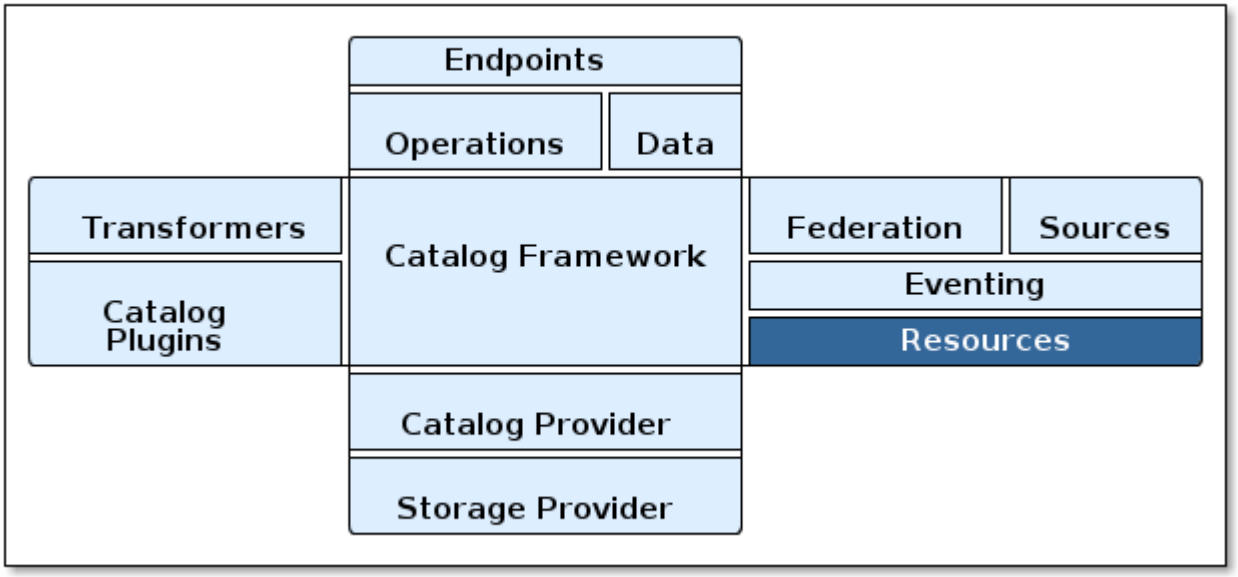


Figure 24. Resources Architecture

Resources are the data that is represented by the cataloged metadata in DDF.

Metacards are used to describe those resources through metadata. This metadata includes the time the resource was created, the location where the resource was created, etc. A DDF `Metacard` contains the `getResourceUri` method, which is used to locate and retrieve its corresponding resource.

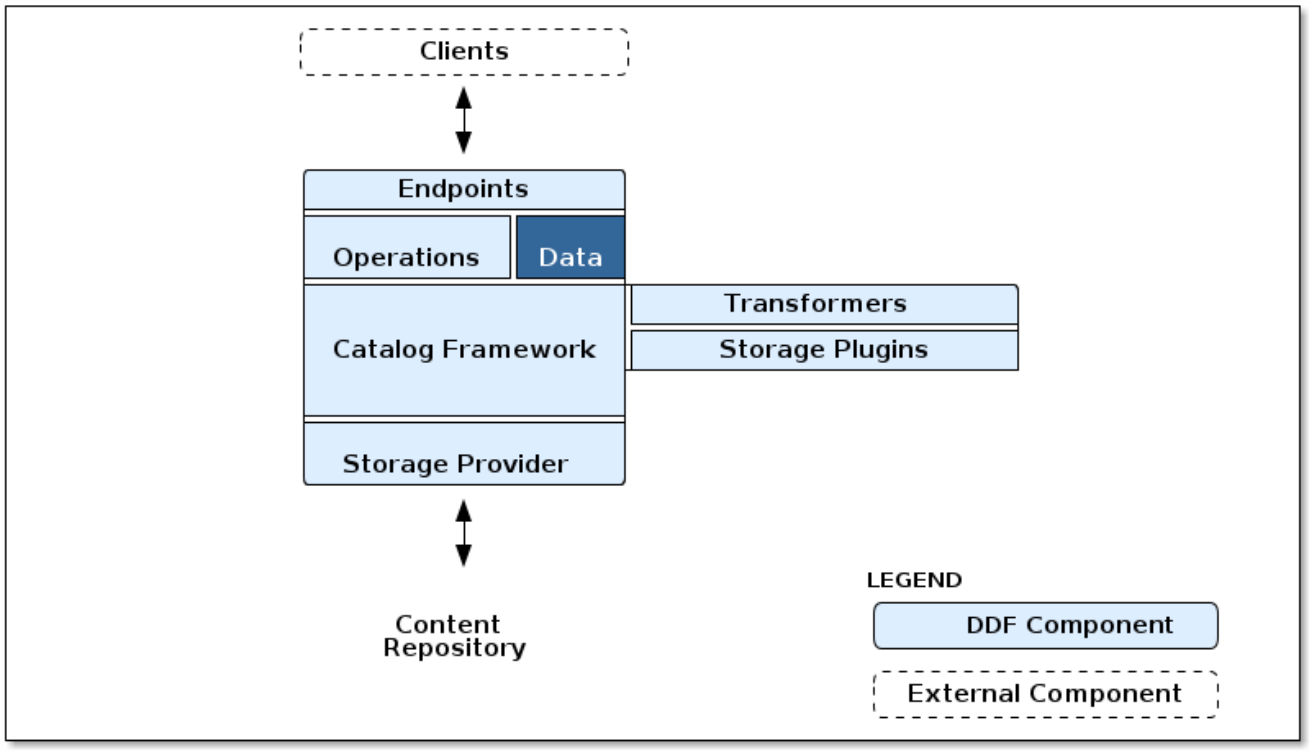


Figure 25. Content Data Component Architecture

18.1. Content Item

Content Item is the domain object populated by the Storage Provider that represents the information about the content to be stored or content that has been stored in the Storage Provider. A `ContentItem` encapsulates the content's globally unique ID, mime type, and input stream (that is, the actual content). The unique ID of a `ContentItem` always corresponds to a Metacard ID.

18.1.1. Retrieving Resources

When a client attempts to retrieve a resource, it must provide a metacard ID or URI corresponding to a unique resource. As mentioned above, the resource URI is obtained from a `Metacard`'s `getResourceUri` method. The `CatalogFramework` has three methods that can be used by clients to obtain a resource: `getEnterpriseResource`, `getResource`, and `getLocalResource`. The `getEnterpriseResource` method invokes the `retrieveResource` method on a local `ResourceReader` as well as all the `Federated` and `Connected` Sources inthe DDF enterprise. The second method, `getResource`, takes in a source ID as a parameter and only invokes `retrieveResource` on the specified `Source`. The third method invokes `retrieveResource` on a local `ResourceReader`.

The parameter for each of these methods in the `CatalogFramework` is a `ResourceRequest`. DDF includes two implementations of `ResourceRequest`: `ResourceRequestById` and `ResourceRequestByProductUri`. Since these implementations extend `OperationImpl`, they can pass a `Map` of generic properties through

the CatalogFramework to customize how the resource request is carried out. One example of this is explained in the Retrieving Resource Options section below. The following is a basic example of how to create a ResourceRequest and invoke the CatalogFramework resource retrieval methods to process the request.

Retrieve Resource Example

```
Map<String, Serializable> properties = new HashMap<String, Serializable>();
properties.put("PropertyKey1", "propertyA"); //properties to customize Resource retrieval
ResourceRequestById resourceRequest = new ResourceRequestById(
"0123456789abcdef0123456789abcdef", properties); //object containing ID of Resource to be
retrieved
String sourceName = "LOCAL_SOURCE"; //the Source ID or name of the local Catalog or a
Federated Source
ResourceResponse resourceResponse; //object containing the retrieved Resource and the
request that was made to get it.
resourceResponse = catalogFramework.getResource(resourceRequest, sourceName); //Source-
based retrieve Resource request
Resource resource = resourceResponse.getResource(); //actual Resource object containing
InputStream, mime type, and Resource name
```

DDF.catalog.resource.ResourceReader instances can be discovered via the OSGi Service Registry. The system can contain multiple ResourceReaders. The CatalogFramework determines which one to call based on the scheme of the resource's URI and what schemes the ResourceReader supports. The supported schemes are obtained by a ResourceReader's getSupportedSchemes method. As an example, one ResourceReader may know how to handle file-based URIs with the scheme file, whereas another ResourceReader may support HTTP-based URIs with the scheme http.

The ResourceReader or Source is responsible for locating the resource, reading its bytes, adding the binary data to a Resource implementation, then returning that Resource in a ResourceResponse. The ResourceReader or Source is also responsible for determining the Resource's name and mime type, which it sends back in the Resource implementation.

18.1.1.1. BinaryContent

BinaryContent is an object used as a container to store translated or transformed DDF components. Resource extends BinaryContent and includes a getName method. BinaryContent has methods to get the InputStream, byte array, MIME type, and size of the represented binary data. An implementation of BinaryContent (BinaryContentImpl) can be found in the Catalog API in the DDF.catalog.data package.

18.1.2. Retrieving Resource Options

Options can be specified on a retrieve resource request made through any of the supporting endpoint. To specify an option for a retrieve resource request, the endpoint needs to first instantiate a ResourceRequestByProductUri or a ResourceRequestById. Both of these ResourceRequest implementations allow a Map of properties to be specified. Put the specified option into the Map under the key

`RESOURCE_OPTION`.

Retrieve Resource with Options

```
Map<String, Serializable> properties = new HashMap<String, Serializable>();
properties.put("RESOURCE_OPTION", "OptionA");
ResourceRequestById resourceRequest = new ResourceRequestById(
"0123456789abcdef0123456789abcdef", properties);
```

Depending on the support that the `ResourceReader` or `Source` provides for options, the `properties Map` is checked for the `RESOURCE_OPTION` entry. If that entry is found, the option is handled. If the `ResourceReader` or `Source` does not support options, that entry is ignored.

A new `ResourceReader` or `Source` implementation can be created to support options in a way that is most appropriate. Since the option is passed through the catalog framework as a property, the `ResourceReader` or `Source` has access to that option as long as the endpoint supports options.

18.1.3. Storing Resources

Resources are saved using a `ResourceWriter`. `DDF.catalog.resource.ResourceWriter` instances can be discovered via the OSGi Service Registry. Once retrieved, the `ResourceWriter` instance provides clients with a way to store resources and get a corresponding URI that can be used to subsequently retrieve the resource via a `ResourceReader`. Simply invoke either of the `storeResource` methods with a resource and any potential arguments. The `ResourceWriter` implementation is responsible for determining where the resource is saved and how it is saved. This allows flexibility for a resource to be saved in any one of a variety of data stores or file systems. The following is an example of how to use a generic implementation of `ResourceWriter`.

Using a ResourceWriter

```
InputStream inputStream = <Video_Input_Stream>; //InputStream of raw Resource data
MimeType mimeType = new MimeType("video/mpeg"); //Mime Type or content type of Resource
String name = "Facility_Video";  //Descriptive Resource name
Resource resource = new ResourceImpl(inputStream, mimeType, name);
Map<String, Object> optionalArguments = new HashMap<String, Object>();
ResourceWriter writer = new ResourceWriterImpl();
URI resourceUri; //URI that can be used to retrieve Resource
resourceUri = writer.storeResource(resource, optionalArguments); //Null can be passed in
here
```

18.2. Resource Components

Resource components are used when working with resources

A resource is a URI-addressable entity that is represented by a metacard.

Resources may exist either locally or on a remote data store.

Examples of resources include:

- NITF image
- MPEG video
- Live video stream
- Audio recording
- Document

A resource object in DDF contains an `InputStream` with the binary data of the resource. It describes that resource with a name, which could be a file name, URI, or another identifier. It also contains a mime type or content type that a client can use to interpret the binary data.

18.3. Resource Readers

A Resource Reader retrieves resources associated with metacards via URIs. Each resource reader must know how to interpret the resource's URI and how to interact with the data store to retrieve the resource.

There can be multiple resource readers in a Catalog instance. The `Catalog Framework` selects the appropriate resource reader based on the scheme of the resource's URI.

In order to make a resource reader available to the Catalog Framework, it must be exported to the OSGi Service Registry as a `DDF.catalog.resource.ResourceReader`.

18.3.1. URL Resource Reader

The `URLResourceReader` is an implementation of `ResourceReader` which is included in the DDF Catalog. It obtains a resource given an http, https, or file-based URL. The `URLResourceReader` connects to the provided Resource URL and reads the resource's bytes into an `InputStream`.

WARNING

When a resource linked using a file-based URL is in the product cache, the `URLResourceReader`'s `rootResourceDirectories` is not checked when downloading. It is downloaded from the product cache which bypasses the `URLResourceReader`. For example, if path `/my/valid/path` is configured in the `URLResourceReader`'s `rootResourceDirectories` and one downloads the product with resource-uri `file:///my/valid/path/product.txt` and then one removes `/my/valid/path` from the `URLResourceReader`'s `rootResourceDirectories` configuration, the product is still accessible via the product cache.

18.3.1.1. Installing the URL Resource Reader

The `URLResourceReader` is installed by default with a standard installation in the Catalog application.

18.3.1.2. Configuring Permissions for the URL Resource Reader

Configuring the URL Resource Reader to retrieve files requires adding Security Manager read permission entries for the directory containing the resources. To add the correct permission entries, edit the file <DDF_HOME>/security/configurations.policy. In the URL Resource Reader section of the file, add two new permission for each top-level directory that the Resource Reader needs to access. The Resource Reader needs one permission to read the directory and another to read its contents.

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

```
grant codeBase "file:/org.apache.tika.core/catalog-core-urlresourcereader" {
//# Add permissions here
permission java.io.FilePermission "<DIRECTORY_PATH>", "read";
permission java.io.FilePermission "<OTHER_DIRECTORY_PATH>", "read";
}
```

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.

18.3.1.3. Configuring the URL Resource Reader

Configure the URL Resource Reader from the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.
4. Select the **URL Resource Reader**.

See URL Resource Reader configurations for all possible configurations.

18.3.1.4. Using the URL Resource Reader

`URLResourceReader` is used by the Catalog Framework to obtain a resource whose metacard is cataloged in the local data store. This particular `ResourceReader` is chosen by the `CatalogFramework` if the requested resource's URL has a protocol of `http`, `https`, or `file`.

For example, requesting a resource with the following URL makes the Catalog Framework invoke the `URLResourceReader` for retrieval.

Example

```
file:///home/users/DDF_user/data/example.txt
```

If a resource was requested with the URL `udp://123.45.67.89:80/SampleResourceStream`, the `URLResourceReader` would *not* be invoked.

Supported Schemes:

- http
- https
- file

NOTE	If a file-based URL is passed to the `URLResourceReader`, that file path needs to be accessible by the DDF instance.

18.4. Resource Writers

A Resource Writer stores a resource and produces a URI that can be used for retrieval. The resource URI uniquely locates and identifies the resource. Resource writers can interact with an underlying data store and store the resource in the proper place. Each implementation can do this differently, providing flexibility in the data stores used to persist the resources.

Resource Writers should be used within the Content Framework if and when implementing a custom Storage Provider to store data. The default Storage Provider that comes with the DDF writes the resources to the file system.

19. Queries

Clients use `ddf.catalog.operation.Query` objects to describe which metacards are needed from Sources.

Query objects have two major components:

- Filters
- Query Options

A Source uses the Filter criteria constraints to find the requested set of metacards within its domain of metacards. The Query Options are used to further restrict the Filter's set of requested metacards.

19.1. Filters

An OGC Filter is a Open Geospatial Consortium (OGC) standard that describes a query expression in

terms of Extensible Markup Language (XML) and key-value pairs (KVP). The OGC Filter is used to represent a query to be sent to sources and the Catalog Provider, as well as to represent a Subscription. The OGC Filter provides support for expression processing, such as adding or dividing expressions in a query, but that is not the intended use for DDF.

The Catalog Framework does not use the XML representation of the OGC Filter standard. DDF instead uses the Java implementation provided by GeoTools. GeoTools provides Java equivalent classes for OGC Filter XML elements. GeoTools originally provided the standard Java classes for the OGC Filter Encoding 1.0 under the package name `org.opengis.filter`. That package name was changed to `org.geotools.api.filter` in version 30.0 and is currently used by DDF. Java developers do not parse or view the XML representation of a Filter in DDF. Instead, developers use only the Java objects to complete query tasks.

Note that the `ddf.catalog.operation.Query` interface extends the `org.geotools.api.filter.Filter` interface, which means that a Query object is an OGC Java Filter with Query Options.

A Query is an OGC Filter

```
public interface Query extends Filter
```

19.1.1. FilterBuilder API

To avoid the complexities of working with the Filter interface directly and implementing the DDF Profile of the Filter specification, the Catalog includes an API, primarily in `DDF.filter`, to build Filters using a fluent API.

To use the `FilterBuilder` API, an instance of `DDF.filter.FilterBuilder` should be used via the OSGi registry. Typically, this is injected via a dependency injection framework. Once an instance of `FilterBuilder` is available, methods can be called to create and combine Filters.

TIP	The fluent API is best accessed using an IDE that supports code-completion. For additional details, refer to the [Catalog API Javadoc].

19.1.1.1. Boolean Operators

Filters use a number of boolean operators.

`FilterBuilder.allOf(Filter …)`

creates a new Filter that requires all provided Filters are satisfied (Boolean AND), either from a List or Array of Filter instances.

`FilterBuilder.anyOf(Filter …)`

creates a new Filter that requires at least one of the provided Filters are satisfied (Boolean OR), either from a List or Array of Filter instances.

`FilterBuilder.not(Filter filter)`

creates a new Filter that requires the provided Filter must not match (Boolean NOT).

19.1.1.2. Attribute

Filters can be based on specific attributes.

`FilterBuilder.attribute(String attributeName)`

begins a fluent API for creating an Attribute-based Filter, which is a Filter that matches on Metacards with Attributes of a particular value.

20. Action Framework

The Action Framework was designed as a way to limit dependencies between applications (apps) in a system. For instance, a feature in an app, such as an Atom feed generator, might want to include an external link as part of its feed's entries. That feature does not have to be coupled to a REST endpoint to work, nor does it have to depend on a specific implementation to get a link. In reality, the feature does not identify how the link is generated, but it does identify whether the link works or does not work when retrieving the intended entry's metadata. Instead of creating its own mechanism or adding an unrelated feature, it could use the Action Framework to query the OSGi container for any service that can provide a link. This does two things: it allows the feature to be independent of implementations, and it encourages reuse of common services.

The Action Framework consists of two major Java interfaces in its API:

1. `ddf.action.Action`
2. `ddf.action.ActionProvider`

Actions

Specific tasks that can be performed as services.

Action Providers

Lists of related actions that a service is capable of performing.

20.1. Action Providers

Included Action Providers

Download Resource ActionProvider

Downloads a resource to the local product cache.

IdP Logout Action Provider

Identity Provider Logout.

Karaf Logout Action

Local Logout.

LDAP Logout Action

Ldap Logout.

Overlay ActionProvider

Provides a metacard URL that transforms the metacard into a geographically aligned image (suitable for overlaying on a map).

View Metacard ActionProvider

Provides a URL to a metacard.

Metacard Transformer ActionProvider

Provides a URL to a metacard that has been transformed into a specified format.

21. Asynchronous Processing Framework

NOTE	This code is experimental. While this interface is functional and tested, it may change or be removed in a future version of the library.

The Asynchronous Processing Framework is a way to run plugins asynchronously. Generally, plugins that take a significant amount of processing time and whose results are not immediately required are good candidates for being asynchronously processed. A **Processing Framework** can either be run on the local or remote system. Once the **Processing Framework** finishes processing incoming requests, it may submit `(Create|Update|Delete)Request`s to the Catalog. The type of plugins that a **Processing Framework** runs are the **Post-Process Plugins**. The **Post-Process Plugins** are triggered by the **Processing Post Ingest Plugin**, which is a **Post-Ingest Plugin**. **Post-Ingest Plugins** are run after the metacard has been ingested into the Catalog. This feature is uninstalled by default.

WARNING	The **Processing Framework** does not support partial updates to the Catalog. This means that if any changes are made to a metacard in the Catalog between the time asynchronous processing starts and ends, those changes are overwritten by the **ProcessingFramework** updates sent back to the Catalog. This feature should be used with caution.

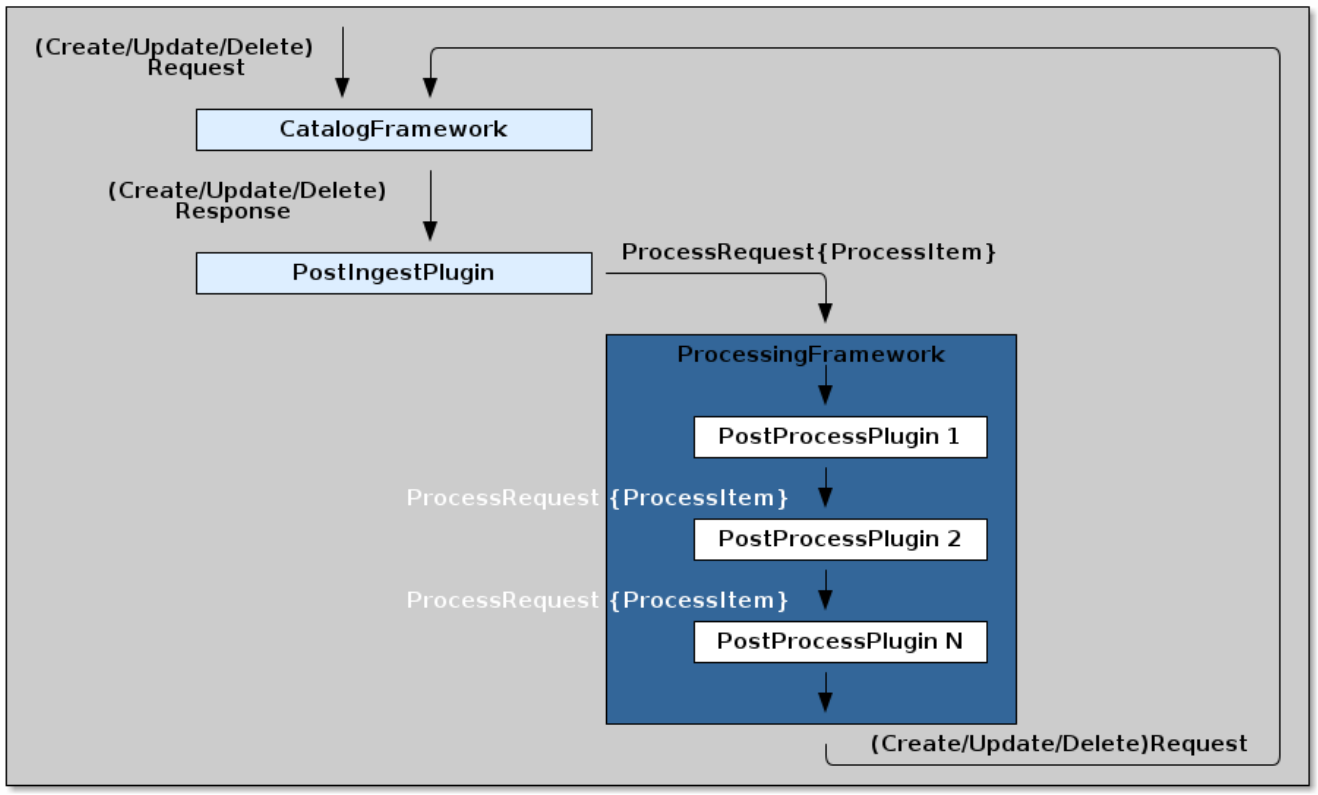


Figure 26. Processing Framework Architecture

The Asynchronous Processing Framework API Interfaces

1. `org.codice.ddf.catalog.async.processingframework.api.internal.ProcessingFramework`
2. `org.codice.ddf.catalog.async.plugin.api.internal.PostProcessPlugin`
3. `org.codice.ddf.catalog.async.data.api.internal.ProcessItem`
4. `org.codice.ddf.catalog.async.data.api.internal.ProcessCreateItem`
5. `org.codice.ddf.catalog.async.data.api.internal.ProcessUpdateItem`
6. `org.codice.ddf.catalog.async.data.api.internal.ProcessDeleteItem`
7. `org.codice.ddf.catalog.async.data.api.internal.ProcessRequest`
8. `org.codice.ddf.catalog.async.data.api.internal.ProcessResoure`
9. `org.codice.ddf.catalog.async.data.api.internal.ProcessResourceItem`

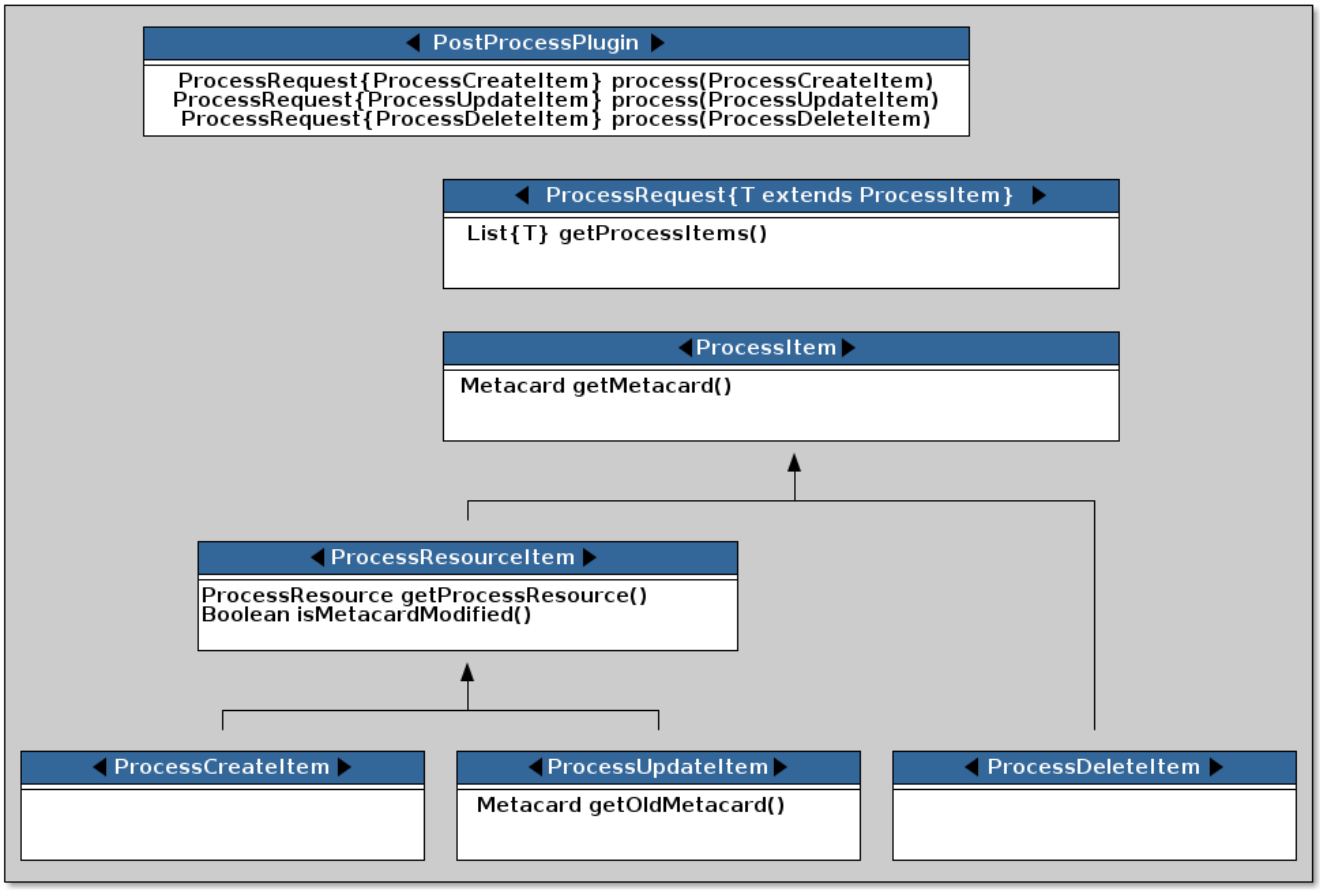


Figure 27. Processing Framework Interface Diagram

ProcessingFramework

The `ProcessingFramework` is responsible for processing incoming `ProcessRequest`s that contain a `ProcessItem`. A `ProcessingFramework` should never block. It receives its `ProcessRequest`s from a `PostIngestPlugin` on all CUD operations to the Catalog. In order to determine whether or not asynchronous processing is required by the `ProcessingFramework`, the `ProcessingFramework` should mark any request it has submitted back the Catalog, otherwise a processing loop may occur. For example, the default **In-Memory Processing Framework** adds a `POST_PROCESS_COMPLETE` flag to the Catalog CUD request after processing. This flag is checked by the `ProcessingPostIngestPlugin` before a `ProcessRequest` is sent to the `ProcessingFramework`. For an example of a `ProcessingFramework`, please refer to the `org.codice.ddf.catalog.async.processingframework.impl.InMemoryProcessingFramework`.

ProcessRequest

A `ProcessRequest` contains a list of `ProcessItem`s for the `ProcessingFramework` to process. Once a `ProcessRequest` has been processed by a `ProcessingFramework`, the `ProcessingFramework` should mark the `ProcessRequest` as already been processed, so that it does not process it again.

PostProcessPlugin

The `PostProcessPlugin` is a plugin that is run by the `ProcessingFramework`. It is capable of processing `ProcessCreateItem`s, `ProcessUpdateItem`s, and `ProcessDeleteItem`s.

WARNING	*ProcessItem* Do not implement `ProcessItem` directly; it is intended for use only as a common base interface for `ProcessResourceItem` and `ProcessDeleteItem`.

The `ProcessItem` is contained by a `ProcessRequest`. It can be either a `ProcessCreateItem`, `ProcessUpdateItem`, or `ProcessDeleteItem`.

ProcessResource

The `ProcessResource` is a piece of content that is attached to a metacard. The piece of content can be either local or remote.

ProcessResourceItem

The `ProcessResourceItem` indicates that the item being processed may have a `ProcessResource` associated with it.

WARNING	*ProcessResourceItem Warning* Do not implement `ProcessResourceItem` directly; it is intended for use only as a common base interface for `ProcessCreateItem` and `ProcessUpdateItem`.

ProcessCreateItem

The `ProcessCreateItem` is an item for a metacard that has been created in the Catalog. It contains the created metacard and, optionally, a `ProcessResource`.

ProcessUpdateItem

The `ProcessUpdateItem` is an item for a metacard that has been updated in the Catalog. It contains the original metacard, the updated metacard and, optionally, a `ProcessResource`.

ProcessDeleteItem

The `ProcessDeleteItem` is an item for a metacard that has been deleted in the Catalog. It contains the deleted metacard.

22. Eventing

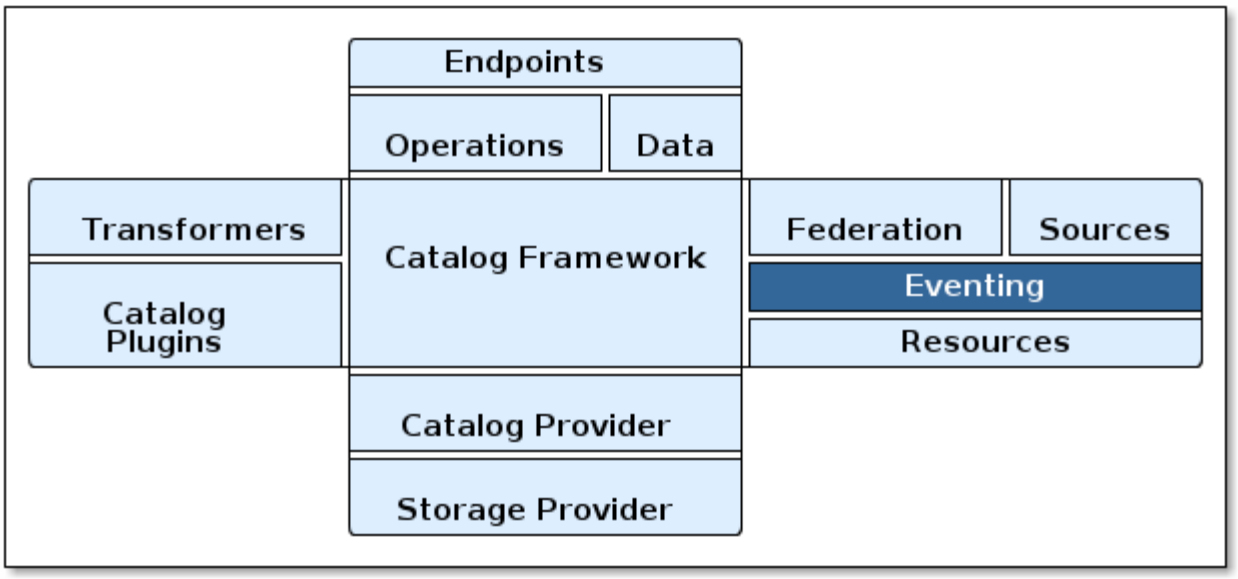


Figure 28. Eventing Architecture

The Eventing capability of the Catalog allows endpoints (and thus external users) to create a "standing query" and be notified when a matching metacard is created, updated, or deleted.

Notably, the Catalog allows event evaluation on both the previous value (if available) and new value of a Metacard when an update occurs.

Eventing allows DDFs to receive events on operations (such as create, update, delete) based on particular queries or actions. Once subscribed, users receive notifications of events such as update or create on any source.

22.1. Eventing Components

The key components of DDF Eventing include:

- Subscription
- Delivery Method
- Event Processor

23. Security Framework

The DDF Security Framework utilizes Apache Shiro as the underlying security framework. The classes mentioned in this section have their full package name listed, to make it easy to tell which classes come

with the core Shiro framework and which are added by DDF.

23.1. Subject

`ddf.security.Subject <extends> org.apache.shiro.subject.Subject`

The Subject is the key object in the security framework. Most of the workflow and implementations revolve around creating and using a Subject. The Subject object in DDF is a class that encapsulates all information about the user performing the current operation. The Subject can also be used to perform permission checks to see if the calling user has acceptable permission to perform a certain action (for example, calling a service or returning a metacard). This class was made DDF-specific because the Shiro interface cannot be added to the Query Request property map.

Table 61. Implementations of Subject:

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

23.1.1. Security Manager

`ddf.security.service.SecurityManager`

The Security Manager is a service that handles the creation of Subject objects. A proxy to this service should be obtained by an endpoint to create a Subject and add it to the outgoing `QueryRequest`. The Shiro framework relies on creating the subject by obtaining it from the current thread. Due to the multi-threaded and stateless nature of the DDF framework, utilizing the Security Manager interface makes retrieving Subjects easier and safer.

Table 62. Implementations of Security Managers:

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

23.1.2. Realms

DDF uses Apache Shiro for the concept of Realms for Authentication and Authorization. Realms are components that access security data such as such as users or permissions.

23.1.2.1. Authenticating Realms

`org.apache.shiro.realm.AuthenticatingRealm`

Authenticating Realms are used to authenticate an incoming Authentication Token and return Authentication Info on successful authentication. This Authentication Info is used by the Shiro framework to put together a resulting Subject. A Subject represents the application user and contains all available security-relevant information about that user.

Table 63. Implementations of Authenticating Realms in DDF:

Classname	Description
`org.codice.ddf.security.guest.realm.GuestRealm`	This realm checks if Guest access is allowed on the incoming Authentication Token, and if so the Guest realm returns the Guest Authentication Info.
`org.codice.ddf.security.oidc.realm.OidcRealm`	This realm takes in any OIDC/OAuth credentials found on the incoming Authentication Token, and if so resolves the ID_Token using those credentials. The ID_Token is then used to put together the resulting Authentication Info.
`ddf.security.realm.sts.StsRealm`	This realm delegates authentication to the Secure Token Service (STS). It creates a `RequestSecurityToken` message from the incoming Authentication Token and converts a successful STS response into Authentication Info.

23.1.2.2. Authorizing Realms

`org.apache.shiro.realm.AuthorizingRealm`

Authorizing Realms are used to perform authorization on the current Subject. These are used when performing both service authorization and filtering. They are passed in the `AuthorizationInfo` of the Subject along with the permissions of the object wanting to be accessed. The response from these realms is a true (if the Subject has permission to access) or false (if the Subject does not).

Table 64. Other implementations of the Security API within DDF

Classname	Description
`org.codice.ddf.platform.filter.delegate.DelegateServletFilter`	The `DelegatingHttpFilterHandler` detects any servlet filters that have been exposed as OSGi services implementing `org.codice.ddf.platform.filter.SecurityFilter` and places them in-order in front of any servlet or web application running on the container.

Classname	Description
`org.codice.ddf.security.filter.websso.WebSSOFilter`	This filter is the main security filter that works with a number of handlers to protect a variety of web contexts, each using different authentication schemes and policies. It attaches an Authentication Token to the request by either checking the session or calling the configured Security Handlers.
`org.codice.ddf.security.handler.basic.BasicAuthenticationHandler`	Checks for basic authentication credentials in the http request header. If no credentials are found, it supports the acquisition of basic credentials on user-agent requests.
`org.codice.ddf.security.handler.pki.PKIHandler`	Handler for PKI based authentication. X509 chain is extracted from the HTTP request.
`org.codice.security.idp.client.IdpHandler`	Handler for IdP/SAML based authentication. If no credentials are found, it supports the acquisition of credentials through the configured SAML IdP.
`org.codice.ddf.security.handler.oidc.OidcHandler`	Handler for OIDC based authentication. If no credentials are found, and is a user-agent request, this handler supports the acquisition of credentials through the configured OIDC IdP.
`org.codice.ddf.security.handler.oauth.OAuthHandler`	Handler for OAuth based authentication. Does not support the acquisition of credentials.
`org.codice.ddf.security.filter.login.LoginFilter`	This filter runs immediately after the WebSSOFilter and exchanges an Authentication Token found in the request with a Subject via Shiro.
`org.codice.ddf.security.filter.authorization.AuthorizationFilter`	This filter runs immediately after the `LoginFilter` and checks any permissions assigned to the web context against the attributes of the Subject via Shiro.
`org.apache.shiro.realm.AuthenticatingRealm`	This is an abstract authenticating realm that exchanges an `org.apache.shiro.authc.AuthenticationToken` for a `org.apache.shiro.authc.AuthenticationInfo`, which is used by the Shiro framework to put together a `ddf.security.Subject`.
`ddf.security.service.AbstractAuthorizingRealm`	This is an abstract authorizing realm that takes care of caching and parsing the Subject's `AuthorizingInfo` and should be extended to allow the implementing realm to focus on making the decision.

Classname	Description
`ddf.security.pdp.realm.AuthZRealm`	This realm performs the authorization decision and may or may not delegate out to the external XACML processing engine. It uses the incoming permissions to create a decision. However, it is possible to extend this realm using the `ddf.security.policy.extension.PolicyExtension interface`. This interface allows an integrator to add additional policy information to the PDP that can't be covered via its generic matching policies. This approach is often easier to configure for those that are not familiar with XACML.
`org.codice.ddf.security.validator.*`	A number of validators are provided for X.509 and Username tokens.

WARNING

An update was made to the IdpHandler to pass SAML assertions through the Authorization HTTP header. Cookies *are* still accepted and processed to maintain legacy federation compatibility, but assertions are sent in the header on outbound requests. While a machine's identity still federates between versions, a user's identity is ONLY federated when a DDF version 2.7.x server communicates with a DDF version 2.8.x+ server, or between two servers whose versions are 2.8.x or higher.

23.2. Security Core

The contains all of the necessary components that are used to perform security operations (authentication, authorization, and auditing) required in the framework.

23.2.1. Security Core API

The Security Core API contains all of the DDF APIs that are used to perform security operations within DDF.

23.2.1.1. Installing the Security Core API

The Security Services App installs the Security Core API by default. Do not uninstall the Security Core API as it is integral to system function and all of the other security services depend upon it.

23.2.1.2. Configuring the Security Core API

The Security Core API has no configurable properties.

23.2.2. Security Core Implementation

The Security Core Implementation contains the reference implementations for the Security Core API interfaces that come with the DDF distribution.

23.2.2.1. Installing the Security Core Implementation

The Security Core app installs this bundle by default. It is **recommended** to use this bundle as it contains the reference implementations for many classes used within the Security Framework.

23.2.2.2. Configuring the Security Core Implementation

The Security Core Implementation has no configurable properties.

23.2.3. Security Core Commons

The Security Core Services bundle contains services that are used within DDF to help with performing common security operations. Most notably, this bundle contains the `ddf.security.audit.SecurityLogger` class that performs the security audit logging within DDF.

23.2.3.1. Configuring the Security Core Commons

The Security Core Commons bundle has no configurable properties.

23.3. Security Encryption

The Security Encryption application offers an encryption framework and service implementation for other applications to use. This service is commonly used to encrypt and decrypt default passwords that are located within the metatype and Admin Console.

The encryption service and encryption command, which are based on tink, provide an easy way for developers to add encryption capabilities to DDF.

23.3.1. Security Encryption API

The Security Encryption API bundle provides the framework for the encryption service. Applications that use the encryption service should use the interfaces defined within it instead of calling an implementation directly.

23.3.1.1. Installing Security Encryption API

This bundle is installed by default as part of the `security-encryption` feature. Many applications that come with DDF depend on this bundle and it should not be uninstalled.

23.3.1.2. Configuring the Security Encryption API

The Security Encryption API has no configurable properties.

23.3.2. Security Encryption Implementation

The Security Encryption Implementation bundle contains all of the service implementations for the Encryption Framework and exports those implementations as services to the OSGi service registry.

23.3.2.1. Installing Security Encryption Implementation

This bundle is installed by default as part of the `security-encryption` feature. Other projects are dependent on the services this bundle exports and it should not be uninstalled unless another security service implementation is being added.

23.3.2.2. Configuring Security Encryption Implementation

The Security Encryption Implementation has no configurable properties.

23.3.3. Security Encryption Commands

The Security Encryption Commands bundle enhances the DDF system console by allowing administrators and integrators to encrypt and decrypt values directly from the console.

The `security:encrypt` command allows plain text to be encrypted using AES for encryption. It uses randomly generated keys and associated data that are created when the system is installed, and can be found in the `<DDF_HOME>/etc/keysets` directory. This is useful when displaying password fields in a GUI.

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

```
ddf@local>security:encrypt myPasswordToEncrypt

bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

23.3.3.1. Installing the Security Encryption Commands

This bundle is installed by default with the `security-encryption` feature. This bundle is tied specifically to the DDF console and can be uninstalled if not needed. When uninstalled, however, administrators are not able to encrypt and decrypt data from the console.

23.3.3.2. Configuring the Security Encryption Commands

The Security Encryption Commands have no configurable properties.

23.4. Security LDAP

The Security LDAP application allows the user to configure either an embedded or a standalone LDAP server. The provided features contain a default set of schemas and users loaded to help facilitate authentication and authorization testing.

23.4.1. Embedded LDAP Server

DDF includes an embedded LDAP server (OpenDJ) for testing and demonstration purposes.

WARNING	The embedded LDAP server is intended for testing purposes only and is not recommended for production use.

23.4.1.1. Installing the Embedded LDAP Server

The embedded LDAP server is not installed by default with a standard installation.

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `opendj-embedded` feature.

23.4.1.2. Configuring the Embedded LDAP

Configure the Embedded LDAP from the Admin Console:

1. Navigate to the **Admin Console**.
2. Select the **OpenDj Embedded** application.
3. Select the **Configuration** tab.

Table 65. OpenDJ Embedded Configurable Properties

Configuration Name	Description
LDAP Port	Sets the port for LDAP (plaintext and startTLS). 0 disables the port.
LDAPS Port	Sets the port for LDAPS. 0 disables the port.

Configuration Name	Description
Base LDIF File	Location on the server for a LDIF file. This file is loaded into the LDAP and overwrites any existing entries. This option should be used when updating the default groups/users with a new LDIF file for testing. The LDIF file being loaded may contain any LDAP entries (schemas, users, groups, etc.). If the location is left blank, the default base LDIF file included with DDF is used.

23.4.1.3. Connecting to Standalone LDAP Servers

DDF instances can connect to external LDAP servers by installing and configuring the `security-jaas-ldap` and `security-claims-ldap` features detailed here.

In order to connect to more than one LDAP server, configure these features for each LDAP server.

23.4.1.4. Embedded LDAP Configuration

The Embedded LDAP application contains an LDAP server (OpenDJ version 2.6.2) that has a default set of schemas and users loaded to help facilitate authentication and authorization testing.

Table 66. Embedded LDAP Default Ports Settings

Protocol	Default Port
`LDAP`	1389
`LDAPS`	1636
`StartTLS`	1389

Table 67. Embedded LDAP Default Users

Username	Password	Groups	Description
`testuser1`	`password1`		General test user for authentication
`testuser2`	`password2`		General test user for authentication
`nromanova`	`password1`	`avengers`	General test user for authentication
`lcage`	`password1`	`admin`, `avengers`	General test user for authentication, Admin user for karaf
`jhowlett`	`password1`	`admin`, `avengers`	General test user for authentication, Admin user for karaf
`pparker`	`password1`	`admin`, `avengers`	General test user for authentication, Admin user for karaf
`jdrew`	`password1`	`admin`, `avengers`	General test user for authentication, Admin user for karaf

Username	Password	Groups	Description
tstark	password1	admin, avengers	General test user for authentication, Admin user for karaf
bbanner	password1	admin, avengers	General test user for authentication, Admin user for karaf
srogers	password1	admin, avengers	General test user for authentication, Admin user for karaf
admin	admin	admin	Admin user for karaf

Table 68. Embedded LDAP Default Admin User Settings

Username	Password	Groups	Attributes	Description
admin	secret			Administrative User for LDAP

23.4.1.5. Schemas

The default schemas loaded into the LDAP instance are the same defaults that come with OpenDJ.

Table 69. Embedded LDAP Default Schemas

Schema File Name	Schema Description
00-core.ldif	This file contains a core set of attribute type and objectclass definitions from several standard LDAP documents, including draft-ietf-boreham-numsubordinates, draft-findlay-ldap-groupofentries, draft-furuseth-ldap-untypedobject, draft-good-ldap-changelog, draft-ietf-ldup-subentry, draft-wahl-ldap-adminaddr, RFC 1274, RFC 2079, RFC 2256, RFC 2798, RFC 3045, RFC 3296, RFC 3671, RFC 3672, RFC 4512, RFC 4519, RFC 4523, RFC 4524, RFC 4530, RFC 5020, and X.501.
01-pwpolicy.ldif	This file contains schema definitions from draft-behera-ldap-password-policy, which defines a mechanism for storing password policy information in an LDAP directory server.
02-config.ldif	This file contains the attribute type and objectclass definitions for use with the directory server configuration.
03-changelog.ldif	This file contains schema definitions from draft-good-ldap-changelog, which defines a mechanism for storing information about changes to directory server data.
03-rfc2713.ldif	This file contains schema definitions from RFC 2713, which defines a mechanism for storing serialized Java objects in the directory server.
03-rfc2714.ldif	This file contains schema definitions from RFC 2714, which defines a mechanism for storing CORBA objects in the directory server.

Schema File Name	Schema Description
03-rfc2739.ldif	This file contains schema definitions from RFC 2739, which defines a mechanism for storing calendar and vCard objects in the directory server. Note that the definition in RFC 2739 contains a number of errors, and this schema file has been altered from the standard definition in order to fix a number of those problems.
03-rfc2926.ldif	This file contains schema definitions from RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.
03-rfc3112.ldif	This file contains schema definitions from RFC 3112, which defines the authentication password schema.
03-rfc3712.ldif	This file contains schema definitions from RFC 3712, which defines a mechanism for storing printer information in the directory server.
03-uddiv3.ldif	This file contains schema definitions from RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.
04-rfc2307bis.ldif	This file contains schema definitions from the `draft-howard-rfc2307bis` specification, used to store naming service information in the directory server.
05-rfc4876.ldif	This file contains schema definitions from RFC 4876, which defines a schema for storing Directory User Agent (DUA) profiles and preferences in the directory server.
05-samba.ldif	This file contains schema definitions required when storing Samba user accounts in the directory server.
05-solaris.ldif	This file contains schema definitions required for Solaris and OpenSolaris LDAP naming services.
06-compat.ldif	This file contains the attribute type and `objectclass` definitions for use with the directory server configuration.

23.4.1.6. Starting and Stopping the Embedded LDAP

The embedded LDAP application installs a feature with the name `ldap-embedded`. Installing and uninstalling this feature starts and stops the embedded LDAP server. This also installs a fresh instance of the server each time. If changes need to persist, stop then start the `embedded-ldap-opendj` bundle (rather than installing/uninstalling the feature).

All settings, configurations, and changes made to the embedded LDAP instances are persisted across DDF restarts. If DDF is stopped while the LDAP feature is installed and started, it automatically restarts with the saved settings on the next DDF start.

23.4.1.7. Limitations of the Embedded LDAP

Current limitations for the embedded LDAP instances include:

- Inability to store the LDAP files/storage outside of the DDF installation directory. This results in any LDAP data (that is, LDAP user information) being lost when the `ldap-embedded` feature is uninstalled.
- Cannot be run standalone from DDF. In order to run `embedded-ldap`, the DDF must be started.

23.4.1.8. External Links for the Embedded LDAP

Location to the default base LDIF file in the DDF source code.

OpenDJ documentation

23.4.1.9. LDAP Administration

OpenDJ provides a number of tools for LDAP administration. Refer to the OpenDJ Admin Guide.

23.4.1.10. Downloading the Admin Tools

Download OpenDJ (Version 2.6.4) and the included tool suite.

23.4.1.11. Using the Admin Tools

The admin tools are located in `<opendj-installation>/bat` for Windows and `<opendj-installation>/bin` for `nix. These tools can be used to administer both local and remote LDAP servers by setting the *host` and **port** parameters appropriately.

In this example, the user **Bruce Banner (uid=bbanner)** is disabled using the **manage-account** command on Windows. Run **manage-account --help** for usage instructions.

Example Commands for Disabling/Enabling a User's Account

```
D:\OpenDJ-2.4.6\bat>manage-account set-account-is-disabled -h localhost -p 4444 -O true
-D "cn=admin" -w secret -b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Account Is Disabled:  true
```

Notice `Account Is Disabled: true` in the listing:

Verifying an Account is Disabled

```
D:\OpenDJ-2.4.6\bat>manage-account get-all -h localhost -p 4444  -D "cn=admin" -w secret
-b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Password Policy DN:  cn=Default Password Policy,cn=Password Policies,cn=config
Account Is Disabled:  true
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

Enabling an Account

```
D:\OpenDJ-2.4.6\bat>manage-account clear-account-is-disabled  -h localhost -p 4444  -D
"cn=admin" -w secret -b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Account Is Disabled:  false
```

Notice `Account Is Disabled: false` in the listing.

Verifying an Account is Enabled

```
D:\OpenDJ-2.4.6\bat>manage-account get-all -h localhost -p 4444  -D "cn=admin" -w secret
-b "uid=bbanner,ou=users,dc=example,dc=com"
The server is using the following certificate:
    Subject DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Issuer DN:  CN=Win7-1, O=Administration Connector Self-Signed Certificate
    Validity:  Wed Sep 04 15:36:46 MST 2013 through Fri Sep 04 15:36:46 MST 2015
Do you wish to trust this certificate and continue connecting to the server?
Please enter "yes" or "no":yes
Password Policy DN:  cn=Default Password Policy,cn=Password Policies,cn=config
Account Is Disabled:  false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

23.5. Security PDP

The Security Policy Decision Point (PDP) module contains services that are able to perform authorization decisions based on configurations and policies. In the Security Framework, these components are called realms, and they implement the `org.apache.shiro.realm.Realm` and `org.apache.shiro.authz.Authorizer` interfaces. Although these components perform decisions on access control, enforcement of this decision is performed by components within the notional PEP application.

23.5.1. Security PDP AuthZ Realm

The Security PDP AuthZ Realm exposes a realm service that makes decisions on authorization requests

using the attributes stored within the metacard to determine if access should be granted. This realm can use XACML and delegates decisions to an external processing engine if internal processing fails. Decisions are first made based on the "match-all" and "match-one" logic. Any attributes listed in the "match-all" or "match-one" sections are not passed to the XACML processing engine and they are matched internally. It is recommended to list as many attributes as possible in these sections to avoid going out to the XACML processing engine for performance reasons. If it is desired that all decisions be passed to the XACML processing engine, remove all of the "match-all" and "match-one" configurations. The configuration below provides the mapping between user attributes and the attributes being asserted - one map exists for each type of mapping (each map may contain multiple values).

Match-All Mapping

This mapping is used to guarantee that all values present in the specified metacard attribute exist in the corresponding user attribute.

Match-One Mapping

This mapping is used to guarantee that at least one of the values present in the specified metacard attribute exists in the corresponding user attribute.

23.5.1.1. Configuring the Security PDP AuthZ Realm

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

See Security AuthZ Realm for all possible configurations.

23.5.2. Guest Interceptor

The goal of the `GuestInterceptor` is to allow non-secure clients (such as SOAP requests without security headers) to access secure service endpoints.

All requests to secure endpoints must satisfy the WS-SecurityPolicy that is included in the WSDL.

Rather than reject requests without user credentials, the guest interceptor detects the missing credentials and inserts an assertion that represents the "guest" user. The attributes included in this guest user assertion are configured by the administrator to represent any unknown user on the current network.

23.5.2.1. Installing Guest Interceptor

The `GuestInterceptor` is installed by default with Security Application.

23.5.2.2. Configuring Guest Interceptor

Configure the Guest Interceptor from the Admin Console:

1. Navigate to the **Admin Console** at https://{FQDN}:{PORT}/admin
2. Select the **Security** application.
3. Select the **Configuration** tab.
4. Select the **Guest Claims Configuration** configuration.
5. Select the + next to Attributes to add a new attribute.
6. Add any additional attributes that apply to every user.
7. Select **Save changes**.

Once these configurations have been added, the GuestInterceptor is ready for use. Both secure and non-secure requests are accepted by all secure DDF service endpoints.

23.6. Web Service Security Architecture

The Web Service Security (WSS) functionality that comes with DDF is integrated throughout the system. This is a central resource describing how all of the pieces work together and where they are located within the system.

DDF comes with a **Security Framework** and **Security Services**. The Security Framework is the set of APIs that define the integration with the DDF framework and the Security Services are the reference implementations of those APIs built for a realistic end-to-end use case.

23.6.1. Securing REST

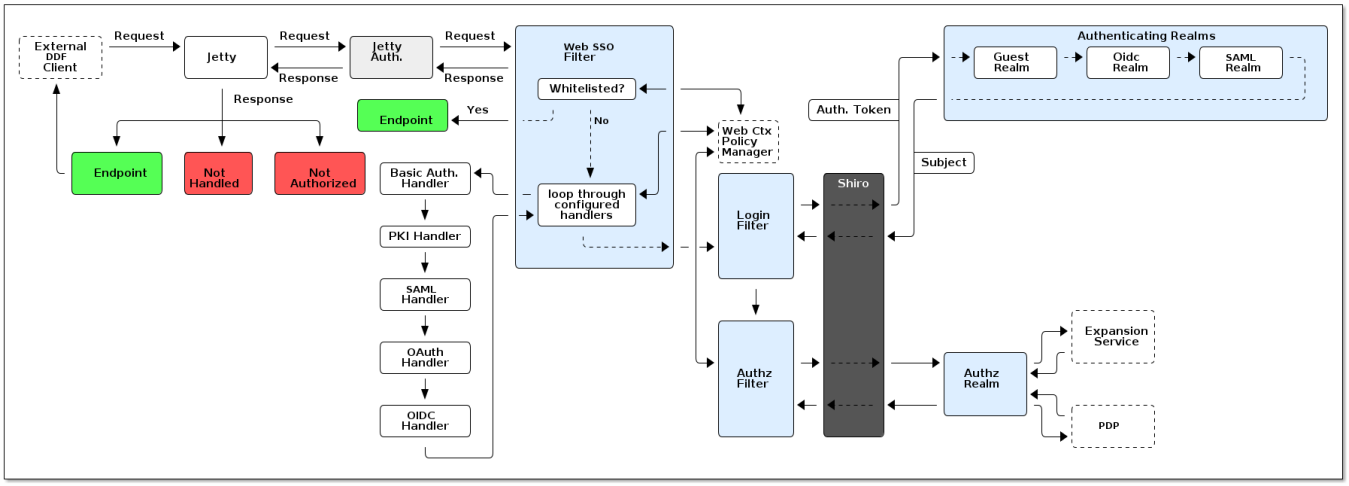


Figure 29. Security Architecture

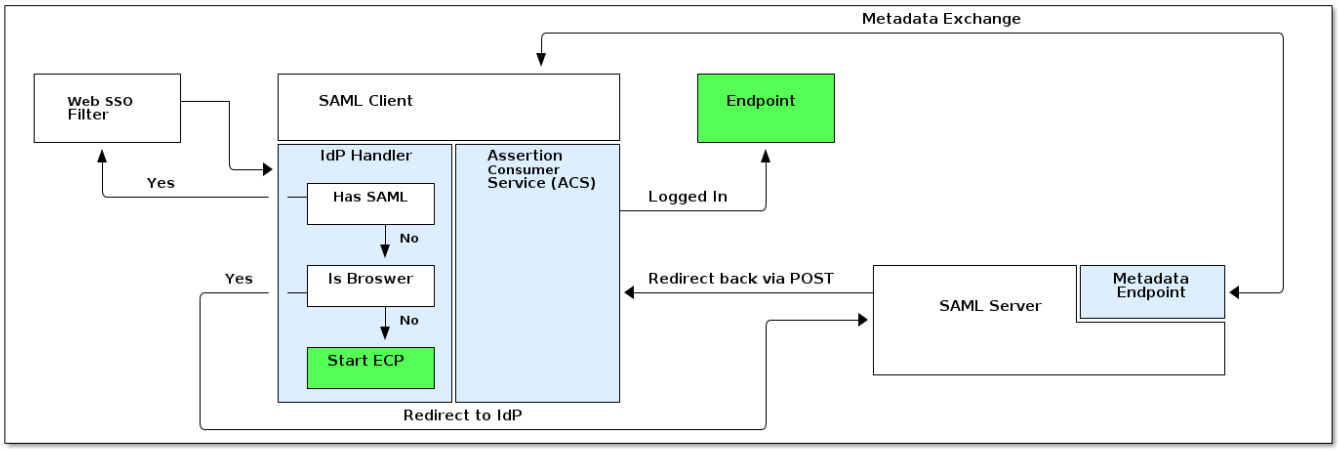
The Jetty Authenticator is the topmost handler of all requests. It initializes all Security Filters and runs them in order according to service ranking:

1. The **Web SSO Filter** reads from the web context policy manager and functions as the first decision point. If the request is from a whitelisted context, no further authentication is needed and the request skips through the rest of the security filters to the desired endpoint.

If the context is not on the whitelist, the filter first attempts to pull authentication information off of the session. If authentication information cannot be found on the session, the filter then attempts to get an authentication handler for the context. The filter loops through all configured context handlers until one signals that it has found authentication information that it can use to build a token. This configuration can be changed by modifying the web context policy manager configuration. If unable to resolve the context, the filter returns an authentication error and the process stops. If a handler is successfully found, an auth token is assigned and the request continues to the login filter.

1. The **Login Filter** receives an authentication token and returns a subject. To retrieve the subject, the authentication token is sent through Shiro to the configured authenticating realms. The realms takes the authentication token and attempts to return authentication info to the Shiro framework in order to put together a subject.
2. If the Subject is returned, the request moves to the **AuthZ Filter** to check permissions on the user. If the user has the correct permissions to access that web context, the request can hit the endpoint.

SAML IdP Architecture



The SAML Handler is a configured handler on the Web SSO Filter just like the other handlers in the previous diagram. The SAML Handler and the Assertion Consumer Service are both part of the IdP client that can be used to interface with any compliant SAML 2.0 Web SSO Identity Provider.

The Metadata Exchange happens asynchronously from any login event. The exchange can happen via HTTP or File, or the metadata XML itself can be pasted into the configuration for the SAML client. The metadata contains information about what bindings are accepted by the client or server and whether or not either expects messages to be signed, etc. The redirect from the Assertion Consumer Service to the Endpoint causes the client to pass back through the entire filter chain, which gets caught at the `Has Session` point of the `WebSsoFilter`. The request proceeds through the rest of the filters as any other connection would in the previous diagram.

Unauthenticated non-browser clients that pass the HTTP headers signaling that they understand SAML ECP can authenticate via that mechanism as explained below.

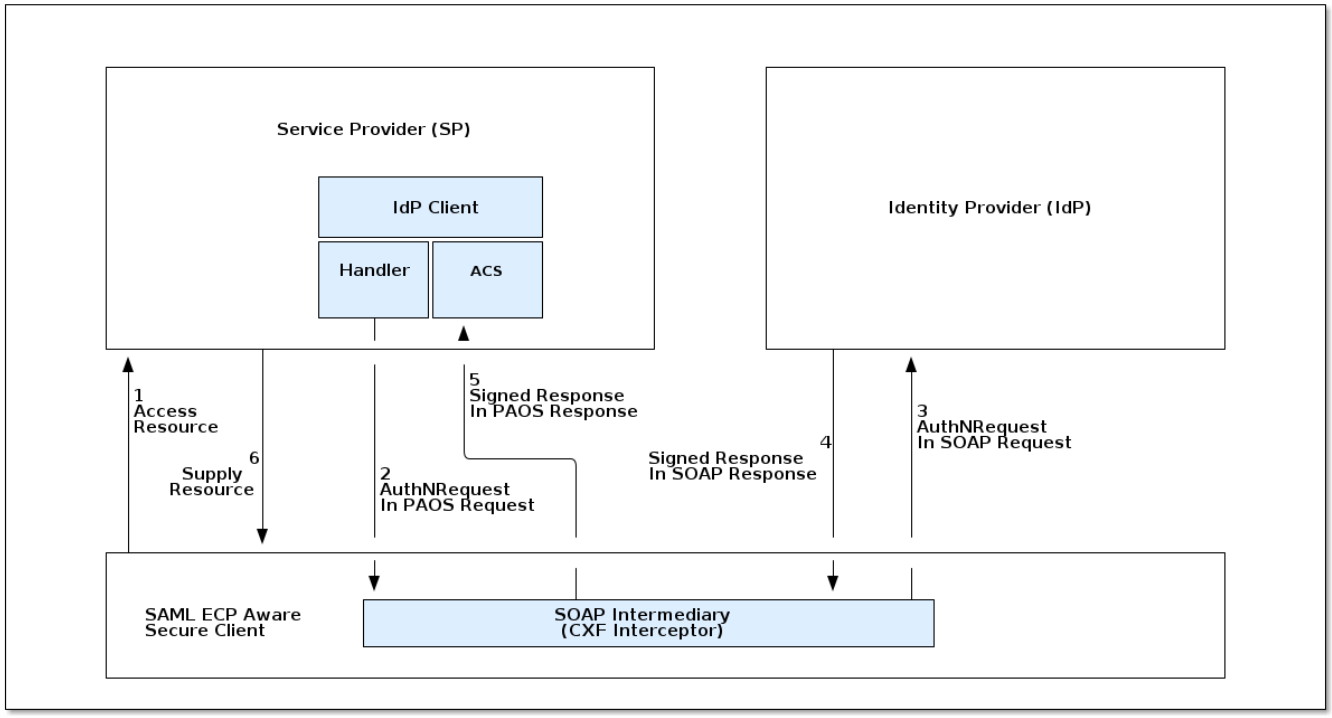


Figure 30. SAML ECP Architecture

SAML ECP can be used to authenticate a non-browser client or non-person entity (NPE). This method of authentication is useful when there is no human in the loop, but authentication with an IdP is still desired. The SAML Handler sends a PAOS (Reverse SOAP) request as an initial response back to the Secure Client, assuming the client has sent the necessary HTTP headers to declare that it supports this function. That response does not complete the request/response loop, but is instead caught by a SOAP intermediary, which is implemented through a CXF interceptor. The PAOS response contains an `<AuthNRequest>` request message, which is intended to be rerouted to a SAML IdP via SOAP. The SOAP intermediary then contacts an IdP (selection of the IdP is not covered by the spec). The IdP either rejects the login attempt, or issues a Signed `<Response>` that is to be delivered to the Assertion Consumer Service by the intermediary. The method of logging into the IdP is not covered by the spec and is up to the implementation. The SP is then signaled to supply the originally requested resource, assuming the signed Response message is valid and the user has permission to view the resource.

The ambiguity in parts of the spec with regard to selecting an IdP to use and logging into that IdP can lead to integration issues between different systems. However, this method of authentication is not necessarily expected to work by default with anything other than other instances of DDF. It does, however, provide a starting point that downstream projects can leverage in order to provide ECP based authentication for their particular scenario or to connect to other systems that utilize SAML ECP.

23.7. Security PEP

The Security Policy Enforcement Point (PEP) application contains bundles that allow for policies to be enforced at various parts of the system, for example: to reach contexts, view metacards, access catalog operations, and others.

23.7.1. Security PEP Interceptor

The Security PEP Interceptor bundle contains the `ddf.security.pep.interceptor.PEPAuthorizingInterceptor` class. This class uses CXF to intercept incoming SOAP messages and enforces service authorization policies by sending the service request to the security framework.

23.7.1.1. Installing the Security PEP Interceptor

This bundle is not installed by default but can be added by installing the `security-pep-serviceauthz` feature.

WARNING	To perform service authorization within a default install of DDF, this bundle MUST be installed.

23.7.1.2. Configuring the Security PEP Interceptor

The Security PEP Interceptor has no configurable properties.

23.8. Filtering

Metacard filtering is performed by the Filter Plugin after a query has been performed, but before the results are returned to the requestor.

Each metacard result contains security attributes that are populated by the CatalogFramework based on the PolicyPlugins that populates this attribute. (These plugins are NOT provided by default. You must create your own plugin for your specific metadata) The security attribute is a HashMap containing a set of keys that map to lists of values. The metacard is then processed by a filter plugin that creates a `KeyValueCollectionPermission` from the metacard's security attribute. This permission is then checked against the user subject to determine if the subject has the correct claims to view that metacard. The decision to filter the metacard eventually relies on the PDP (`feature:install security-pdp-authz`). The PDP returns a decision, and the metacard is either filtered or allowed to pass through.

The security attributes populated on the metacard are completely dependent on the type of the metacard. Each type of metacard must have its own PolicyPlugin that reads the metadata being returned and returns the metacard's security attribute. If the subject permissions are missing during filtering, all resources are filtered.

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

```
<metacard>
    <security>
        <map>
            <entry key="entry1" value="A,B" />
            <entry key="entry2" value="X,Y" />
            <entry key="entry3" value="USA,GBR" />
            <entry key="entry4" value="USA,AUS" />
        </map>
    </security>
</metacard>
```

```
<user>
    <claim name="claim1">
        <value>A</value>
        <value>B</value>
    </claim>
    <claim name="claim2">
        <value>X</value>
        <value>Y</value>
    </claim>
    <claim name="claim3">
        <value>USA</value>
    </claim>
    <claim name="claim4">
        <value>USA</value>
    </claim>
</user>
```

In the above example, the user's claims are represented very simply and are similar to how they would actually appear in a SAML 2 assertion. Each of these user (or subject) claims are converted to a `KeyValuePermission` object. These permission objects are implied against the permission object generated from the metacard record. In this particular case, the metacard might be allowed if the policy is configured appropriately because all of the permissions line up correctly.

To enable filtering on a new type of record, implement a PolicyPlugin that is able to read the string metadata contained within the metacard record. Note that, in DDF, there is no default plugin that parses a metacard. A plugin must be created to create a policy for the metacard.

23.9. Expansion Service

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

in their own implementations.

Expansion Service Rulesets

Each instance of the expansion service consists of a collection of rulesets. Each ruleset consists of a key value and its associated set of rules. Callers of the expansion service provide a key and a value to be expanded. The expansion service then looks up the set of rules for the specified key. The expansion service cumulatively applies each of the rules in the set, starting with the original value. The result is returned to the caller.

Table 70. Expansion Service Ruleset Format

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

Included Expansions

Note that the rules listed for each key are processed in order, so they may build upon each other. That is, a new value from the new replacement string may be expanded by a subsequent rule. In the example `Location:Goodyear` would expand to `Goodyear AZ USA` and `Title:VP-Sales` would expand to `VP-Sales VP Sales`.

To use the expansion service, modify the following two files within the `<DDF_HOME>/etc/pdp` directory:

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

The examples below use the following collection of rulesets:

Table 71. Expansion Service Example Ruleset

Key (Attribute)	Rules (original → new)	
Location	`Goodyear`	`Goodyear AZ`
	AZ	`AZ USA`
	CA	`CA USA`
Title	`VP-Sales`	`VP-Sales VP Sales`
	VP-Engineering	`VP-Engineering VP Engineering`

It is expected that multiple instances of the expansion service are running at the same time. Each instance of the service defines a unique property that is useful for retrieving specific instances of the expansion service. There are two pre-defined instances used by DDF: one for expanding user attributes

and one for metacard attributes.

Property Name	Value	Description
mapping	`security.user.attribute.mapping`	This instance is configured with rules that expand the user's attribute values for security checking.
mapping	`security.metacard.attribute.mapping`	This instance is configured with rules that expand the metacard's security attributes before comparing with the user's attributes.

Expansion Service Configuration Files

Additional instance of the expansion service can be configured using a configuration file. The configuration file can have three different types of lines:

comments

any line prefixed with the `#` character is ignored as a comment (for readability, blank lines are also ignored)

attribute separator

a line starting with `separator=` defines the attribute separator string.

rule

all other lines are assumed to be rules defined in a string format `<key>:<original value>:<new value>`

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

Sample Expansion Configuration File

```
# This defines the separator that will be used when the expansion string contains
multiple
# values - each will be separated by this string. The expanded string will be split at
the
# separator string and each resulting attribute added to the attribute set (duplicates
are
# suppressed). No value indicates the default value of ' ' (space).
separator=

# The following rules define the attribute expansion to be performed. The rules are of
the
# form:
#       <attribute name>:<original value>:<expanded value>
# The rules are ordered, so replacements from the first rules may be found in the
original
```

```
# values of subsequent rules.
Location:Goodyear:Goodyear AZ
Location:AZ:AZ USA
Location:CA:CA USA
Title:VP-Sales:VP-Sales VP Sales
Title:VP-Engineering:VP-Engineering VP Engineering
```

Expansion Commands

DDF includes commands to work with the Expansion service.

Table 72. Included Expansion Commands

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

Command	Description	Sample Input	Results
`security:expand`	Runs the expansion service on the provided data returning the expanded value. It takes an attribute and an original value, expands the original value using the current expansion service and ruleset and dumps the results.	`ddf@local>security:expand Location Goodyear`	`[Goodyear, USA, AZ]`
		`ddf@local>security:expand Title VP-Engineering`	`[VP-Engineering, Engineering, VP]`
		`ddf@local>expand Title "VP-Engineering Manager"`	`[VP-Engineering, Engineering, VP, Manager]`

`security:expansions`	Displays the ruleset for each active expansion service.	`Expansion service configured:`
		
		`ddf@local>security:expansions`
			`[Location : Goodyear : Goodyear AZ`
			`Location : AZ : AZ USA`
			`Location : CA : CA USA`
			`Title : VP-Sales : VP-Sales VP Sales`
			`Title : VP-Engineering : VP-Engineering VP Engineering]`

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

23.10. Federated Identity

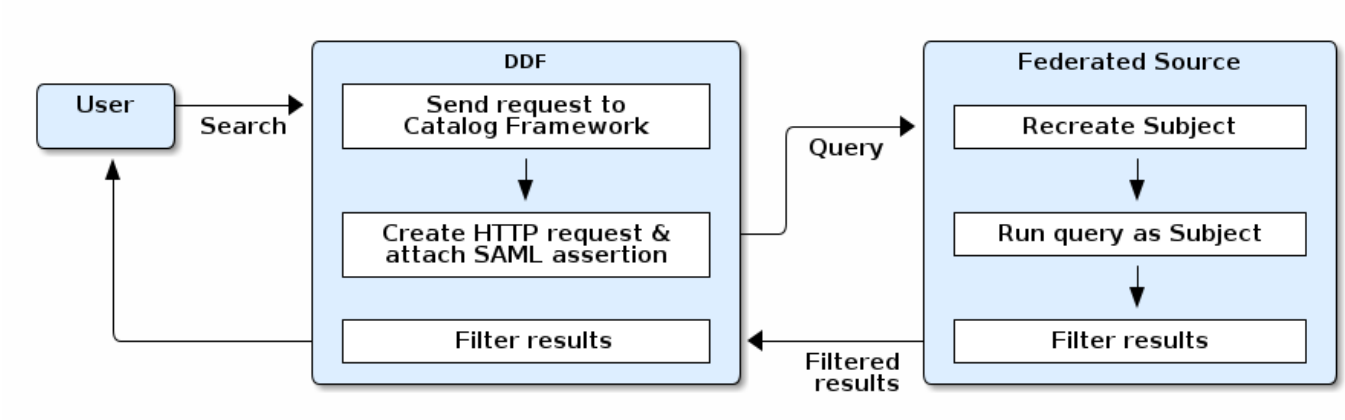
Each instance of DDF may be configured with its own security policy that determines the resources a user may access and the actions they may perform. To decide whether a given request is permitted, DDF references any attributes stored internally in the requestor's Subject. Based on these attributes and the configured policy, DDF makes an authorization decision. See Security PDP for more information.

This authorization process works when the requestor authenticates directly with DDF as they are guaranteed to have a Subject. However, when federating, DDF proxies requests to federated Sources and this poses a problem. The requestor doesn't authenticate with federated Sources, but Sources still need to make authorization decisions.

To solve this problem, DDF uses federated identity. When performing any federated request (query, resource retrival, etc), DDF attaches the requestor's SAML assertion to the outgoing request. The federated Source extracts the assertion and validates its signature to make sure it was generated by a trusted entity. If so, the federated Source constructs a Subject for the requestor and perform the request using that Subject. The Source can then make authorization decisions using the process already described.

How DDF attaches SAML assertions to federated requests depends on the endpoint used to connect to a federated Source. When using a REST endpoint such as CSW, DDF places the assertion in the HTTP Authorization header. When using a SOAP endpoint, it places the assertion in the SOAP security header.

The figure below shows a federated query between two instances of DDF that support federated identity.



1. A user submits a search to DDF.
2. DDF generates a catalog request, attaches the user's Subject, and sends the request to the Catalog Framework.

3. The Catalog Framework extracts the SAML assertion from the Subject and sends an HTTP request to each federated Source with the assertion attached.
4. A federated Source receives this request and extracts the SAML assertion. The federated Source then validates the authenticity of the SAML Assertion. If the assertion is valid, the federated Source generates a Subject from the assertion to represent the user who initiated the request.
5. The federated Source filters all results that the user is not authorized to view and returns the rest to DDF.
6. DDF takes the results from all Sources, filters those that the user is not authorized to view and returns the remaining results to the user.

NOTE	With federated identity, results are filtered both by the federated Source and client DDF. This is important as each may have different authorization policies.

WARNING	Support for federated identity was added in DDF 2.8.x. Federated Sources older than this do not perform any filtering. Instead, they return all available results and leave filtering up to the client.

24. Developing DDF Components

Create custom implementations of DDF components.

24.1. Developing Complementary Catalog Frameworks

DDF and the underlying OSGi technology can serve as a robust infrastructure for developing frameworks that complement the Catalog.

24.1.1. Simple Catalog API Implementations

The Catalog API implementations, which are denoted with the suffix of `Impl` on the Java file names, have multiple purposes and uses:

- First, they provide a good starting point for other developers to extend functionality in the framework. For instance, extending the `MetacardImpl` allows developers to focus less on the inner workings of DDF and more on the developer's intended purposes and objectives.
- Second, the Catalog API Implementations display the proper usage of an interface and an interface's intentions. Also, they are good code examples for future implementations. If a developer does not want to extend the simple implementations, the developer can at least have a working code reference on which to base future development.

24.1.2. Use of the Whiteboard Design Pattern

The Catalog makes extensive use of the Whiteboard Design Pattern. Catalog Components are registered

as services in the OSGi Service Registry, and the Catalog Framework or any other clients tracking the OSGi Service Registry are automatically notified by the OSGi Framework of additions and removals of relevant services.

The Whiteboard Design Pattern is a common OSGi technique that is derived from a technical whitepaper provided by the OSGi Alliance in 2004. It is recommended to use the Whiteboard pattern over the Listener pattern in OSGi because it provides less complexity in code (both on the client and server sides), fewer deadlock possibilities than the Listener pattern, and closely models the intended usage of the OSGi framework.

24.1.3. Recommendations for Framework Development

- Provide extensibility similar to that of the Catalog.
  - Provide a stable API with interfaces and simple implementations (refer to http://www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html).
- Make use of the Catalog wherever possible to store, search, and transform information.
- Utilize OSGi standards wherever possible.
  - ConfigurationAdmin
  - MetaType
- Utilize the sub-frameworks available in DDF.
  - Karaf
  - CXF
  - PAX Web and Jetty

24.1.4. Catalog Framework Reference

The Catalog Framework can be requested from the OSGi Service Registry.

Blueprint Service Reference

```
<reference id="catalogFramework" interface="DDF.catalog.CatalogFramework" />
```

24.1.4.1. Methods

The `CatalogFramework` provides convenient methods to transform `Metacards` and `QueryResponses` using a reference to the `CatalogFramework`.

24.1.4.1.1. Create, Update, and Delete Methods

Create, Update, and Delete (CUD) methods add, change, or remove stored metadata in the local Catalog Provider.

Example Create, Update, Delete Methods

```
public CreateResponse create(CreateRequest createRequest) throws IngestException,
SourceUnavailableException;
public UpdateResponse update(UpdateRequest updateRequest) throws IngestException,
SourceUnavailableException;
public DeleteResponse delete(DeleteRequest deleteRequest) throws IngestException,
SourceUnavailableException;
```

CUD operations process `PolicyPlugin`, `AccessPlugin`, and `PreIngestPlugin` instances before execution and `PostIngestPlugin` instances after execution.

24.1.4.1.2. Query Methods

Query methods search metadata from available Sources based on the `QueryRequest` properties and Federation Strategy. Sources could include Catalog Provider, Connected Sources, and Federated Sources.

Example Query Methods

```
public QueryResponse query(QueryRequest query) throws UnsupportedQueryException
,SourceUnavailableException, FederationException;
public QueryResponse query(QueryRequest queryRequest, FederationStrategy strategy) throws
SourceUnavailableException, UnsupportedQueryException, FederationException;
```

Query requests process `PolicyPlugin`, `AccessPlugin`, and `PreQueryPlugin` instances before execution and `PolicyPlugin`, `AccessPlugin`, and `PostQueryPlugin` instances after execution.

24.1.4.1.3. Resource Methods

Resource methods retrieve data resources from Sources.

Example Resource Methods

```
public ResourceResponse getEnterpriseResource(ResourceRequest request) throwsIOException,
ResourceNotFoundException, ResourceNotSupportedException;
public ResourceResponse getLocalResource(ResourceRequest request) throws IOException,
ResourceNotFoundException, ResourceNotSupportedException;
public ResourceResponse getResource(ResourceRequest request, String resourceSiteName)
throws IOException, ResourceNotFoundException, ResourceNotSupportedException;
```

Resource requests process `PreResourcePlugin`s before execution and `PostResourcePlugin`s after execution.

24.1.4.1.4. Source Methods

Source methods can get a list of Source identifiers or request descriptions about Sources.

Example Source Methods

```
public Set<String> getSourceIds();
public SourceInfoResponse getSourceInfo(SourceInfoRequest sourceInfoRequest) throws
SourceUnavailableException;
```

24.1.4.1.5. Transform Methods

Transform methods provide convenience methods for using Metacard Transformers and Query Response Transformers.

Transform Methods

```
// Metacard Transformer
public BinaryContent transform(Metacard metacard, String transformerId, Map<String
,Serializable> requestProperties) throws CatalogTransformerException;

// Query Response Transformer
public BinaryContent transform(SourceResponse response, String transformerId, Map<String,
Serializable> requestProperties) throws CatalogTransformerException;
```

24.1.4.2. Implementing Catalog Methods

Query Response Transform Example

```
// inject CatalogFramework instance or retrieve an instance
private CatalogFramework catalogFramework;

public RSSEndpoint(CatalogFramework catalogFramework)
{
     this.catalogFramework = catalogFramework ;
     // implementation
}

 // Other implementation details ...

private void convert(QueryResponse queryResponse ) {
    // ...
    String transformerId = "rss";

    BinaryContent content = catalogFramework.transform(queryResponse, transformerId,
null);
```

```
    // ...

}
```

24.1.4.3. Dependency Injection

Using Blueprint or another injection framework, transformers can be injected from the OSGi Service Registry.

Blueprint Service Reference

```
<reference id="[[Reference Id" interface="DDF.catalog.transform.[[Transformer Interface
Name]]" filter="(shortname=[[Transformer Identifier]])" />
```

Each transformer has one or more `transform` methods that can be used to get the desired output.

Input Transformer Example

```
DDF.catalog.transform.InputTransformer inputTransformer = retrieveInjectedInstance() ;

Metacard entry = inputTransformer.transform(messageInputStream);
```

Metacard Transformer Example

```
DDF.catalog.transform.MetacardTransformer metacardTransformer = retrieveInjectedInstance
() ;

BinaryContent content = metacardTransformer.transform(metacard, arguments);
```

Query Response Transformer Example

```
DDF.catalog.transform.QueryResponseTransformer queryResponseTransformer =
retrieveInjectedInstance() ;

BinaryContent content = queryResponseTransformer.transform(sourceSesponse, arguments);
```

24.1.4.4. OSGi Service Registry

IMPORTANT	In the vast majority of cases, working with the OSGi Service Reference directly should be avoided. Instead, dependencies should be injected via a dependency injection framework like Blueprint.

Transformers are registered with the OSGi Service Registry. Using a `BundleContext` and a filter, references to a registered service can be retrieved.

OSGi Service Registry Reference Example

```
ServiceReference[] refs =
    bundleContext.getServiceReferences(DDF.catalog.transform.InputTransformer.class
.getName(),"(shortname=" + transformerId + ")");
InputTransformer inputTransformer = (InputTransformer) context.getService(refs[0]);
Metacard entry = inputTransformer.transform(messageInputStream);
```

24.2. Developing Metacard Types

Create custom Metacard types with Metacard Type definition files.

24.2.1. Metacard Type Definition File

To define Metacard Types, the definition file must have a `metacardTypes` key in the root object.

```
{
    "metacardTypes": [...]
}
```

The value of `metacardTypes` must be an array of Metacard Type Objects, which are composed of the `type` (required), `extendsTypes` (optional), and `attributes` (optional) keys.

Sample Top Level metacardTypes Definition

```
{
    "metacardTypes": [
        {
            "type": "my-metacard-type",
            "extendsTypes": ["core", "security"],
            "attributes": {...}
        }
    ]
}
```

The value of the `type` key is the name of the metacard type being defined. **This field is required.**

The value of the `extendsTypes` key is an array of metacard type names (strings) whose attributes you wish to include in your type. Valid Metacard Types already defined in the system or any Metacard Types already defined in this file work. Please note this section is evaluated from top to bottom so order any types used in other definitions above where they are used in the `extendsTypes` of other definitions. This key and value may be completely omitted to not extend any types.

The value of the `attributes` key is a map where each key is the name of an attribute type to include in

this metacard type and each value is a map with a single key named `required` and a boolean value. Required attributes are used for metacard validation - metacards that lack required attributes are flagged with validation errors. `attributes` may be completely omitted. `required` may be omitted.

Sample Complete metacardTypes Definition

```
{
    "metacardTypes": [
        {
            "type": "my-metacard-type",
            "attributes": {
                "resolution": {
                    "required": true
                },
                "target-areas": {
                    "required": false
                },
                "expiration": {},
                "point-of-contact": {
                    "required": true
                }
            }
        }
    ]
}
```

NOTE

The DDF basic metacard attribute types are added to custom metacard types by default. If any attribute types are required by a metacard type, just include them in the `attributes` map and set `required` to `true`, as shown in the above example with `point-of-contact`.

Multiple Metacard Types in a Single File

```
{
    "metacardTypes": [
        {
            "type": "my-metacard-type",
            "attributes": {
                "resolution": {
                    "required": true
                },
                "target-areas": {
                    "required": false
                }
            }
        },
```

```
        {
            "type": "another-metacard-type",
            "attributes": {
                "effective": {
                    "required": true
                },
                "resolution": {
                    "required": false
                }
            }
        }
    ]
}
```

24.3. Developing Metacard Validators

24.3.1. Metacard Validator Definition

Metacard Validator definitions are similar to the Validators definitions. To define Metacard Validators, your definition file must have a `metacardvalidators` key in the root object.

```
{
    "metacardvalidators": {...}
}
```

The value of `metacardvalidators` is a list of maps, where each map contains a key that is the name of the metacard type the corresponding validators apply to. Its value is a list of maps of the validator configuration parameters that specify which validators to configure for the given metacard type.

```
{
    "metacardvalidators": [{
        "metacard.type" : [{
            "validator": "requiredattributes",
            "key": "value"
        }]
      }]
}
```

IMPORTANT	The configuration for the metacard validator must always contain a `validator` key that specifies the type of validator to create. Additional key/value pairs may be required to configure the validator.

The `validator` key must have a value of one of the following:

- `requiredattributes` (checks that metacards of the specified type contain certain attributes)
  - Must add the key `requiredattributes` whose value is a list of strings naming the required attributes

Examples:

```
{
  "metacardvalidators": [{
    "fallback.common": [{
        "validator": "requiredattributes",
        "requiredattributes": [
          "id",
          "title",
          "attr"
        ]
    }]
  }]
}
```

24.4. Developing Global Attribute Validators

24.4.1. Global Attribute Validators File

To define Validators, the definition file must have a `validators` key in the root object.

```
{
    "validators": {...}
}
```

The value of `validators` is a map of the attribute name to a list of validators for that attribute.

```
{
    "validators": {
        "point-of-contact": [...]
    }
}
```

Each object in the list of validators is the validator name and list of arguments for that validator.

```
{
    "validators": {
        "point-of-contact": [
```

```
            {
                "validator": "pattern",
                "arguments": [".*regex.+\\s"]
            }
        ]
    }
}
```

WARNING	The value of the `arguments` key must always be an array of strings, even for numeric arguments, for example, `["1", "10"]`

The `validator` key must have a value of one of the following:

1. `validator` Possible Values

- `size` (validates the size of Strings, Arrays, Collections, and Maps)
  - `arguments`: (2) [integer: lower bound (inclusive), integer: upper bound (inclusive)]
    - lower bound must be greater than or equal to zero and the upper bound must be greater than or equal to the lower bound
- `pattern`
  - `arguments`: (1) [regular expression]
- `pastdate`
  - `arguments`: (0) [NO ARGUMENTS]
- `futuredate`
  - `arguments`: (0) [NO ARGUMENTS]
- `range`
  - (2) [number (decimal or integer): inclusive lower bound, number (decimal or integer): inclusive upper bound]
    - uses a default epsilon of 1E-6 on either side of the range to account for floating point representation inaccuracies
  - (3) [number (decimal or integer): inclusive lower bound, number (decimal or integer): inclusive upper bound, decimal number: epsilon (the maximum tolerable error on either side of the range)]
- `enumeration`
  - `arguments`: (unlimited) [list of strings: each argument is one case-sensitive, valid enumeration value]
- `relationship`
  - `arguments`: (4+) [attribute value or null, one of mustHave|cannotHave|canOnlyHave, target

attribute name, null or target attribute values as additional arguments]

- `match_any`
  - `validators`: (unlimited) [list of previously defined validators: valid if any validator succeeds]

Example Validator Definition

```
{
    "validators": {
        "title": [
            {
                "validator": "size",
                "arguments": ["1", "50"]
            },
            {
                "validator": "pattern",
                "arguments": ["\\D+"]
            }
        ],
        "created": [
            {
                "validator": "pastdate",
                "arguments": []
            }
        ],
        "expiration": [
            {
                "validator": "futuredate",
                "arguments": []
            }
        ],
        "page-count": [
            {
                "validator": "range",
                "arguments": ["1", "500"]
            }
        ],
        "temperature": [
            {
                "validator": "range",
                "arguments": ["12.2", "19.8", "0.01"]
            }
        ],
        "resolution": [
            {
                "validator": "enumeration",
```

```
                "arguments": ["1080p", "1080i", "720p"]
            }
        ],
        "datatype": [
            {
                "validator": "match_any",
                "validators": [
                    {
                        "validator": "range",
                        "arguments": ["1", "25"]
                    },
                    {
                        "validator": "enumeration",
                        "arguments": ["Collection", "Dataset", "Event"]
                    }
                ]
            }
        ],
        "topic.vocabulary": [
             {
                 "validator": "relationship",
                 "arguments": ["animal", "canOnlyHave", "topic.category", "cat", "dog",
"lizard"]
             }
         ]
    }
}
```

24.5. Developing Attribute Types

Create custom attribute types with Attribute Type definition files.

24.5.1. Attribute Type Definition File

To define Attribute Types, the definition file must have an `attributeTypes` key in the root object.

```
{
    "attributeTypes": {...}
}
```

The value of `attributeTypes` must be a map where each key is the attribute type's name and each value is a map that includes the data type and whether the attribute type is stored, indexed, tokenized, or multi-valued.

Attribute Types

```
{
    "attributeTypes": {
        "temperature": {
            "type": "DOUBLE_TYPE",
            "stored": true,
            "indexed": true,
            "tokenized": false,
            "multivalued": false
        }
    }
}
```

The attributes `stored`, `indexed`, `tokenized`, and `multivalued` must be included and must have a boolean value.

2. Required Attribute Definitions

`stored`

: If true, the value of the attribute should be stored in the underlying datastore. Some attributes may only be indexed or used in transit and do not need to be persisted.

`indexed`

: If true, then the value of the attribute should be included in the datastore's index and therefore be part of query evaluation.

`tokenized`

: Only applicable to STRING_TYPE attributes, if true then stopwords and punctuation are stripped prior to storing and/or indexing. If false, only an exact string matches.

`multi-valued`

: If true, then the attribute values are Lists of the attribute type rather than single values.

The `type` attribute must also be included and must have one of the allowed values:

3. `type` *Attribute Possible Values*

- `DATE_TYPE`
- `STRING_TYPE`
- `XML_TYPE`
- `LONG_TYPE`
- `BINARY_TYPE`

- `GEO_TYPE`
- `BOOLEAN_TYPE`
- `DOUBLE_TYPE`
- `FLOAT_TYPE`
- `INTEGER_TYPE`
- `OBJECT_TYPE`
- `SHORT_TYPE`

An example with multiple attributes defined:

Multiple Attributes Defined

```
{
    "attributeTypes": {
        "resolution": {
            "type": "STRING_TYPE",
            "stored": true,
            "indexed": true,
            "tokenized": false,
            "multivalued": false
        },
        "target-areas": {
            "type": "GEO_TYPE",
            "stored": true,
            "indexed": true,
            "tokenized": false,
            "multivalued": true
        }
    }
}
```

24.6. Developing Default Attribute Types

Create custom default attribute types.

24.6.1. Default Attribute Values

To define default attribute values, the definition file must have a `defaults` key in the root object.

```
{
    "defaults": [...]
```

```
}
```

The value of `defaults` is a list of objects where each object contains the keys `attribute`, `value`, and optionally `metacardTypes`.

```
{
    "defaults": [
        {
            "attribute": ...,
            "value": ...,
            "metacardTypes": [...]
        }
    ]
}
```

The value corresponding to the `attribute` key is the name of the attribute to which the default value is applied. The value corresponding to the `value` key is the default value of the attribute.

NOTE

The attribute's default value must be of the same type as the attribute, but it has to be written as a string (enclosed in quotation marks) in the JSON file.

Dates must be UTC datetimes in the ISO 8601 format, `yyyy-MM-ddTHH:mm:ssZ`

The `metacardTypes` key is optional. If it is left out, then the default attribute value is applied to every metacard that has that attribute. It can be thought of as a 'global' default value. If the `metacardTypes` key is included, then its value must be a list of strings where each string is the name of a metacard type. In this case, the default attribute value is applied only to metacards that match one of the types given in the list.

NOTE

In the event that an attribute has a 'global' default value as well as a default value for a specific metacard type, the default value for the specific metacard type is applied (the more specific default value wins).

Example:

```
{
    "defaults": [
        {
            "attribute": "title",
            "value": "Default Title"
        },
        {
            "attribute": "description",
            "value": "Default video description",
```

```
            "metacardTypes": ["video"]
        },
        {
            "attribute": "expiration",
            "value": "2020-05-06T12:00:00Z",
            "metacardTypes": ["video", "nitf"]
        },
        {
            "attribute": "frame-rate",
            "value": "30"
        }
    ]
}
```

24.7. Developing Attribute Injections

Attribute injections are defined attributes that are injected into all metacard types or into specific metacard types. This capability allows metacard types to be extended with new attributes.

24.7.1. Attribute Injection Definition

To define attribute injections, create a JSON file in the `<DDF_HOME>/etc/definitions` directory. The definition file must have an `inject` key in the root object.

Inject Key

```
{
  "inject": [...]
}
```

The value of `inject` is simply a list of objects where each object contains the key `attribute` and optionally `metacardTypes`.

Inject Values

```
{
  "inject": [
    {
      "attribute": ...,
      "metacardTypes": [...]
    }
  ]
}
```

The value corresponding to the `attribute` key is the name of the attribute to inject.

The `metacardTypes` key is optional. If it is left out, then the attribute is injected into every metacard type. In that case it can be thought of as a 'global' attribute injection. If the `metacardTypes` key is included, then its value must be a list of strings where each string is the name of a metacard type. In this case, the attribute is injected only into metacard types that match one of the types given in the list.

Global and Specific Inject Values

```
{
  "inject": [
    // Global attribute injection, all metacards
    {
      "attribute": "rating"
    },
    // Specific attribute injection, only "video" metacards
    {
      "attribute": "cloud-cover",
      "metacardTypes": "video"
    }
  ]
}
```

NOTE	Attributes must be registered in the attribute registry (see the `AttributeRegistry` interface) to be injected into metacard types. For example, attributes defined in JSON definition files are placed in the registry, so they can be injected.

Add a second key for `attributeTypes` to register the new types defined previously. For each attribute injections, specify the name and properties for that attribute.

- type: Data type of the possible values for this attribute.
- indexed: Boolean, attribute is indexed.
- stored: Boolean, attribute is stored.
- tokenized: Boolean, attribute is stored.
- multivalued: Boolean, attribute can hold multiple values.

Sample Attribute Injection File

```
{
  "inject": [
    // Global attribute injection, all metacards
    {
      "attribute": "rating"
    },
    // Specific attribute injection, only "video" metacards
    {
      "attribute": "cloud-cover",
```

```
      "metacardTypes": "video"
    }
  ],
  "attributeTypes": {
    "rating": {
    "type": "STRING_TYPE",
    "indexed": true,
    "stored": true,
    "tokenized": true,
    "multivalued": true
    },
    "cloud-cover": {
      "type": "STRING_TYPE",
      "indexed": true,
      "stored": true,
      "tokenized": true,
      "multivalued": false
    }
  }
}
```

24.8. Developing Endpoints

Custom endpoints can be created, if necessary. See Endpoints for descriptions of provided endpoints.

Complete the following procedure to create an endpoint.

1. Create a Java class that implements the endpoint's business logic. Example: Creating a web service that external clients can invoke.
2. Add the endpoint's business logic, invoking `CatalogFramework` calls as needed.
3. Import the DDF packages to the bundle's manifest for run-time (in addition to any other required packages):
   `Import-Package: ddf.catalog, ddf.catalog.*`
4. Retrieve an instance of `CatalogFramework` from the OSGi registry. (Refer to OSGi Basics - Service Registry for examples.)
5. Deploy the packaged service to DDF. (Refer to OSGi Basics - Bundles.)

NOTE	It is recommended to use the maven bundle plugin to create the Endpoint bundle's manifest as opposed to directly editing the manifest file.

TIP	**No implementation of an interface is required** Unlike other DDF components that require you to implement a standard interface, no implementation of an interface is required in order to create an endpoint.

Table 73. Common Endpoint Business Logic

Methods	Use
`Ingest`	Add, modify, and remove metadata using the ingest-related `CatalogFramework` methods: create, update, and delete.
`Query`	Request metadata using the `query` method.
`Source`	Get available `Source` information.
`Resource`	Retrieve resources referenced in Metacards from Sources.
`Transform`	Convert common Catalog Framework data types to and from other data formats.

24.9. Developing Input Transformers

DDF supports the creation of custom input transformers for use cases not covered by the included implementations.

Creating a custom input Transformer:

1. Create a new Java class that implements `ddf.catalog.transform.InputTransformer`.
   `public class SampleInputTransformer implements ddf.catalog.transform.InputTransformer`
2. Implement the transform methods.
   `public Metacard transform(InputStream input) throws IOException, CatalogTransformerException`
   `public Metacard transform(InputStream input, String id) throws IOException, CatalogTransformerException`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.transform`
4. Create an OSGi descriptor file to communicate with the OSGi Service Registry (described in the OSGi Basics section). Export the service to the OSGi Registry and declare service properties.

   Input Transformer Blueprint Descriptor Example

```
...
<service ref="SampleInputTransformer" interface=
"ddf.catalog.transform.InputTransformer">
    <service-properties>
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Input Transformer]]" />
        <entry key="description" value="[[A new transformer for metacard input.]]" />
    </service-properties>
</service>
```

```
...
```

Table 74. Input Transformer Variable Descriptions / Blueprint Service Properties

Key	Description of Value	Example
`shortname`	(Required) An abbreviation for the return-type of the `BinaryContent` being sent to the user.	`atom`
`title`	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	`Atom Entry Transformer Service`
`description`	(Optional) A short, human-readable description that describes the functionality of the service and the output.	`This service converts a single metacard xml document to an atom entry element.`

5. Deploy OSGi Bundle to OSGi runtime.

24.9.1. Create an XML Input Transformer using SaxEventHandlers

For a transformer to transform XML, (as opposed to JSON or a Word document, for example) there is a simpler solution than fully implementing a `MetacardValidator`. DDF includes an extensible, configurable `XmlInputTransformer`. This transformer can be instantiated via blueprint as a managed service factory and configured via metatype. The `XmlInputTransformer` takes a configuration of `SaxEventHandlers`. A `SaxEventHandler` is a class that handles SAX Events (a very fast XML parser) to parse metadata and create metacards. Any number of `SaxEventHandlers` can be implemented and included in the `XmlInputTransformer` configuration. See the `catalog-transformer-streaming-impl` bundle for examples (`XmlSaxEventHandlerImpl` which parses the DDF Metacard XML Metadata and the `GmlHandler` which parses GML 2.0) Each `SaxEventHandler` implementation has a `SaxEventHandlerFactory` associated with it. The `SaxEventHandlerFactory` is responsible for instantiating new `SaxEventHandlers` - each transform request gets a new instance of `XmlInputTransformer` and set of `SaxEventHandlers` to be *thread-* and *state-safe*.

The following diagrams intend to clarify implementation details:

The `XmlInputTransformer` Configuration diagram shows the `XmlInputTransformer` configuration, which is configured using the metatype and has the `SaxEventHandlerFactory` ids. Then, when a transform request is received, the `ManagedServiceFactory` instantiates a new `XmlInputTransformer`. This `XmlInputTransformer` then instantiates a new `SaxEventHandlerDelegate` with the configured `SaxEventHandlersFactory` ids. The factories all in turn instantiate a `SaxEventHandler`. Then, the `SaxEventHandlerDelegate` begins parsing the XML input document, handing the SAX Events off to each `SaxEventHandler`, which handle them if they can. After parsing is finished, each `SaxEventHandler` returns a list of `Attributes` to the `SaxEventHandlerDelegate` and `XmlInputTransformer` which add the attributes to the metacard and then return the fully constructed metacard.

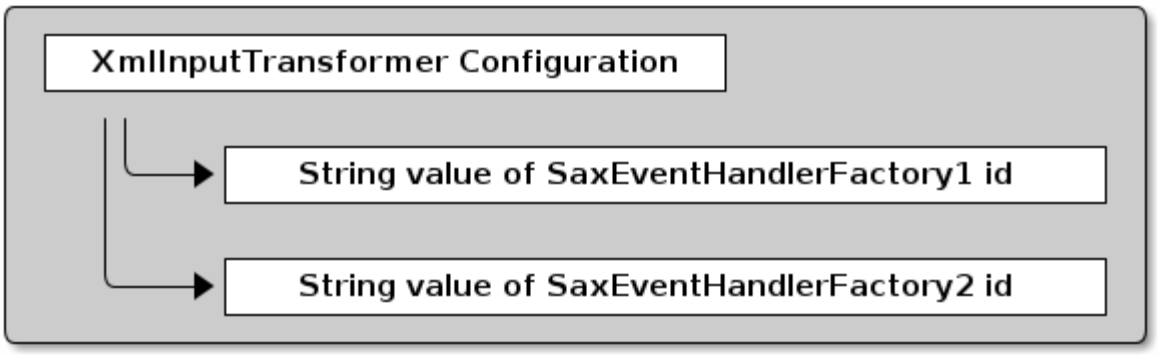


Figure 31. `XMLInputTransformer` *Configuration*

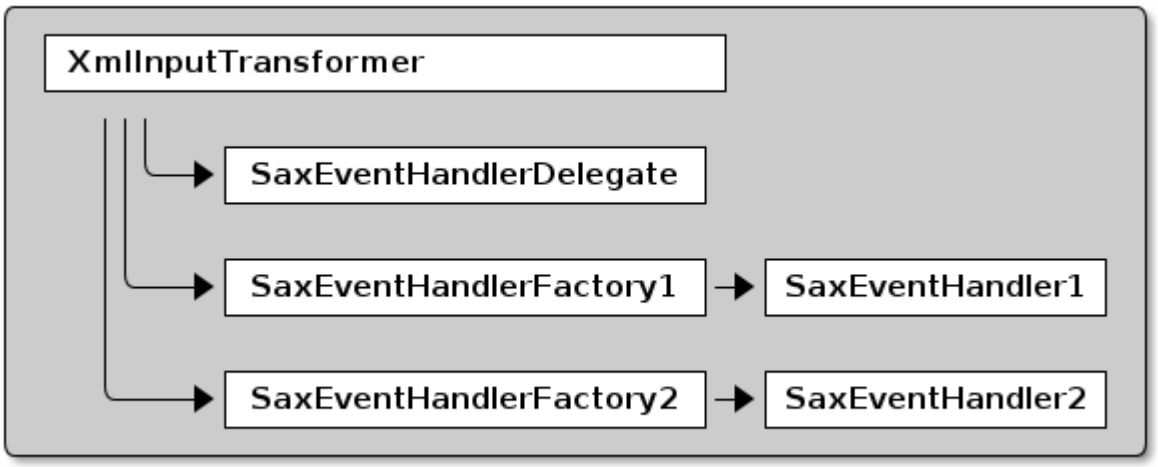


Figure 32. `XMLInputTransformer` `SaxEventHandlerDelegate` *Configuration*

For more specific details, see the Javadoc for the `org.codice.ddf.transformer.xml.streaming.*` package. Additionally, see the source code for the `org.codice.ddf.transformer.xml.streaming.impl.GmlHandler.java`, `org.codice.ddf.transformer.xml.streaming.impl.GmlHandlerFactory`, `org.codice.ddf.transformer.xml.streaming.impl.XmlInputTransformerImpl`, and `org.codice.ddf.transformer.xml.streaming.impl.XmlInputTransformerImplFactory`.

NOTE

1. The `XmlInputTransformer` & `SaxEventHandlerDelegate` create and configure themselves based on String matches of the configuration ids with the `SaxEventHandlerFactory` ids, so ensure these match.
2. The `XmlInputTransformer` uses a `DynamicMetacardType`. This is pertinent because a metacards attributes are only stored in the `CatalogProvider` if they are declared on the `MetacardType`. Since the `DynamicMetacardType` is constructed dynamically, attributes are declared by the `SaxEventHandlerFactory` that parses them, as opposed to the `MetacardType`. See

> `org.codice.ddf.transformer.xml.streaming.impl.XmlSaxEventHandlerFactoryImpl.java` vs `ddf.catalog.data.impl.BasicTypes.java`

24.9.2. Create an Input Transformer Using Apache Camel

Alternatively, make an Apache Camel route in a blueprint file and deploy it using a feature file or via hot deploy.

24.9.2.1. Input Transformer Design Pattern (Camel)

Follow this design pattern for compatibility:

From

When using **from**, `catalog:inputtransformer?id=text/xml`, an Input Transformer is created and registered in the OSGi registry with an id of `text/xml`.

To

When using **to**, `catalog:inputtransformer?id=text/xml`, an Input Transformer with an id matching text/xml is discovered from the OSGi registry and invoked.

Table 75. InputTransformer Message Formats

Exchange Type	Field	Type
Request (comes from `<from>` in the route)	body	`java.io.InputStream`
Response (returned after called via `<to>` in the route)	body	`ddf.catalog.data.Metacard`

TIP Its always a good idea to wrap the `mimeType` value with the RAW parameter as shown in the example above. This ensures that the value is taken exactly as is, and is especially useful when you are using special characters.

InputTransformer Creation Example

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <route>
            <from uri="catalog:inputtransformer?mimeType=RAW(id=text/xml;id=vehicle)"/>
            <to uri="xslt:vehicle.xslt" /> <!-- must be on classpath for this bundle -->
            <to uri=
"catalog:inputtransformer?mimeType=RAW(id=application/json;id=geojson)" />
        </route>
    </camelContext>
</blueprint>
```

InputTransformer Creation Details

1. Defines this as an Apache Aries blueprint file.
2. Defines the Apache Camel context that contains the route.
3. Defines start of an Apache Camel route.
4. Defines the endpoint/consumer for the route. In this case it is the DDF custom catalog component that is an `InputTransformer` registered with an id of `text/xml;id=vehicle` meaning it can transform an `InputStream` of vehicle data into a metacard. **Note that the specified XSL stylesheet must be on the classpath of the bundle that this blueprint file is packaged in.**
5. Defines the XSLT to be used to transform the vehicle input into GeoJSON format using the Apache Camel provided XSLT component.
6. Defines the route node that accepts GeoJSON formatted input and transforms it into a Mmtacard, using the DDF custom catalog component that is an InputTransformer registered with an id of application/json;id=geojson.

NOTE	An example of using an Apache Camel route to define an `InputTransformer` in a blueprint file and deploying it as a bundle to an OSGi container can be found in the DDF SDK examples at `DDF/sdk/sample-transformers/xslt-identity-input-transformer`

24.9.3. Input Transformer Boot Service Flag

The `org.codice.ddf.platform.bootflag.BootServiceFlag` service with a service property of `id=inputTransformerBootFlag` is used to indicate certain Input Transformers are ready in the system. Adding an Input Transformers ID to a new or existing JSON file under `<DDF_HOME>/etc/transformers` causes the service to wait for an Input Transformer with the given ID.

24.10. Developing Metacard Transformers

In general, a `MetacardTransformer` is used to transform a `Metacard` into some desired format useful to the end user or as input to another process. Programmatically, a `MetacardTransformer` transforms a `Metacard` into a `BinaryContent` instance, which translates the `Metacard` into the desired final format. Metacard transformers can be used through the Catalog Framework `transform` convenience method or requested from the OSGi Service Registry by endpoints or other bundles.

24.10.1. Creating a New Metacard Transformer

Existing metacard transformers are written as Java classes, and these steps walk through the steps to create a custom metacard transformer.

1. Create a new Java class that implements `ddf.catalog.transform.MetacardTransformer`.
   `public class SampleMetacardTransformer implements ddf.catalog.transform.MetacardTransformer`
2. Implement the `transform` method.
   `public BinaryContent transform(Metacard metacard, Map<String, Serializable> arguments) throws`

`CatalogTransformerException`

a. `transform` must return a `Metacard` or throw an exception. It cannot return null.

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
`Import-Package: ddf.catalog,ddf.catalog.transform`

4. Create an OSGi descriptor file to communicate with the OSGi Service registry (described in the OSGi Basics section). Export the service to the OSGi registry and declare service properties.

Metacard Transformer Blueprint Descriptor Example

```
...
<service ref="SampleMetacardTransformer" interface=
"ddf.catalog.transform.MetacardTransformer">
    <service-properties>
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Metacard Transformer]]" />
        <entry key="description" value="[[A new transformer for metacards.]]" />
    </service-properties>
</service>
...
```

5. Deploy OSGi Bundle to OSGi runtime.

Table 76. Metacard Transformer Blueprint Service Properties / Variable Descriptions

Key	Description of Value	Example
`shortname`	(Required) An abbreviation for the return type of the BinaryContent being sent to the user.	atom
`title`	(Optional) A user-readable title that describes (in greater detail than the shortname) the service.	Atom Entry Transformer Service
`description`	(Optional) A short, human-readable description that describes the functionality of the service and the output.	This service converts a single metacard xml document to an atom entry element.

24.11. Developing Query Response Transformers

A Query Response Transformer is used to transform a List of Results from a `SourceResponse`. Query Response Transformers can be used through the Catalog transform convenience method or requested from the OSGi Service Registry by endpoints or other bundles.

1. Create a new Java class that implements `ddf.catalog.transform.QueryResponseTransformer`.
   `public class SampleResponseTransformer implements ddf.catalog.transform.QueryResponseTransformer`
2. Implement the `transform` method.
   `public BinaryContent transform(SourceResponse upstreamResponse, Map<String, Serializable> arguments) throws CatalogTransformerException`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog, ddf.catalog.transform`
4. Create an OSGi descriptor file to communicate with the OSGi Service Registry (described in OSGi Basics). Export the service to the OSGi registry and declare service properties.
5. Deploy OSGi Bundle to OSGi runtime.

Query Response Transformer Blueprint Descriptor Example

```
...
<service ref="SampleResponseTransformer" interface=
"ddf.catalog.transform.QueryResponseTransformer">
    <service-properties>
        <entry key="id" value="[[sampleId]]" />
        <entry key="shortname" value="[[sampletransform]]" />
        <entry key="title" value="[[Sample Response Transformer]]" />
        <entry key="description" value="[[A new transformer for response queues.]]" />
    </service-properties>
</service>
...
```

Table 77. Query Response Transformer Blueprint Service Properties / Variable Descriptions

Key	Description of Value	Example
`id`	A unique identifier to target a specific query response transformer.	atom
`shortname`	An abbreviation for the return type of the BinaryContent being sent to the user.	atom
`title`	A user-readable title that describes (in greater detail than the shortname) the service.	Atom Entry Transformer Service

Key	Description of Value	Example
`description`	A short, human-readable description that describes the functionality of the service and the output.	*This service converts a single metacard xml document to an atom entry element.*

24.12. Developing Sources

Sources are components that enable DDF to talk to back-end services. They let DDF perform query and ingest operations on catalog stores and query operations on federated sources.

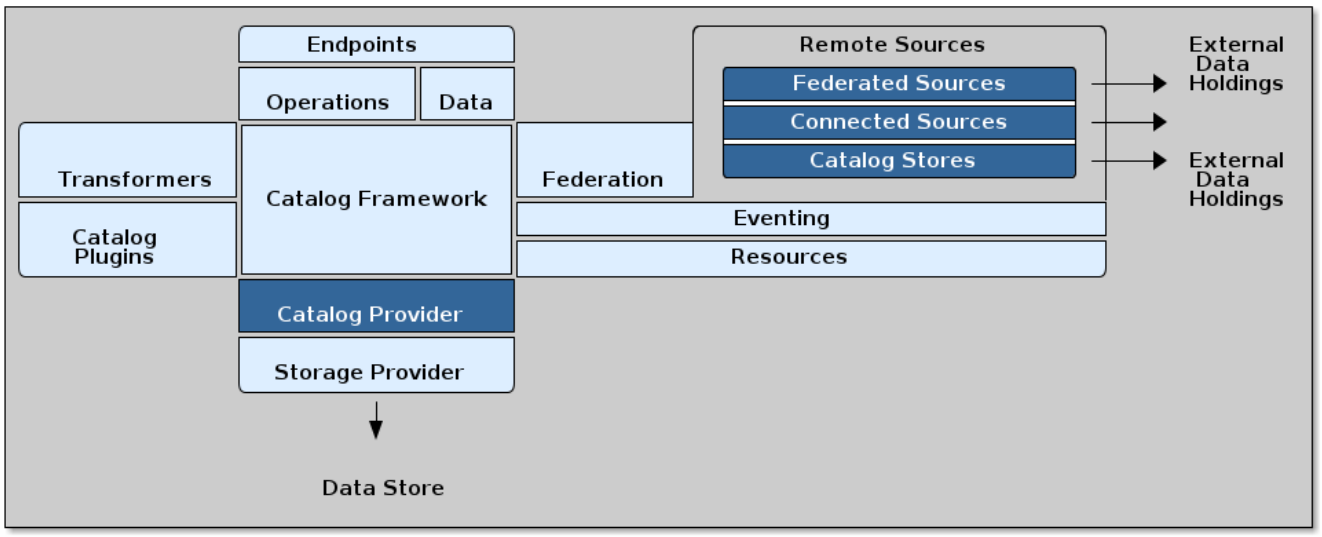


Figure 33. Source Architecture

24.12.1. Implement a Source Interface

There are three types of sources that can be created to perform query operations. All of these sources must also be able to return their availability and the list of content types currently stored in their back-end data stores.

Catalog Provider

`ddf.catalog.source.CatalogProvider` *is used to communicate with back-end storage and allows for Query and Create/Update/Delete operations.*

Federated Source

`ddf.catalog.source.FederatedSource` *is used to communicate with remote systems and only allows query operations.*

Connected Source

`ddf.catalog.source.ConnectedSource` *is similar to a Federated Source with the following exceptions:*

- *Queried on all local queries*
- `SiteName` *is hidden (masked with the DDF sourceId) in query results*
- `SiteService` *does not show this Source's information separate from DDF's.*

Catalog Store

`catalog.store.interface` *is used to store data.*

The procedure for implementing any of the source types follows a similar format:

1. Create a new class that implements the specified Source interface, the `ConfiguredService` and the required methods.
2. Create an OSGi descriptor file to communicate with the OSGi registry. (Refer to OSGi Services.)
   a. Import DDF packages.
   b. Register source class as service to the OSGi registry.
3. Deploy to DDF.

IMPORTANT	The `factory-pid` property of the metatype must contain one of the following in the name: service, Service, source, Source

NOTE	Remote sources currently extend the `ResourceReader` interface. However, a `RemoteSource` is not treated as a `ResourceReader`. The `getSupportedSchemes()` method should never be called on a `RemoteSource`, thus the suggested implementation for a `RemoteSource` is to return an empty set. The `retrieveResource( ⋯ )` and `getOptions( ⋯ )` methods are called and MUST be properly implemented by a `RemoteSource`.

24.12.1.1. Developing Catalog Providers

Create a custom implementation of a catalog provider.

1. Create a Java class that implements `CatalogProvider`.
   `public class TestCatalogProvider implements ddf.catalog.source.CatalogProvider`
2. Implement the required methods from the `ddf.catalog.source.CatalogProvider` interface.
   `public CreateResponse create(CreateRequest createRequest) throws IngestException; public UpdateResponset update(UpdateRequest updateRequest) throws IngestException; public DeleteResponse delete(DeleteRequest deleteRequest) throws IngestException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog, ddf.catalog.source`
4. Export the service to the OSGi registry.

Catalog Provider Blueprint example

```
<service ref="TestCatalogProvider" interface="ddf.catalog.source.CatalogProvider" />
```

See the existing Catalog Provider list for examples of Catalog Providers included in DDF.

24.12.1.2. Developing Federated Sources

1. Create a Java class that implements `FederatedSource` and `ConfiguredService`.
   `public class TestFederatedSource implements ddf.catalog.source.FederatedSource, ddf.catalog.service.ConfiguredService`
2. Implement the required methods of the `ddf.catalog.source.FederatedSource` and `ddf.catalog.service.ConfiguredService` interfaces.
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog, ddf.catalog.source`
4. Export the service to the OSGi registry.

Federated Source Blueprint example

```
<service ref="TestFederatedSource" interface="ddf.catalog.source.FederatedSource" />
```

24.12.1.3. Developing Connected Sources

Create a custom implementation of a connected source.

1. Create a Java class that implements `ConnectedSource` and `ConfiguredService`.
   `public class TestConnectedSource implements ddf.catalog.source.ConnectedSource, ddf.catalog.service.ConfiguredService`
2. Implement the required methods of the `ddf.catalog.source.ConnectedSource` and `ddf.catalog.service.ConfiguredService` interfaces.
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog, ddf.catalog.source`
4. Export the service to the OSGi registry.

Connected Source Blueprint example

```
<service ref="TestConnectedSource" interface="ddf.catalog.source.ConnectedSource" />
```

IMPORTANT	In some Providers that are created, there is a need to make Web Service calls through JAXB clients. It is best to NOT create a JAXB client as a global variable.

> There may be intermittent failures with the creation of Providers and federated sources when clients are created in this manner. To avoid this issue, create any JAXB within the methods requiring it.

24.12.1.4. Exception Handling

In general, sources should only send information back related to the call, not implementation details.

24.12.1.4.1. Exception Examples

Follow these guidelines for effective exception handling:

- Use a "Site XYZ not found" message rather than the full stack trace with the original site not found exception.
- If the caller issues a malformed search request, return an error describing the right form, or specifically what was not recognized in the request. Do not return the exception and stack trace where the parsing broke.
- If the caller leaves something out, do not return the null pointer exception with a stack trace, rather return a generic exception with the message "xyz was missing."

24.12.1.4.2. External Resources for Developing Sources

- Three Rules for Effective Exception Handling.

24.13. Developing Catalog Plugins

Catalog Plugins extend the functionality of the Catalog Framework by performing actions at specified times during a transaction. Plugin interfaces are located in the Catalog Core API. By implementing a plugin interface, actions can be performed at the desired time.

The following types of plugins can be created:

Table 78. Plugin Interfaces

Plugin Type	Plugin Interface	Invocation Order
Pre-Authorization	`ddf.catalog.plugin.PreAuthorizationPlugin`	Before any security rules are applied.
Policy	`ddf.catalog.plugin.PolicyPlugin`	After pre-authorization plugins, but before other catalog plugins to establish the policy for requests/responses.
Access	`ddf.catalog.plugin.AccessPlugin`	Directly after any policy plugins

Plugin Type	Plugin Interface	Invocation Order
Pre-Ingest	`ddf.catalog.plugin.PreIngestPlugin`	Before the Create/Update/Delete method is sent to the Catalog Provider.
Post-Ingest	`ddf.catalog.plugin.PostIngestPlugin`	After the Create/Update/Delete method is sent to the Catalog Provider.
Pre-Query	`ddf.catalog.plugin.PreQueryPlugin`	Prior to the Query/Read method being sent to the Source.
Post-Query	`ddf.catalog.plugin.PostQueryPlugin`	After results have been retrieved from the query but before they are posted to the Endpoint.
Pre-Federated-Query	`ddf.catalog.plugin.PreFederatedQueryPlugin`	Before a federated query is executed.
Post-Federated-Query	`ddf.catalog.plugin.PostFederatedQueryPlugin`	After a federated query has been executed.
Pre-Resource	`ddf.catalog.plugin.PreResourcePlugin`	Prior to a Resource being retrieved.
Post-Resource	`ddf.catalog.plugin.PostResourcePlugin`	After a Resource is retrieved, but before it is sent to the Endpoint.
Pre-Create Storage	`ddf.catalog.content.plugin.PreCreateStoragePlugin`	Experimental Before an item is created in the content repository.
Post-Create Storage	`ddf.catalog.content.plugin.PostCreateStoragePlugin`	Experimental After an item is created in the content repository.
Pre-Update Storage	`ddf.catalog.content.plugin.PreUpdateStoragePlugin`	Experimental Before an item is updated in the content repository.
Post-Update Storage	`ddf.catalog.content.plugin.PostUpdateStoragePlugin`	Experimental After an item is updated in the content repository.

Plugin Type	Plugin Interface	Invocation Order
Pre-Subscription	`ddf.catalog.plugin.PreSubscriptionPlugin`	Prior to a Subscription being created or updated.
Pre-Delivery	`ddf.catalog.plugin.PreDeliveryPlugin`	Prior to the delivery of a Metacard when an event is posted.

24.13.1. Implementing Catalog Plugins

The procedure for implementing any of the plugins follows a similar format:

1. Create a new class that implements the specified plugin interface.
2. Implement the required methods.
3. Create an OSGi descriptor file to communicate with the OSGi registry.
   a. Register the plugin class as a service to OSGi registry.
4. Deploy to DDF.

NOTE

Plugin Performance Concerns

Plugins should include a check to determine if requests are local or not. It is usually preferable to take no action on non-local requests.

TIP

Refer to the Javadoc for more information on all Requests and Responses in the `ddf.catalog.operation` and `ddf.catalog.event` packages.

24.13.1.1. Catalog Plugin Failure Behavior

In the event that this Catalog Plugin cannot operate but does not wish to fail the transaction, a `PluginExecutionException` should be thrown. If processing is to be explicitly stopped, a `StopProcessingException` should be thrown. For any other exceptions, the Catalog should "fail fast" and cancel the Operation.

24.13.1.2. Implementing Pre-Ingest Plugins

Develop a custom Pre-Ingest Plugin.

1. Create a Java class that implements `PreIngestPlugin`.
   `public class SamplePreIngestPlugin implements ddf.catalog.plugin.PreIngestPlugin`
2. Implement the required methods.
   - `public CreateRequest process(CreateRequest input) throws PluginExecutionException, StopProcessingException;`

- `public UpdateRequest process(UpdateRequest input) throws PluginExecutionException, StopProcessingException;`
- `public DeleteRequest process(DeleteRequest input) throws PluginExecutionException, StopProcessingException;`

3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin`
4. Export the service to the OSGi registry.
   Blueprint descriptor example `<service ref="SamplePreIngestPlugin" interface="ddf.catalog.plugin.PreIngestPlugin" />`

24.13.1.3. Implementing Post-Ingest Plugins

Develop a custom Post-Ingest Plugin.

1. Create a Java class that implements `PostIngestPlugin`.
   `public class SamplePostIngestPlugin implements ddf.catalog.plugin.PostIngestPlugin`
2. Implement the required methods.
   - `public CreateResponse process(CreateResponse input) throws PluginExecutionException;`
   - `public UpdateResponse process(UpdateResponse input) throws PluginExecutionException;`
   - `public DeleteResponse process(DeleteResponse input) throws PluginExecutionException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin`
4. Export the service to the OSGi registry.
   Blueprint descriptor example `<service ref="SamplePostIngestPlugin" interface="ddf.catalog.plugin.PostIngestPlugin" />`

24.13.1.4. Implementing Pre-Query Plugins

Develop a custom Pre-Query Plugin

1. Create a Java class that implements `PreQueryPlugin`.
   `public class SamplePreQueryPlugin implements ddf.catalog.plugin.PreQueryPlugin`
2. Implement the required method.
   `public QueryRequest process(QueryRequest input) throws PluginExecutionException, StopProcessingException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin`
4. Export the service to the OSGi registry.

```
<service ref="SamplePreQueryPlugin" interface="ddf.catalog.plugin.PreQueryPlugin" />
```

24.13.1.5. Implementing Post-Query Plugins

Develop a custom Post-Query Plugin

1. Create a Java class that implements `PostQueryPlugin`.
   ```
   public class SamplePostQueryPlugin implements ddf.catalog.plugin.PostQueryPlugin
   ```
2. Implement the required method.
   ```
   public QueryResponse process(QueryResponse input) throws PluginExecutionException, StopProcessingException;
   ```
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   ```
   Import-Package: ddf.catalog,ddf.catalog.plugin
   ```
4. Export the service to the OSGi registry.
   ```
   <service ref="SamplePostQueryPlugin" interface="ddf.catalog.plugin.PostQueryPlugin" />
   ```

24.13.1.6. Implementing Pre-Delivery Plugins

Develop a custom Pre-Delivery Plugin.

1. Create a Java class that implements `PreDeliveryPlugin`.
   ```
   public class SamplePreDeliveryPlugin implements ddf.catalog.plugin.PreDeliveryPlugin
   ```
2. Implement the required methods.
   ```
   public Metacard processCreate(Metacard metacard) throws PluginExecutionException, StopProcessingException; public Update processUpdateMiss(Update update) throws PluginExecutionException, StopProcessingException;
   ```
   - ```
     public Update processUpdateHit(Update update) throws PluginExecutionException, StopProcessingException;
     ```
   - ```
     public Metacard processCreate(Metacard metacard) throws PluginExecutionException, StopProcessingException;
     ```
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   ```
   Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation,ddf.catalog.event
   ```
4. Export the service to the OSGi registry.
   Blueprint descriptor example
   ```
   <service ref="SamplePreDeliveryPlugin" interface="ddf.catalog.plugin.PreDeliveryPlugin" />
   ```

24.13.1.7. Implementing Pre-Subscription Plugins

Develop a custom Pre-Subscription Plugin.

1. Create a Java class that implements `PreSubscriptionPlugin`.

`public class SamplePreSubscriptionPlugin implements ddf.catalog.plugin.PreSubscriptionPlugin`

2. Implement the required method.
   - `public Subscription process(Subscription input) throws PluginExecutionException, StopProcessingException;`

24.13.1.8. Implementing Pre-Resource Plugins

Develop a custom Pre-Resource Plugin.

1. Create a Java class that implements `PreResourcePlugin`. `public class SamplePreResourcePlugin implements ddf.catalog.plugin.PreResourcePlugin`
2. Implement the required method.
   - `public ResourceRequest process(ResourceRequest input) throws PluginExecutionException, StopProcessingException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation`
4. Export the service to the OSGi registry. .Blueprint descriptor example

```
<service ref="SamplePreResourcePlugin" interface="ddf.catalog.plugin.PreResourcePlugin"
/>
```

24.13.1.9. Implementing Post-Resource Plugins

Develop a custom Post-Resource Plugin.

1. Create a Java class that implements `PostResourcePlugin`.
   `public class SamplePostResourcePlugin implements ddf.catalog.plugin.PostResourcePlugin`
2. Implement the required method.
   - `public ResourceResponse process(ResourceResponse input) throws PluginExecutionException, StopProcessingException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation`
4. Export the service to the OSGi registry.

Blueprint descriptor example

```
<]]" inter"[[SamplePostResourcePlugin" interface="ddf.catalog.plugin.PostResourcePlugin"
/>
```

24.13.1.10. Implementing Policy Plugins

Develop a custom Policy Plugin.

1. Create a Java class that implements `PolicyPlugin`.
   `public class SamplePolicyPlugin implements ddf.catalog.plugin.PolicyPlugin`
2. Implement the required methods.
   - `PolicyResponse processPreCreate(Metacard input, Map<String, Serializable> properties) throws StopProcessingException;`
   - `PolicyResponse processPreUpdate(Metacard input, Map<String, Serializable> properties) throws StopProcessingException;`
   - `PolicyResponse processPreDelete(String attributeName, List<Serializable> attributeValues, Map<String, Serializable> properties) throws StopProcessingException;`
   - `PolicyResponse processPreQuery(Query query, Map<String, Serializable> properties) throws StopProcessingException;`
   - `PolicyResponse processPostQuery(Result input, Map<String, Serializable> properties) throws StopProcessingException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation`
4. Export the service to the OSGi registry.
   Blueprint descriptor example
   `<]]" inter"[[SamplePolicyPlugin" interface="ddf.catalog.plugin.PolicyPlugin" />`

24.13.1.11. Implementing Access Plugins

Develop a custom Access Plugin.

1. Create a Java class that implements `AccessPlugin`.
   `public class SamplePostResourcePlugin implements ddf.catalog.plugin.AccessPlugin`
2. Implement the required methods.
   - `CreateRequest processPreCreate(CreateRequest input) throws StopProcessingException;`
   - `UpdateRequest processPreUpdate(UpdateRequest input) throws StopProcessingException;`
   - `DeleteRequest processPreDelete(DeleteRequest input) throws StopProcessingException;`
   - `QueryRequest processPreQuery(QueryRequest input) throws StopProcessingException;`
   - `QueryResponse processPostQuery(QueryResponse input) throws StopProcessingException;`
3. Import the DDF interface packages to the bundle manifest (in addition to any other required packages).
   `Import-Package: ddf.catalog,ddf.catalog.plugin,ddf.catalog.operation`
4. Export the service to the OSGi registry.

Blueprint descriptor example

```
<]]" inter"[[SampleAccessPlugin" interface="ddf.catalog.plugin.AccessPlugin" />
```

24.14. Developing Token Validators

Token validators are used by the Security Token Service (STS) to validate incoming token requests. The `TokenValidator` CXF interface must be implemented by all custom token validators. The `canHandleToken` and `validateToken` methods must be overridden. The `canHandleToken` method should return true or false based on the `ValueType` value of the token that the validator is associated with. The validator may be able to handle any number of different tokens that you specify. The `validateToken` method returns a `TokenValidatorResponse` object that contains the `Principal` of the identity being validated and also validates the `ReceivedToken` object collected from the RST (`RequestSecurityToken`) message.

24.15. Developing STS Claims Handlers

Develop a custom claims handler to retrieve attributes from an external attribute store.

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

The following steps define the procedure for adding a custom claims handler to the STS.

1. The new claims handler must implement the `org.apache.cxf.sts.claims.ClaimsHander` interface.

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership. The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */

package org.apache.cxf.sts.claims;
```

```
import java.net.URI;
import java.util.List;

/**
 * This interface provides a pluggable way to handle Claims.
 */
public interface ClaimsHandler {

    List<URI> getSupportedClaimTypes();

    ClaimCollection retrieveClaimValues(RequestClaimCollection claims,
ClaimsParameters parameters);

}
```

2. Expose the new claims handler as an OSGi service under the `org.apache.cxf.sts.claims.ClaimsHandler` interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

    <bean id="CustomClaimsHandler" class=
"security.sts.claimsHandler.CustomClaimsHandler" />

    <service ref="customClaimsHandler" interface=
"org.apache.cxf.sts.claims.ClaimsHandler"/>

</blueprint>
```

3. Deploy the bundle.

If the new claims handler is hitting an external service that is secured with SSL/TLS, a developer may need to add the root CA of the external site to the DDF trustStore and add a valid certificate into the DDF keyStore. For more information on certificates, refer to Configuring a Java Keystore for Secure Communications.

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

STS WS-Trust WSDL Document

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:tns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
xmlns:wstrust="http://docs.oasis-open.org/ws-sx/ws-trust/200512/" xmlns:wsdl=
```

```
"http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsap10="http://www.w3.org/2006/05/addressing/wsdl" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp=
"http://www.w3.org/ns/ws-policy" xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsam=
"http://www.w3.org/2007/05/addressing/metadata" targetNamespace="http://docs.oasis-
open.org/ws-sx/ws-trust/200512/">
    <wsdl:types>
        <xs:schema elementFormDefault="qualified" targetNamespace="http://docs.oasis-
open.org/ws-sx/ws-trust/200512">
            <xs:element name="RequestSecurityToken" type=
"wst:AbstractRequestSecurityTokenType"/>
            <xs:element name="RequestSecurityTokenResponse" type=
"wst:AbstractRequestSecurityTokenType"/>
            <xs:complexType name="AbstractRequestSecurityTokenType">
                <xs:sequence>
                    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="Context" type="xs:anyURI" use="optional"/>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:complexType>
            <xs:element name="RequestSecurityTokenCollection" type=
"wst:RequestSecurityTokenCollectionType"/>
            <xs:complexType name="RequestSecurityTokenCollectionType">
                <xs:sequence>
                    <xs:element name="RequestSecurityToken" type=
"wst:AbstractRequestSecurityTokenType" minOccurs="2" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:element name="RequestSecurityTokenResponseCollection" type=
"wst:RequestSecurityTokenResponseCollectionType"/>
            <xs:complexType name="RequestSecurityTokenResponseCollectionType">
                <xs:sequence>
                    <xs:element ref="wst:RequestSecurityTokenResponse" minOccurs="1"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
    <!-- WS-Trust defines the following GEDs -->
    <wsdl:message name="RequestSecurityTokenMsg">
        <wsdl:part name="request" element="wst:RequestSecurityToken"/>
    </wsdl:message>
    <wsdl:message name="RequestSecurityTokenResponseMsg">
        <wsdl:part name="response" element="wst:RequestSecurityTokenResponse"/>
    </wsdl:message>
```

```
    <wsdl:message name="RequestSecurityTokenCollectionMsg">
        <wsdl:part name="requestCollection" element="wst:RequestSecurityTokenCollection
"/>
    </wsdl:message>
    <wsdl:message name="RequestSecurityTokenResponseCollectionMsg">
        <wsdl:part name="responseCollection" element=
"wst:RequestSecurityTokenResponseCollection"/>
    </wsdl:message>
    <!-- This portType an example of a Requestor (or other) endpoint that
         Accepts SOAP-based challenges from a Security Token Service -->
    <wsdl:portType name="WSSecurityRequestor">
        <wsdl:operation name="Challenge">
            <wsdl:input message="tns:RequestSecurityTokenResponseMsg"/>
            <wsdl:output message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
    </wsdl:portType>
    <!-- This portType is an example of an STS supporting full protocol -->
    <wsdl:portType name="STS">
        <wsdl:operation name="Cancel">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Cancel" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/CancelFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
        <wsdl:operation name="Issue">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTRC/IssueFinal" message="tns:RequestSecurityTokenResponseCollectionMsg"/>
        </wsdl:operation>
        <wsdl:operation name="Renew">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Renew" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/RenewFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
        <wsdl:operation name="Validate">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Validate" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/ValidateFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
        <wsdl:operation name="KeyExchangeToken">
            <wsdl:input wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/KET" message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output wsam:Action="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RSTR/KETFinal" message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
```

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

```
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="KeyExchangeToken">
            <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/KeyExchangeToken"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="RequestCollection">
            <soap:operation soapAction="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/RequestCollection"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsp:Policy wsu:Id="STS_policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <wsap10:UsingAddressing/>
                <wsp:ExactlyOne>
                    <sp:TransportBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
                        <wsp:Policy>
                            <sp:TransportToken>
                                <wsp:Policy>
                                    <sp:HttpsToken>
                                        <wsp:Policy/>
                                    </sp:HttpsToken>
                                </wsp:Policy>
                            </sp:TransportToken>
                            <sp:AlgorithmSuite>
                                <wsp:Policy>
                                    <sp:Basic128/>
                                </wsp:Policy>
                            </sp:AlgorithmSuite>
```

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

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

24.15.1. Example Requests and Responses for SAML Assertions

A client performs a RequestSecurityToken operation against the STS to receive a SAML assertion. The DDF STS offers several different ways to request a SAML assertion. For help in understanding the various request and response formats, samples have been provided. The samples are divided out into different request token types.

24.15.2. BinarySecurityToken (CAS) SAML Security Token Samples

Most endpoints in DDF require the X.509 PublicKey SAML assertion.

BinarySecurityToken (CAS) SAML Security Token Sample Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RST/Issue</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:60652909-faca-
4e4a-a4a7-8a5ce243a7cb</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>https://server:8993/services/SecurityTokenService</To>
        <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
            <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
        </ReplyTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-1">
                <wsu:Created>2013-04-29T18:35:10.688Z</wsu:Created>
                <wsu:Expires>2013-04-29T18:40:10.688Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512">
            <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
                    <wsa:Address>
https://server:8993/services/SecurityTokenService</wsa:Address>
                </wsa:EndpointReference>
            </wsp:AppliesTo>
            <wst:Claims xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512" Dialect=
"http://schemas.xmlsoap.org/ws/2005/05/identity">
```

```
                <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
Optional="true" Uri="
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
                <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
"/>
                <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
                <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
                <ic:ClaimType xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
Optional="true" Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
            </wst:Claims>
            <wst:OnBehalfOf>
                <BinarySecurityToken ValueType="#CAS" EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ns1:Id="
CAS" xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd" xmlns:ns1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
>U1QtMTQtYUtmcDYxcFRtS0FxZG1pVDMzOWMtY2FzfGh0dHBzOi8vdG9rZW5pc3N1ZXI6ODk5My9zZXJ2aWNlcy9T
ZWN1cml0eVRva2VuU2VydmljZQ==</BinarySecurityToken>
            </wst:OnBehalfOf>
            <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0</wst:TokenType>
            <wst:KeyType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/PublicKey</wst:KeyType>
            <wst:UseKey>
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data>
                        <ds:X509Certificate>
MIIC5DCCAk2gAwIBAgIJAKj7ROPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUzEQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZHllYXIxGDAWBgNVBAoMD0xvY2toZWVkIE1h
cnRpbjENMAsGA1UECwwESTRDRTEPMA0GA1UEAwwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGNl
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTAlVT
MRAwDgYDVQQIDAdBcml6b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2hlZWQg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbGllbnQxHDAaBgkqhkiG9w0BCQEWDWk0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVCnEmHndkf9bWUlAOfNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnhlNW9+lxBDAfBgNVHSMEGDAWgBQx11VLtYXL
vFGpFdHnhlNW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4GBAHYs2OI0K6yVXzyS
sKcv2fmfw6XCICGTnyA7BOdAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcDTR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/O5tywXRT1+freI3bwAN0
L6tQ
</ds:X509Certificate>
                    </ds:X509Data>
                </ds:KeyInfo>
            </wst:UseKey>
```

```
            <wst:Renewing/>
        </wst:RequestSecurityToken>
    </soap:Body>
</soap:Envelope>
```

BinarySecurityToken (CAS) SAML Security Token Sample Response

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

```
                                <ds:Reference URI="#_BDC44EB8593F47D1B213672605113671">
                                    <ds:Transforms>
                                        <ds:Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                        <ds:Transform Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#">
                                            <ec:InclusiveNamespaces xmlns:ec=
"http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs"/>
                                        </ds:Transform>
                                    </ds:Transforms>
                                    <ds:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
                                    <ds:DigestValue>
6wnWbft6Pz5XOF5Q9AG59gcGwLY=</ds:DigestValue>
                                </ds:Reference>
                            </ds:SignedInfo>

<ds:SignatureValue>h+NvkgXGdQtca3/eKebhAKgG38tHp3i2n5uLLy8xXXIg02qyKgEP0FCowp2LiYlsQU9YjK
fSwCUbH3WR6jhbAv9zj29CE+ePfEny7MeXvgNl3wId+vcHqti/DGGhhgtO2Mbx/tyX1BhHQUwKRlcHajxHeecwmvV
7D85NMdV48tI=</ds:SignatureValue>
                            <ds:KeyInfo>
                                <ds:X509Data>

<ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA
1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxMVoXDTIz
MDQwNzE4MzcxMVowgaYxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdBcml6b25hMREwDwYDVQQHEwhH
b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtxtN9
uB44diiIqq3JOzDGfDhGLu6mjpuHO1hrKItv42hBOhhmH7lS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lfi7IvsVVsNrtAEQljOMmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqpV8Kb0COOGwIDAQAB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2Vy
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgNVHSMEgZ8wgZyAFBcn
en6/j05DzaVwORwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk7OcwO7gwwDQYJKoZIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZlO6VrAxRQ+eFeN6NyuyhgYy5K6l/sIWdaGou5iJOQx
2pQYWx1v8Klyl0W22IfEAXYv/epiO89hpdACryuDJpioXI/X8TAwvRwLKL21Dk3k2b+eyCgA0O++
HM0dPfiQLQ99ElWkv/0=</ds:X509Certificate>
                                </ds:X509Data>
                            </ds:KeyInfo>
                        </ds:Signature>
                        <saml2:Subject>
                            <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="http://cxf.apache.org/sts">srogers</saml2:NameID>
                            <saml2:SubjectConfirmation Method=
```

```
"urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
                                <saml2:SubjectConfirmationData xsi:type=
"saml2:KeyInfoConfirmationDataType">
                                    <ds:KeyInfo xmlns:ds=
"http://www.w3.org/2000/09/xmldsig#">
                                        <ds:X509Data>

<ds:X509Certificate>MIIC5DCCAk2gAwIBAgIJAKj7ROPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGE
wJVUzEQ
MA4GA1UECAwHQXJpem9uYTERMA8GA1UEBwwIR29vZHllYXIxGDAWBgNVBAoMD0xvY2toZWVkIE1h
cnRpbjENMAsGA1UECwwESTRDRTEPMA0GA1UEAwwGY2xpZW50MRwwGgYJKoZIhvcNAQkBFg1pNGNl
QGxtY28uY29tMB4XDTEyMDYyMDE5NDMwOVoXDTIyMDYxODE5NDMwOVowgYoxCzAJBgNVBAYTAlVT
MRAwDgYDVQQIDAdBcml6b25hMREwDwYDVQQHDAhHb29keWVhcjEYMBYGA1UECgwPTG9ja2hlZWQg
TWFydGluMQ0wCwYDVQQLDARJNENFMQ8wDQYDVQQDDAZjbGllbnQxHDAaBgkqhkiG9w0BCQEWDWk0
Y2VAbG1jby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIpHxCBLYE7xfDLcITS9SsPG
4Q04Z6S32/+TriGsRgpGTj/7GuMG7oJ98m6Ws5cTYl7nyunyHTkZuP7rBzy4esDIHheyx18EgdSJ
vvACgGVCnEmHndkf9bWUlAOfNaxW+vZwljUkRUVdkhPbPdPwOcMdKg/SsLSNjZfsQIjoWd4rAgMB
AAGjUDBOMB0GA1UdDgQWBBQx11VLtYXLvFGpFdHnhlNW9+lxBDAfBgNVHSMEGDAWgBQx11VLtYXL
vFGpFdHnhlNW9+lxBDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4GBAHYs2OI0K6yVXzyS
sKcv2fmfw6XCICGTnyA7BOdAjYoqq6wD+33dHJUCFDqye7AWdcivuc7RWJt9jnlfJZKIm2BHcDTR
Hhk6CvjJ14Gf40WQdeMHoX8U8b0diq7Iy5Ravx+zRg7SdiyJUqFYjRh/O5tywXRT1+freI3bwAN0
L6tQ</ds:X509Certificate>
                                        </ds:X509Data>
                                    </ds:KeyInfo>
                                </saml2:SubjectConfirmationData>
                            </saml2:SubjectConfirmation>
                        </saml2:Subject>
                        <saml2:Conditions NotBefore="2013-04-29T18:35:11.407Z"
NotOnOrAfter="2013-04-29T19:05:11.407Z">
                            <saml2:AudienceRestriction>

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

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml2:Aut
hnContextClassRef>
                            </saml2:AuthnContext>
                        </saml2:AuthnStatement>
                        <saml2:AttributeStatement>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
                            </saml2:Attribute>
```

```
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string"
>srogers@example.com</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
srogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
                            </saml2:Attribute>
                        </saml2:AttributeStatement>
                    </saml2:Assertion>
                </RequestedSecurityToken>
                <RequestedAttachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedAttachedReference>
                <RequestedUnattachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
```

```
_BDC44EB8593F47D1B213672605113671</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedUnattachedReference>
                <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                    <wsa:EndpointReference xmlns:wsa=
"http://www.w3.org/2005/08/addressing">
                        <wsa:Address>
https://server:8993/services/SecurityTokenService</wsa:Address>
                    </wsa:EndpointReference>
                </wsp:AppliesTo>
                <Lifetime>
                    <ns2:Created>2013-04-29T18:35:11.444Z</ns2:Created>
                    <ns2:Expires>2013-04-29T19:05:11.444Z</ns2:Expires>
                </Lifetime>
            </RequestSecurityTokenResponse>
        </RequestSecurityTokenResponseCollection>
    </soap:Body>
</soap:Envelope>
```

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

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

24.15.3. UsernameToken Bearer SAML Security Token Sample

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

UsernameToken Bearer SAML Security Token Sample Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
```

```
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-1">
                <wsu:Created>2013-04-29T17:47:37.817Z</wsu:Created>
                <wsu:Expires>2013-04-29T17:57:37.817Z</wsu:Expires>
            </wsu:Timestamp>
            <wsse:UsernameToken wsu:Id="UsernameToken-1">
                <wsse:Username>srogers</wsse:Username>
                <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">password1</wsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
        <wsa:Action>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/RST/Issue</wsa:Action>
        <wsa:MessageID>uuid:a1bba87b-0f00-46cc-975f-001391658cbe</wsa:MessageID>
        <wsa:To>https://server:8993/services/SecurityTokenService</wsa:To>
    </soap:Header>
    <soap:Body>
        <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512">
            <wst:SecondaryParameters>
                <t:TokenType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</t:TokenType>
                <t:KeyType xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer</t:KeyType>
                <t:Claims xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512" Dialect=
"http://schemas.xmlsoap.org/ws/2005/05/identity">
                    <!--Add any additional claims you want to grab for the service-->
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/uid"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
                    <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
                </t:Claims>
            </wst:SecondaryParameters>
            <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
                    <wsa:Address>https://server:8993/services/QueryService</wsa:Address>
                </wsa:EndpointReference>
```

```
            </wsp:AppliesTo>
            <wst:Renewing/>
        </wst:RequestSecurityToken>
    </soap:Body>
</soap:Envelope>
```

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

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

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

The `AttributeStatement` block contains all the Claims requested.

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

UsernameToken Bearer SAML Security Token Sample Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <Action xmlns="http://www.w3.org/2005/08/addressing">http://docs.oasis-
open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</Action>
        <MessageID xmlns="http://www.w3.org/2005/08/addressing">urn:uuid:eee4c6ef-ac10-
4cbc-a53c-13d960e3b6e8</MessageID>
        <To xmlns="http://www.w3.org/2005/08/addressing"
>http://www.w3.org/2005/08/addressing/anonymous</To>
        <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:a1bba87b-0f00-46cc-
975f-001391658cbe</RelatesTo>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS-2">
                <wsu:Created>2013-04-29T17:49:12.624Z</wsu:Created>
                <wsu:Expires>2013-04-29T17:54:12.624Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/ws-
sx/ws-trust/200512" xmlns:ns2="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:ns3="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns4="http://www.w3.org/2005/08/addressing"
xmlns:ns5="http://docs.oasis-open.org/ws-sx/ws-trust/200802">
            <RequestSecurityTokenResponse>
                <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
```

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

<ds:SignatureValue>X5Kzd54PrKIlGVV2XxzCmWFRzHRoybF7hU6zxbEhSLMR0AWS9R7Me3epq91XqeOwvIDDbw
mE/oJNC7vI0fIw/rqXkx4aZsY5a5nbAs7f+aXF9TGdk82x2eNhNGYpViq0YZJfsJ5WSyMtG8w5nRekmDMy9oTLsHG
+Y/OhJDEwq58=</ds:SignatureValue>
                            <ds:KeyInfo>
                                <ds:X509Data>

<ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA
1UECBMH
QXJpem9uYTERMA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4
YW1wbGUxEDAOBgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBMB4XDTEzMDQwOTE4MzcxMVoXDTIz
MDQwNzE4MzcxMVowgaYxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdBcml6b25hMREwDwYDVQQHEwhH
b29keWVhcjEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UEChMHRXhhbXBsZTEQMA4GA1UECxMHRXhh
bXBsZTEUMBIGA1UEAxMLdG9rZW5pc3N1ZXIxJjAkBgkqhkiG9w0BCQEWF3Rva2VuaXNzdWVyQGV4
YW1wbGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDfktpA8Lrp9rTfRibKdgtxtN9
uB44diiIqq3JOzDGfDhGLu6mjpuHO1hrKItv42hBOhhmH7lS9ipiaQCIpVfgIG63MB7fa5dBrfGF
G69vFrU1Lfi7IvsVVsNrtAEQljOMmw9sxS3SUsRQX+bD8jq7Uj1hpoF7DdqpV8Kb0COOGwIDAQAB
o4IBBjCCAQIwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2Vy
dGlmaWNhdGUwHQYDVR0OBBYEFD1mHviop2Tc4HaNu8yPXR6GqWP1MIGnBgNVHSMEgZ8wgZyAFBcn
```

```
en6/j05DzaVwORwrteKc7TZOoXmkdzB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTER
MA8GA1UEBxMIR29vZHllYXIxEDAOBgNVBAoTB0V4YW1wbGUxEDAOBgNVBAoTB0V4YW1wbGUxEDAO
BgNVBAsTB0V4YW1wbGUxCzAJBgNVBAMTAkNBggkAwXk7OcwO7gwwDQYJKoZIhvcNAQEEBQADgYEA
PiTX5kYXwdhmijutSkrObKpRbQkvkkzcyZlO6VrAxRQ+eFeN6NyuyhgYy5K6l/sIWdaGou5iJOQx
2pQYWx1v8Klyl0W22IfEAXYv/epiO89hpdACryuDJpioXI/X8TAwvRwLKL21Dk3k2b+eyCgA0O++
HM0dPfiQLQ99ElWkv/0=</ds:X509Certificate>
                                </ds:X509Data>
                            </ds:KeyInfo>
                        </ds:Signature>
                        <saml2:Subject>
                            <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="http://cxf.apache.org/sts">srogers</saml2:NameID>
                            <saml2:SubjectConfirmation Method=
"urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
                        </saml2:Subject>
                        <saml2:Conditions NotBefore="2013-04-29T17:49:12.614Z"
NotOnOrAfter="2013-04-29T18:19:12.614Z">
                            <saml2:AudienceRestriction>
                                <saml2:Audience>
https://server:8993/services/QueryService</saml2:Audience>
                            </saml2:AudienceRestriction>
                        </saml2:Conditions>
                        <saml2:AuthnStatement AuthnInstant="2013-04-29T17:49:12.613Z">
                            <saml2:AuthnContext>

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

```
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">Steve
Rogers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
                            </saml2:Attribute>
                            <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                                <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
                            </saml2:Attribute>
                        </saml2:AttributeStatement>
                    </saml2:Assertion>
                </RequestedSecurityToken>
                <RequestedAttachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_7437C1A55F19AFF22113672577526132</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedAttachedReference>
                <RequestedUnattachedReference>
                    <ns3:SecurityTokenReference xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
                        <ns3:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">
_7437C1A55F19AFF22113672577526132</ns3:KeyIdentifier>
                    </ns3:SecurityTokenReference>
                </RequestedUnattachedReference>
                <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
                    <wsa:EndpointReference xmlns:wsa=
"http://www.w3.org/2005/08/addressing">
                        <wsa:Address>
https://server:8993/services/QueryService</wsa:Address>
                    </wsa:EndpointReference>
                </wsp:AppliesTo>
                <Lifetime>
```

```
                    <ns2:Created>2013-04-29T17:49:12.620Z</ns2:Created>
                    <ns2:Expires>2013-04-29T18:19:12.620Z</ns2:Expires>
                </Lifetime>
            </RequestSecurityTokenResponse>
        </RequestSecurityTokenResponseCollection>
    </soap:Body>
</soap:Envelope>
```

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

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

24.15.4. X.509 PublicKey SAML Security Token Sample

X.509 PublicKey SAML Security Token Request

The STS that comes with DDF requires the following to be in the RequestSecurityToken request in order to issue a valid SAML assertion. See the request block below for an example of how these components should be populated.

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

X.509 PublicKey SAML Security Token Sample Request

```
<soapenv:Envelope xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
```

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

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

         <!-- OPTIONAL: Only required if the endpoint that the SAML assertion will be
sent to requires claims. -->
         <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity" xmlns:ic=
"http://schemas.xmlsoap.org/ws/2005/05/identity">
            <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"/>
            <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"/>
            <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
            <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"/>
            <ic:ClaimType Optional="true" Uri=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
         </wst:Claims>
         <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue</wst:RequestType>
```

```
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
               <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
               <wsa:Address>https://server:8993/services/QueryService</wsa:Address>
               </wsa:EndpointReference>
            </wsp:AppliesTo>
            <wst:UseKey>
               <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                  <ds:X509Data>

<ds:X509Certificate>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWB
gNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjU0OVoXDTE2MDUwNzAwMjU0OVowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxDzANBgNVBAMTBmNsaWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAOq6L1/jjZ5cyhjhHEbOHr5WQpboKACYbrsn8lg85LGNoAfcwImr9KBmOxGb
ZCxHYIhkW7pJ+kppyH8DbbbDMviIvvdkvrAIU0l8OBRn2wReCBGQ01Imdc3+WzFF2svW75d6wii2ZVd
eMvUO15p/pAD/sdIfXmAfyu8+tqtiO8KVZGkTnlg3AMzfeSrkci5UHMVWj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpdZN2pC85tj9xIoWGh5dFWG1fPcYRgzGPxsybiGOylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAhiWRGyPlY08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNXy43NZLBBlnDjDplNZJoMB0GA1UdDgQWBBRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2NybC5nZHMu
bml0LmRpc2EubWlsL2NybC9ET0RKSVRDQ0FfMjcuY3JsMIGUoIGRoIGOhoGLbGRhcDovL2NybC5n
ZHMubml0LmRpc2EubWlsL2NuJTNkRE9EJTIwSklUQyUyMENBLTI3JTJjb3UlM2RQS0klMmNvdSUz
ZERvRCUyY28lM2RVLlMuJTIwR292ZXJubWVudCUyY2MlM2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN0O2JpbmFyeTAjBgNVHSAEHDAaMAsGCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzAChjFodHRwOi8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVRDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzcC5uc24wLnJjdnMubml0LmRp
c2EubWlsMA0GCSqGSIb3DQEBBQUAA4IBAQCGUJPGh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSgzxnT4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIHr
hwtE3xo6qQRRWlO3/clC4RnTev1crFVJQVBF3yfpRu8udJ2SOGdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCxg1LjC4dmbjKVr9s4QK+/JLNjxD8IkJiZE</ds:X509Certific
ate>
                  </ds:X509Data>
               </ds:KeyInfo>
            </wst:UseKey>
         </wst:RequestSecurityToken>
      </soapenv:Body>
   </soapenv:Envelope>
```

X.509 PublicKey SAML Security Token Response

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

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

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

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

X.509 PublicKey SAML Security Token Sample Response

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

```
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                        <ds:Reference URI="#_90DBA0754E55B4FE7013928310431176">
                           <ds:Transforms>
                              <ds:Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
                                 <ec:InclusiveNamespaces PrefixList="xs" xmlns:ec=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
                              </ds:Transform>
                           </ds:Transforms>
                           <ds:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
                           <ds:DigestValue>/bEGqsRGHVJbx298WPmGd8I53zs=</ds:DigestValue>
                        </ds:Reference>
                     </ds:SignedInfo>
                     <ds:SignatureValue>
mYR7w1/dnuh8Z7t9xjCb4XkYQLshj+UuYlGOuTwDYsUPcS2qI0nAgMD1VsDP7y1fDJxeqsq7HYhFKsnqRfebMM4WL
H1D/lJ4rD4UO+i9l3tuiHml7SN24WM1/bOqfDUCoDqmwG8afUJ3r4vmTNPxfwfOss8BZ/8ODgZzm08ndlkxDfvcN7
OrExbV/3/45JwF/MMPZoqvi2MJGfX56E9fErJNuzezpWnRqPOlWPxyffKMAlVaB9zF6gvVnUqcW2k/Z8X9lN7O5jo
uBI281ZnIfsIPuBJERFtYNVDHsIXM1pJnrY6FlKIaOsi55LQu3Ruir/n82pU7BT5aWtxwrn7akBg==
</ds:SignatureValue>
                     <ds:KeyInfo>
                        <ds:X509Data>

<ds:X509Certificate>MIIFHTCCBAWgAwIBAgICJe8wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWB
gNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjYzN1oXDTE2MDUwNzAwMjYzN1owbjELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxFDASBgNVBAMTC3Rva2VuaXNzdWVyMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAx01/U4M1wG+wL1JxX2RL1glj101FkJXMk3KFt3zD//N8x/Dcwwvs
ngCQjXrV6YhbB2V7scHwnThPv3RSwYYiO62z+g6ptfBbKGGBLSZOzLe3fyJR4RxblFKsELFgPHfX
vgUHS/keG5uSRk9S/Okqps/yxKB7+ZlxeFxsIz5QywXvBpMiXtc2zF+M7BsbSIdSx5LcPcDFBwjF
c66rE3/y/25VMht9EZX1QoKr7f8rWD4xgd5J6DYMFWEcmiCz4BDJH9sfTw+n1P+CYgrhwslWGqxt
cDME9t6SWR3GLT4Sdtr8ziIM5uUteEhPIV3rVC3/u23JbYEeS8mpnp0bxt5eHQIDAQABo4IB1TCC
AdEwHwYDVR0jBBgwFoAUIxQ0IE1fLjc1ksEGWcOMOmU1kmgwHQYDVR0OBBYEFGBjdkdey+bMHMhC
Z7gwiQ/mJf5VMA4GA1UdDwEB/wQEAwIFoDCB2gYDVR0fBIHSMIHPMDagNKAyhjBodHRwOi8vY3Js
Lmdkcy5uaXQuZGlzYS5taWwvY3JsL0RPREpJVENDQV8yNy5jcmwwgZSggZGggY6GgYtsZGFwOi8v
Y3JsLmdkcy5uaXQuZGlzYS5taWwvY24lM2RET0QlMjBKSVRDJTIwQ0EtMjclMmNvdSUzZFBLSSUy
Y291JTNkRG9EJTJjbyUzZFUuUy4lMjBHb3Zlcm5tZW50JTJjYyUzZFVTP2NlcnRpZmljYXRlcmV2
b2NhdGlvbmxpc3Q7YmluYXJ5MCMGA1UdIAQcMBowCwYJYIZIAWUCAQsFMAsGCWCGSAFlAgELEjB9
BggrBgEFBQcBAQRxMG8wPQYIKwYBBQUHMAKGMWh0dHA6Ly9jcmwuZ2RzLm5pdC5kaXNhLm1pbC9z
aWduL0RPREpJVENDQV8yNy5jZXIwLgYIKwYBBQUHMAGGImh0dHA6Ly9vY3NwLm5zbjAucmN2cy5u
aXQuZGlzYS5taWwwDQYJKoZIhvcNAQEFBQADggEBAIHZQTINU3bMpJ/PkwTYLWPmwCqAYgEUzSYx
bNcVY5MWD8b4XCdw5nM3GnFlOqr4IrHeyyOzsEbIebTe3bv0l1pHx0Uyj059nAhx/AP8DjVtuRU1
/Mp4b6uJ/4yaoMjIGceqBzHqhHIJinG0Y2azua7eM9hVbWZsa912ihbiupCq22mYuHFP7NUNzBvV
j03YUcsy/sES5sRx9Rops/CBN+LUUYOdJOxYWxo8oAbtF8ABE5ATLAwqz4ttsToKPUYh1sxdx5Ef
```

```
APeZ+wYDmMu4OfLckwnCKZgkEtJOxXpdIJHY+VmyZtQSB0LkR5toeH/ANV4259Ia5ZT8h2/vIJBg
6B4=</ds:X509Certificate>
                        </ds:X509Data>
                     </ds:KeyInfo>
                  </ds:Signature>
                  <saml2:Subject>
                     <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified" NameQualifier="http://cxf.apache.org/sts">pparker</saml2:NameID>
                     <saml2:SubjectConfirmation Method=
"urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
                        <saml2:SubjectConfirmationData xsi:type=
"saml2:KeyInfoConfirmationDataType">
                           <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                              <ds:X509Data>

<ds:X509Certificate>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWB
gNVBAoT
D1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kxFzAVBgNVBAMTDkRP
RCBKSVRDIENBLTI3MB4XDTEzMDUwNzAwMjU0OVoXDTE2MDUwNzAwMjU0OVowaTELMAkGA1UEBhMC
VVMxGDAWBgNVBAoTD1UuUy4gR292ZXJubWVudDEMMAoGA1UECxMDRG9EMQwwCgYDVQQLEwNQS0kx
EzARBgNVBAsTCkNPTlRSQUNUT1IxDzANBgNVBAMTBmNsaWVudDCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAOq6L1/jjZ5cyhjhHEbOHr5WQpboKACYbrsn8lg85LGNoAfcwImr9KBmOxGb
ZCxHYIhkW7pJ+kppyH8bbbviIvvdkvrAIU0l8OBRn2wReCBGQ01Imdc3+WzFF2svW75d6wii2ZVd
eMvUO15p/pAD/sdIfXmAfyu8+tqtiO8KVZGkTnlg3AMzfeSrkci5UHMVWj0qUSuzLk9SAg/9STgb
Kf2xBpHUYecWFSB+dTpdZN2pC85tj9xIoWGh5dFWG1fPcYRgzGPxsybiGOylbJ7rHDJuL7IIIyx5
EnkCuxmQwoQ6XQAhiWRGyPlY08w1LZixI2v+Cv/ZjUfIHv49I9P4Mt8CAwEAAaOCAdUwggHRMB8G
A1UdIwQYMBaAFCMUNCBNXy43NZLBBlnDjDplNZJoMB0GA1UdDgQWBBRPGiX6zZzKTqQSx/tjg6hx
9opDoTAOBgNVHQ8BAf8EBAMCBaAwgdoGA1UdHwSB0jCBzzA2oDSgMoYwaHR0cDovL2NybC5nZHMu
bml0LmRpc2EubWlsL2NybC9ET0RKSVRDQ0FfMjcuY3JsMIGUoIGRoIGOhoGLbGRhcDovL2NybC5n
ZHMubml0LmRpc2EubWlsL2NuJTNkRE9EJTIwSklUQyUyMENBLTI3JTJjb3UlM2RQS0klMmNvdSUz
ZERvRCUyY28lM2RVLlMuJTIwR292ZXJubWVudCUyY2MlM2RVUz9jZXJ0aWZpY2F0ZXJldm9jYXRp
b25saXN0O2JpbmFyeTAjBgNVHSAEHDAaMAsGCWCGSAFlAgELBTALBglghkgBZQIBCxIwfQYIKwYB
BQUHAQEEcTBvMD0GCCsGAQUFBzAChjFodHRwOi8vY3JsLmdkcy5uaXQuZGlzYS5taWwvc2lnbi9E
T0RKSVRDQ0FfMjcuY2VyMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzcC5uc24wLnJjdnMubml0LmRp
c2EubWlsMA0GCSqGSIb3DQEBBQUAA4IBAQCGUJPGh4iGCbr2xCMqCq04SFQ+iaLmTIFAxZPFvup1
4E9Ir6CSDalpF9eBx9fS+Z2xuesKyM/g3YqWU1LtfWGRRIxzEujaC4YpwHuffkx9QqkwSkXXIsim
EhmzSgzxnT4Q9X8WwalqVYOfNZ6sSLZ8qPPFrLHkkw/zIFRzo62wXLu0tfcpOr+iaJBhyDRinIHr
hwtE3xo6qQRRWlO3/clC4RnTev1crFVJQVBF3yfpRu8udJ2SOGdqU0vjUSu1h7aMkYJMHIu08Whj
8KASjJBFeHPirMV1oddJ5ydZCQ+Jmnpbwq+XsCxg1LjC4dmbjKVr9s4QK+/JLNjxD8IkJiZE</ds:X509Certific
ate>
                              </ds:X509Data>
                           </ds:KeyInfo>
                        </saml2:SubjectConfirmationData>
                     </saml2:SubjectConfirmation>
                  </saml2:Subject>
                  <saml2:Conditions NotBefore="2014-02-19T17:30:43.119Z" NotOnOrAfter=
"2014-02-19T18:00:43.119Z"/>
                  <saml2:AuthnStatement AuthnInstant="2014-02-19T17:30:43.117Z">
```

```
                    <saml2:AuthnContext>

<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml2:Auth
nContextClassRef>
                    </saml2:AuthnContext>
                 </saml2:AuthnStatement>

                 <!-- This block will only be included if Claims were requested in the
RST. -->
                 <saml2:AttributeStatement>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                       <saml2:AttributeValue xsi:type="xs:string">
pparker</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                       <saml2:AttributeValue xsi:type="xs:string">
pparker@example.com</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                       <saml2:AttributeValue xsi:type="xs:string">
pparker</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                       <saml2:AttributeValue xsi:type="xs:string">Peter
Parker</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                       <saml2:AttributeValue xsi:type="xs:string">
users</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                       <saml2:AttributeValue xsi:type="xs:string">
users</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
```

```
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
avengers</saml2:AttributeValue>
                    </saml2:Attribute>
                    <saml2:Attribute Name=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" NameFormat=
"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
                        <saml2:AttributeValue xsi:type="xs:string">
admin</saml2:AttributeValue>
                    </saml2:Attribute>
                </saml2:AttributeStatement>
            </saml2:Assertion>
        </ns2:RequestedSecurityToken>
        <ns2:RequestedAttachedReference>
            <ns4:SecurityTokenReference wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0" xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
                <ns4:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-
saml-token-profile-1.1#SAMLID">_90DBA0754E55B4FE7013928310431176</ns4:KeyIdentifier>
            </ns4:SecurityTokenReference>
        </ns2:RequestedAttachedReference>
        <ns2:RequestedUnattachedReference>
            <ns4:SecurityTokenReference wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0" xmlns:wsse11="http://docs.oasis-
open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
                <ns4:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-
saml-token-profile-1.1#SAMLID">_90DBA0754E55B4FE7013928310431176</ns4:KeyIdentifier>
            </ns4:SecurityTokenReference>
        </ns2:RequestedUnattachedReference>
        <ns2:Lifetime>
            <ns3:Created>2014-02-19T17:30:43.119Z</ns3:Created>
            <ns3:Expires>2014-02-19T18:00:43.119Z</ns3:Expires>
        </ns2:Lifetime>
    </ns2:RequestSecurityTokenResponse>
</ns2:RequestSecurityTokenResponseCollection>
</soap:Body>
</soap:Envelope>
```

24.16. Developing Registry Clients

Registry Clients create Federated Sources using the OSGi Configuration Admin. Developers should reference an individual `Source`'s (Federated, Connected, or Catalog Provider) documentation for the Configuration properties (such as a Factory PID, addresses, intervals, etc) necessary to establish that `Source` in the framework.

Creating a Source Configuration

```
org.osgi.service.cm.ConfigurationAdmin configurationAdmin = getConfigurationAdmin() ;
org.osgi.service.cm.Configuration currentConfiguration = configurationAdmin
.createFactoryConfiguration(getFactoryPid(), null);
Dictionary properties = new Dictionary() ;
properties.put(QUERY_ADDRESS_PROPERTY,queryAddress);
currentConfiguration.update( properties );
```

Note that the `QUERY_ADDRESS_PROPERTY` is specific to this Configuration and might not be required for every `Source`. The properties necessary for creating a Configuration are different for every `Source`.

24.17. Developing Resource Readers

A Resource Reader is a class that retrieves a resource from a native/external source and returns it to DDF. A simple example is that of a File `ResourceReader`. It takes a file from the local file system and passes it back to DDF. New implementations can be created in order to support obtaining Resources from various Resource data stores.

24.17.1. Creating a New `ResourceReader`

Complete the following procedure to create a `ResourceReader`.

1. Create a Java class that implements the `DDF.catalog.resource.ResourceReader` interface.
2. Deploy the OSGi bundled packaged service to the DDF run-time.

24.17.1.1. Implementing the `ResourceReader` Interface

```
public class TestResourceReader implements DDF.catalog.resource.ResourceReader
```

`ResourceReader` has a couple of key methods where most of the work is performed.

NOTE	**URI** It is recommended to become familiar with the Java API URI class in order to properly build a `ResourceReader`. Furthermore, a URI should be used according to its specification.

24.17.1.2. retrieveResource

```
public ResourceResponse retrieveResource( URI uri, Map<String, Serializable> arguments
)throws IOException, ResourceNotFoundException, ResourceNotSupportedException;
```

This method is the main entry to the ResourceReader. It is used to retrieve a Resource and send it back to the caller (generally the CatalogFramework). Information needed to obtain the entry is contained in the URI reference. The URI Scheme needs to match a scheme specified in the getSupportedSchemes method. This is how the CatalogFramework determines which ResourceReader implementation to use. If there are multiple ResourceReaders supporting the same scheme, these ResourceReaders are invoked iteratively. Invocation of the ResourceReaders stops once one of them returns a Resource.

Arguments are also passed in. These can be used by the ResourceReader to perform additional operations on the resource.

The URLResourceReader is an example ResourceReader that reads a file from a URI.

NOTE	The Map<String, Serializable> arguments parameter is passed in to support any options or additional information associated with retrieving the resource.

24.17.1.3. Implement retrieveResource()

1. Define supported schemes (such as file, http, etc.).
2. Check if the incoming URI matches a supported scheme. If it does not, throw ResourceNotSupportedException.

Example:

```
if ( !uri.getScheme().equals("http") )
 {
   throw new ResourceNotSupportedException("Unsupported scheme received, was expecting
http")
 }
```

1. Implement the business logic.
2. For example, the URLResourceReader obtains the resource through a connection:

```
URL url = uri.toURL();
URLConnection conn = url.openConnection();
String mimeType = conn.getContentType();
if ( mimeType == null ) {
    mimeType = URLConnection.guessContentTypeFromName( url.getFile() );
}
InputStream is = conn.getInputStream();
```

NOTE	The Resource needs to be accessible from the DDF installation (see the rootResourceDirectories property of the URLResourceReader). This includes being able to find a file locally or reach out to a remote URI. This may require Internet access, and DDF may need to be configured to use a proxy (http.proxyHost and http.proxyPort can

be added to the system properties on the command line script).

1. Return `Resource` in `ResourceResponse`.

For example:

```
return ResourceResponseImpl( new ResourceImpl( new BufferedInputStream( is ), new
MimeType( mimeType ), url.getFile() ) );
```

If the Resource cannot be found, throw a `ResourceNotFoundException`.

24.17.1.4. `getSupportedSchemes`

```
public Set<String> getSupportedSchemes();
```

This method lets the `ResourceReader` inform the CatalogFramework about the type of URI scheme that it accepts and should be passed. For single-use ResourceReaders (like a URLResourceReader), there may be only one scheme that it can accept while others may understand more than one. A ResourceReader must, at minimum, accept one qualifier. As mentioned before, this method is used by the `CatalogFramework` to determine which `ResourceReader` to invoke.

NOTE

`ResourceReader` extends `Describable`
Additionally, there are other methods that are used to uniquely describe a `ResourceReader`. The `describe` methods are straight-forward and can be implemented with guidance from the Javadoc.

24.17.1.5. Export to OSGi Service Registry

In order for the `ResourceReader` to be used by the `CatalogFramework`, it should be exported to the OSGi Service Registry as a `DDF.catalog.resource.ResourceReader`.

See the XML below for an example:

Blueprint example

```
<bean id="customResourceReaderId]" class=
"example.resource.reader.impl.CustomResourceReader" />
<service ref="customResourceReaderId" interface="DDF.catalog.source.ResourceReader" />
```

24.18. Developing Resource Writers

A Resource Writer is an object used to store or delete a `Resource`. `ResourceWriter` objects should be registered within the OSGi Service Registry, so clients can retrieve an instance when they need to store

a `Resource`.

24.18.1. Create a New `ResourceWriter`

Complete the following procedure to create a `ResourceWriter`.

1. Create a Java class that implements the `DDF.catalog.resource.ResourceWriter` interface.

ResourceWriter Implementation Skeleton

```
import java.io.IOException;
import java.net.URI;
import java.util.Map;
import DDF.catalog.resource.Resource;
import DDF.catalog.resource.ResourceNotFoundException;
import DDF.catalog.resource.ResourceNotSupportedException;
import DDF.catalog.resource.ResourceWriter;

public class SampleResourceWriter implements ResourceWriter {

    @Override
    public void deleteResource(URI uri, Map<String, Object> arguments) throws
ResourceNotFoundException, IOException {
       // WRITE IMPLEMENTATION
     }

    @Override
    public URI storeResource(Resource resource, Map<String, Object> arguments)throws
ResourceNotSupportedException, IOException {
       // WRITE IMPLEMENTATION
       return null;
    }

    @Override
    public URI storeResource(Resource resource, String id, Map<String, Object> arguments)
throws ResourceNotSupportedException, IOException {
       // WRITE IMPLEMENTATION
       return null;
    }

}
```

1. Register the implementation as a Service in the OSGi Service Registry.

Blueprint Service Registration Example

```
...
```

```
<service ref="ResourceWriterReference" interface="DDF.catalog.resource.ResourceWriter" />
...
```

1. Deploy the OSGi bundled packaged service to the DDF run-time (Refer to the OSGi Basics - Bundles section.)

TIP	**ResourceWriter Javadoc** Refer to the Catalog API Javadoc for more information about the methods required for implementing the interface.

24.19. Developing Filters

The common way to create a Filter is to use the GeoTools `FilterFactoryImpl` object, which provides Java implementations for the various types of filters in the Filter Specification. Examples are the easiest way to understand how to properly create a `Filter` and a `Query`.

NOTE	Refer to the GeoTools javadoc for more information on `FilterFactoryImpl`.

WARNING	Implementing the Filter interface directly is only for extremely advanced use cases and is highly discouraged. Instead, use of the DDF-specific `FilterBuilder` API is recommended.

Developers create a `Filter` object in order to filter or constrain the amount of records returned from a `Source`. The OGC Filter Specification has several types of filters that can be combined in a tree-like structure to describe the set of metacards that should be returned.

Categories of Filters

- Comparison Operators
- Logical Operators
- Expressions
- Literals
- Functions
- Spatial Operators
- Temporal Operators

24.19.1. Units of Measure

According to the OGC Filter Specifications: 09-026r1 and OGC Filter Specifications: 04-095, units of measure can be expressed as a URI. To fulfill that requirement, DDF utilizes the GeoTools class `org.geotools.styling.UomOgcMapping` for spatial filters requiring a standard for units of measure for scalar distances. Essentially, the `UomOgcMapping` maps the OGC Symbology Encoding standard URIs to

Java Units. This class provides three options for units of measure:

- FOOT
- METRE
- PIXEL

DDF only supports FOOT and METRE since they are the most applicable to scalar distances.

24.19.2. Filter Examples

The example below illustrates creating a query, and thus an OGC Filter, that does a case-insensitive search for the phrase "mission" in the entire metacard's text. Note that the OGC `PropertyIsLike` Filter is used for this simple contextual query.

Simple Contextual Search

```
org.geotools.api.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;
boolean isCaseSensitive = false ;

String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard, singleChar,
and the escapeChar itself

String searchPhrase = "mission" ;
org.geotools.api.filter.Filter propertyIsLikeFilter =
    filterFactory.like(filterFactory.property(Metacard.ANY_TEXT), searchPhrase,
wildcardChar, singleChar, escapeChar, isCaseSensitive);
DDF.catalog.operation.QueryImpl query = new QueryImpl( propertyIsLikeFilter );
```

The example below illustrates creating an absolute temporal query, meaning the query is searching for Metacards whose modified timestamp occurred during a specific time range. Note that this query uses the `During` OGC Filter for an absolute temporal query.

Absolute Temporal Search

```
org.geotools.api.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;
org.geotools.api.temporal.Instant startInstant = new org.geotools.temporal.object
.DefaultInstant(new DefaultPosition(start));

org.geotools.api.temporal.Instant endInstant = new org.geotools.temporal.object
.DefaultInstant(new DefaultPosition(end));

org.geotools.api.temporal.Period period =  new org.geotools.temporal.object.
DefaultPeriod(startInstant, endInstant);
```

```
String property = Metacard.MODIFIED ; // modified date of a metacard

org.geotools.api.filter.Filter filter = filterFactory.during( filterFactory.property
(property), filterFactory.literal(period)  );

DDF.catalog.operation.QueryImpl query = new QueryImpl(filter) ;
```

24.19.2.1. Contextual Searches

Most contextual searches can be expressed using the `PropertyIsLike` filter. The special characters that have meaning in a `PropertyIsLike` filter are the wildcard, single wildcard, and escape characters (see Example Creating-Filters-1).

Table 79. `PropertyIsLike` Special Characters

Character	Description
Wildcard	Matches zero or more characters.
Single Wildcard	Matches exactly one character.
Escape	Escapes the meaning of the Wildcard, Single Wildcard, and the Escape character itself

Characters and words, such as `AND`, `&`, `and`, `OR`, `|`, `or`, `NOT`, `~`, `not`, `{`, and `}`, are treated as literals in a `PropertyIsLike` filter. In order to create equivalent logical queries, a developer must instead use the Logical Operator filters {`AND`, `OR`, `NOT`}. The Logical Operator filters can be combined together with `PropertyIsLike` filters to create a tree that represents the search phrase expression.

Creating the search phrase "mission and planning"

```
org.geotools.api.filter.FilterFactory filterFactory = new FilterFactoryImpl() ;

boolean isCaseSensitive = false ;

String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard, singleChar, and
the escapeChar itself

Filter filter =
    filterFactory.and(
       filterFactory.like(filterFactory.property(Metacard.METADATA), "mission" ,
wildcardChar, singleChar, escapeChar, isCaseSensitive),
       filterFactory.like(filterFactory.property(Metacard.METADATA), "planning" ,
wildcardChar, singleChar, escapeChar, isCaseSensitive)
    );
```

```
DDF.catalog.operation.QueryImpl query = new QueryImpl( filter );
```

24.19.2.1.1. Tree View of Creating Filters

Filters used in DDF can always be represented in a tree diagram.

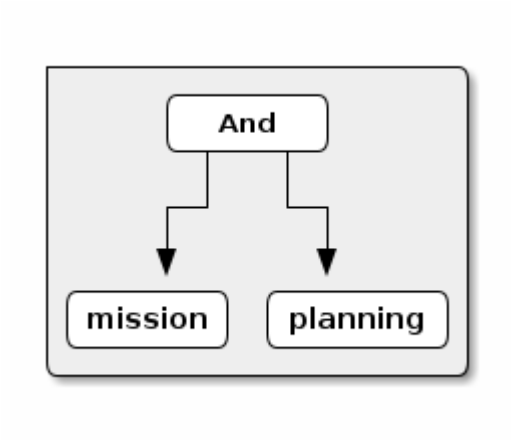


Figure 34. Filter Example Tree Diagram

24.19.2.1.2. XML View of Creating Filters

Another way to view this type of Filter is through an XML model, which is shown below.

Pseudo XML of Example Creating-Filters-3

```
<Filter>
   <And>
      <PropertyIsLike wildCard="*" singleChar="?" escapeChar="\">
           <PropertyName>metadata</PropertyName>
           <Literal>mission</Literal>
      </PropertyIsLike>
      <PropertyIsLike wildCard="*" singleChar="?" escapeChar="\">
           <PropertyName>metadata</PropertyName>
           <Literal>planning</Literal>
      </PropertyIsLike>
   <And>
</Filter>
```

Using the Logical Operators and `PropertyIsLike` filters, a developer can create a whole language of search phrase expressions.

24.19.2.2. Fuzzy Operations

DDF only supports one custom function. The Filter specification does not include a fuzzy operator, so a Filter function was created to represent a fuzzy operation. The function and class is called `FuzzyFunction`, which is used by clients to notify the Sources to perform a fuzzy search. The syntax

expected by providers is similar to the Fuzzy Function. Refer to the example below.

```
String wildcardChar = "*" ; // used to match zero or more characters
String singleChar = "?" ; // used to match exactly one character
String escapeChar = "\\" ; // used to escape the meaning of the wildCard, singleChar

boolean isCaseSensitive = false ;

Filter fuzzyFilter = filterFactory.like(
     new DDF.catalog.impl.filter.FuzzyFunction(
          Arrays.asList((Expression) (filterFactory.property(Metacard.ANY_TEXT))),
          filterFactory.literal("")),
     searchPhrase,
     wildcardChar,
     singleChar,
     escapeChar,
     isCaseSensitive);

QueryImpl query = new QueryImpl(fuzzyFilter);
```

24.19.3. Parsing Filters

According to the OGC Filter Specification 04-095: a "(filter expression) representation can be ... parsed and then transformed into whatever target language is required to retrieve or modify object instances stored in some persistent object store." Filters can be thought of as the `WHERE` clause for a SQL SELECT statement to "fetch data stored in a SQL-based relational database."

Sources can parse OGC Filters using the `FilterAdapter` and `FilterDelegate`. See Developing a Filter Delegate for more details on implementing a new `FilterDelegate`. This is the preferred way to handle OGC Filters in a consistent manner.

Alternately, `org.geotools.api.filter.Filter` implementations can be parsed using implementations of the interface `org.geotools.api.filter.FilterVisitor`. The `FilterVisitor` uses the Visitor pattern. Essentially, `FilterVisitor` instances "visit" each part of the `Filter` tree allowing developers to implement logic to handle the filter's operations. GeoTools 8 includes implementations of the `FilterVisitor` interface. The `DefaultFilterVisitor`, as an example, provides only business logic to visit every node in the `Filter` tree. The `DefaultFilterVisitor` methods are meant to be overwritten with the correct business logic. The simplest approach when using `FilterVisitor` instances is to build the appropriate query syntax for a target language as each part of the `Filter` is visited. For instance, when given an incoming `Filter` object to be evaluated against a RDBMS, a `CatalogProvider` instance could use a `FilterVisitor` to interpret each filter operation on the `Filter` object and translate those operations into SQL. The `FilterVisitor` may be needed to support `Filter` functionality not currently handled by the `FilterAdapter` and `FilterDelegate` reference implementation.

24.19.3.1. Interpreting a Filter to Create SQL

If the `FilterAdapter` encountered or "visited" a `PropertyIsLike` filter with its property assigned as `title` and its literal expression assigned as `mission`, the `FilterDelegate` could create the proper SQL syntax similar to title `LIKE` mission.

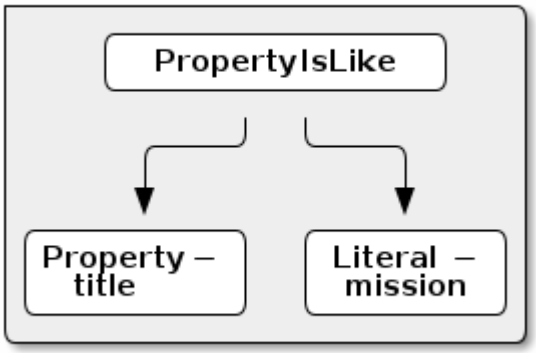


Figure 35. Parsing Filters Tree Diagram

24.19.3.2. Interpreting a Filter to Create XQuery

If the `FilterAdapter` encountered an `OR` filter, such as in Figure Parsing-Filters2 and the target language was XQuery, the `FilterDelegate` could yield an expression such as

```
ft:query(//inventory:book/@subject,'math') union
ft:query(//inventory:book/@subject,'science').
```

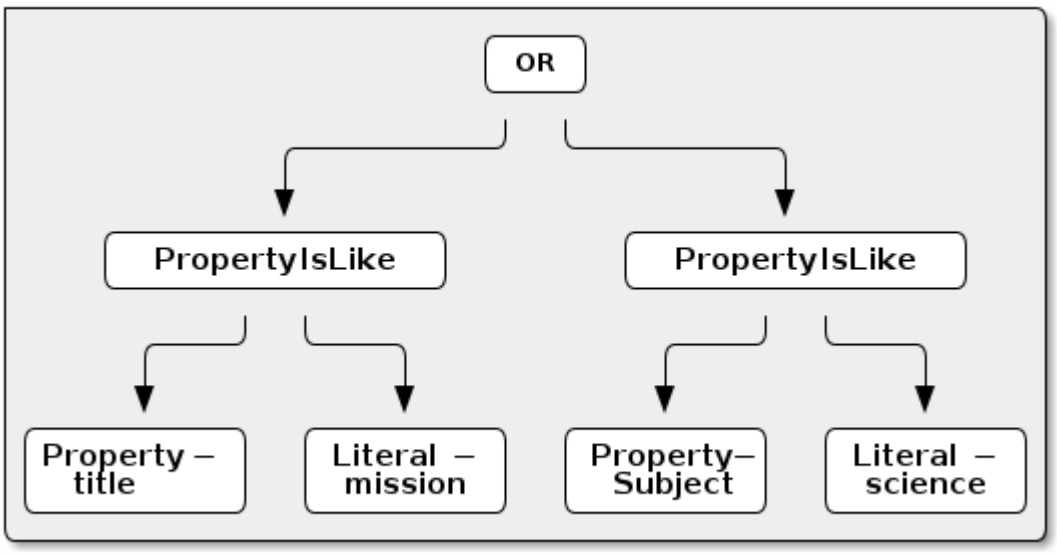


Figure 36. Parsing Filters XQuery

24.19.3.2.1. FilterAdapter/Delegate Process for Figure Parsing

1. `FilterAdapter` visits the `OR` filter first.
2. `OR` filter visits its children in a loop.
3. The first child in the loop that is encountered is the LHS `PropertyIsLike`.
4. The `FilterAdapter` calls the `FilterDelegate PropertyIsLike` method with the LHS property and literal.
5. The LHS `PropertyIsLike` delegate method builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject` Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
6. The `FilterAdapter` then moves back to the `OR` filter, which visits its second child.
7. The `FilterAdapter` calls the `FilterDelegate PropertyIsLike` method with the RHS property and literal.
8. The RHS `PropertyIsLike` delegate method builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject` Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
9. The `FilterAdapter` then moves back to its `OR` Filter which is now done with its children.
10. It then collects the output of each child and sends the list of results to the `FilterDelegate OR` method.
11. The final result object is returned from the `FilterAdapter` adapt method.

24.19.3.2.2. FilterVisitor Process for Figure Parsing

1. FilterVisitor visits the `OR` filter first.
2. `OR` filter visits its children in a loop.
3. The first child in the loop that is encountered is the LHS `PropertyIsLike`.
4. The LHS `PropertyIsLike` builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject`. Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.
5. The FilterVisitor then moves back to the `OR` filter, which visits its second child.
6. The RHS `PropertyIsLike` builds the XQuery syntax that makes sense for this particular underlying object store. In this case, the *subject* property is specific to this XML database, and the business logic maps the *subject* property to its index at `//inventory:book/@subject`. Note that `ft:query` in this instance is a custom XQuery module for this specific XML database that does full text searches.

7. The FilterVisitor then moves back to its `OR` filter, which is now done with its children. It then collects the output of each child and could potentially execute the following code to produce the above expression.

```
public visit( Or filter, Object data) {
...
   /* the equivalent statement for the OR filter in this domain (XQuery) */
   xQuery = childFilter1Output + " union " + childFilter2Output;
...
}
```

24.19.4. Filter Profile

The filter profile maps filters to metacard types.

24.19.4.1. Role of the OGC Filter

Both Queries and Subscriptions extend the OGC GeoAPI Filter interface.

The Filter Builder and Adapter do not fully implement the OGC Filter Specification. The filter support profile contains suggested filter to metacard type mappings. For example, even though a Source could support a `PropertyIsGreaterThan` filter on `XML_TYPE`, it would not likely be useful.

24.19.4.2. Catalog Filter Profile

The following table displays the common metacard attributes with their respective types for reference.

Table 80. Metacard Attribute To Type Mapping

Metacard Attribute	Metacard Type
ANY_DATE	DATE_TYPE
ANY_GEO	GEO_TYPE
ANY_TEXT	STRING_TYPE
CONTENT_TYPE	STRING_TYPE
CONTENT_TYPE_VERSION	STRING_TYPE
CREATED	DATE_TYPE
EFFECTIVE	DATE_TYPE
GEOGRAPHY	GEO_TYPE
ID	STRING_TYPE
METADATA	XML_TYPE
MODIFIED	DATE_TYPE

Metacard Attribute	Metacard Type
RESOURCE_SIZE	STRING_TYPE
RESOURCE_URI	STRING_TYPE
SOURCE_ID	STRING_TYPE
TARGET_NAMESPACE	STRING_TYPE
THUMBNAIL	BINARY_TYPE
TITLE	STRING_TYPE

24.19.4.2.1. Comparison Operators

Comparison operators compare the value associated with a property name with a given Literal value. Endpoints and sources should try to use metacard types other than the object type. The object type only supports backwards compatibility with `java.net.URI`. Endpoints that send other objects are not supported by standard sources. The following table maps the metacard types to supported comparison operators.

Table 81. Metacard Types to Comparison Operators

PropertyIs	Between	EqualTo	GreaterThan	GreaterThan	OrEqualTo	LessThan	LessThan	OrEqualTo	Like	NotEqualTo	Null
BINARY_TYPE		X									
BOOLEAN_TYPE		X									
DATE_TYPE	X	X	X	X	X	X	X	X		X	X
DOUBLE_TYPE	X	X	X	X	X	X	X	X		X	X
FLOAT_TYPE	X	X	X	X	X	X	X	X		X	X
GEO_TYPE											X
INTEGER_TYPE	X	X	X	X	X	X	X	X		X	X
LONG_TYPE	X	X	X	X	X	X	X	X		X	X

PropertyIs	Between	EqualTo	GreaterThan	GreaterThan	OrEqualTo	LessThan	LessThan	OrEqualTo	Like	NotEqualTo	Null
OBJECT_TYPE	X	X	X	X	X	X	X	X		X	X
SHORT_TYPE	X	X	X	X	X	X	X	X		X	X
STRING_TYPE	X	X	X	X	X	X	X	X	X	X	X
XML_TYPE		X							X		X

Table 82. Comparison Operators

Operator	Description
PropertyIsBetween	Lower ⇐ Property ⇐ Upper
PropertyIsEqualTo	Property == Literal
PropertyIsGreaterThan	Property > Literal
PropertyIsGreaterThanOrEqualTo	Property >= Literal
PropertyIsLessThan	Property < Literal
PropertyIsLessThanOrEqualTo	Property ⇐ Literal
PropertyIsLike	Property LIKE Literal Equivalent to SQL "like"
PropertyIsNotEqualTo	Property != Literal
PropertyIsNull	Property == null

24.19.4.2.2. Logical Operators

Logical operators apply Boolean logic to one or more child filters.

Table 83. Supported Logical Operators

	And	Not	Or
Supported Filters	X	X	X

24.19.4.2.3. Temporal Operators

Temporal operators compare a date associated with a property name to a given Literal date or date range.

Table 84. Supported Temporal Operators

	After	AnyInteracts	Before	Begins	BegunBy	During	EndedBy	Meets	MetBy	OverlappedBy	TContains
DATE_TYPE	**X**		**X**			**X**					

Literal values can be either date instants or date periods.

Table 85. Temporal Operator Descriptions

Operator	Description
After	Property > (Literal || Literal.end)
Before	Property < (Literal || Literal.start)
During	Literal.start < Property < Literal.end

24.19.4.2.4. Spatial Operators

Spatial operators compare a geometry associated with a property name to a given Literal geometry.

Table 86. Supported Spatial Operators.

BBox	Beyond	Contains	Crosses	Disjoint	Equals	DWithin	Intersects	Overlaps	Touches	Within
GEO_TYPE		**X**	**X**	**X**	**X**		**X**	**X**	**X**	

Geometries are usually represented as Well-Known Text (*WKT*).

Table 87. Spatial Operator Descriptions

Operator	Description
Beyond	Property geometries beyond given distance of Literal geometry
Contains	Property geometry contains Literal geometry
Crosses	Property geometry crosses Literal geometry
Disjoint	Property geometry direct positions are not interior to Literal geometry
DWithin	Property geometry lies within distance to Literal geometry
Intersects	Property geometry intersects Literal geometry; opposite to the Disjoint operator

Operator	Description
Overlaps	Property geometry interior overlaps Literal geometry interior somewhere
Touches	Property geometry touches but does not overlap Literal geometry
Within	Property geometry completely contains Literal geometry

24.20. Developing Filter Delegates

Filter Delegates help reduce the complexity of parsing OGC Filters. The reference Filter Adapter implementation contains the necessary boilerplate visitor code and input normalization to handle commonly supported OGC Filters.

24.20.1. Creating a New Filter Delegate

A Filter Delegate contains the logic that converts normalized filter input into a form that the target data source can handle. Delegate methods are called in a depth-first order as the Filter Adapter visits filter nodes.

24.20.1.1. Implementing the Filter Delegate

1. Create a Java class extending `FilterDelegate`.
   `public class ExampleDelegate extends DDF.catalog.filter.FilterDelegate<ExampleReturnObjectType> {`
2. `FilterDelegate` throws an appropriate exception for all methods not implemented. Refer to the DDF JavaDoc for more details about what is expected of each `FilterDelegate` method.

NOTE	A code example of a Filter Delegate can be found in `DDF.catalog.filter.proxy.adapter.test` of the `filter-proxy` bundle.

24.20.1.2. Throwing Exceptions

Filter delegate methods can throw `UnsupportedOperationException` run-time exceptions. The `GeotoolsFilterAdapterImpl` catches and re-throws these exceptions as `UnsupportedQueryExceptions`.

24.20.1.3. Using the Filter Adapter

The FilterAdapter can be requested from the OSGi registry.

```
<reference id="filterAdapter" interface="DDF.catalog.filter.FilterAdapter" />
```

The Query in a QueryRequest implements the Filter interface. The Query can be passed to a `FilterAdapter` and `FilterDelegate` to process the Filter.

```
@Override
public DDF.catalog.operation.QueryResponse query(DDF.catalog.operation.QueryRequest
queryRequest)
    throws DDF.catalog.source.UnsupportedQueryException {

    DDF.catalog.operation.Query query = queryRequest.getQuery();

    DDF.catalog.filter.FilterDelegate<ExampleReturnObjectType> delegate = new
ExampleDelegate();

    // DDF.catalog.filter.FilterAdapter adapter injected via Blueprint
    ExampleReturnObjectType result = adapter.adapt(query, delegate);
}
```

Import the Catalog API Filter package and the reference implementation package of the Filter Adapter in the bundle manifest (in addition to any other required packages).
`Import-Package: DDF.catalog, DDF.catalog.filter, DDF.catalog.source`

24.20.1.4. Filter Support

Not all OGC Filters are exposed at this time. If demand for further OGC Filter functionality is requested, it can be added to the Filter Adapter and Delegate so sources can support more complex filters. The following OGC Filter types are currently available:

Logical
And
Or
Not
Include
Exclude

Property Comparison
`PropertyIsBetween`
`PropertyIsEqualTo`
`PropertyIsGreaterThan`
`PropertyIsGreaterThanOrEqualTo`
`PropertyIsLessThan`
`PropertyIsLessThanOrEqualTo`

Property Comparison
`PropertyIsLike`
`PropertyIsNotEqualTo`
`PropertyIsNull`

`Spatial`	Definition
`Beyond`	True if the geometry being tested is beyond the stated distance of the geometry provided.
`Contains`	True if the second geometry is wholly inside the first geometry.
`Crosses`	True if: * the intersection of the two geometries results in a value whose dimension is less than the geometries * the maximum dimension of the intersection value includes points interior to both the geometries * the intersection value is not equal to either of the geometries.
`Disjoint`	True if the two geometries do not touch or intersect.
`DWithin`	True if the geometry being tested is within the stated distance of the geometry provided.
`Intersects`	True if the two geometries intersect. This is a convenience method as `Not Disjoint(A,B)` gets the same result.
`Overlaps`	True if the intersection of the geometries results in a value of the same dimension as the geometries that is different from both of the geometries.
`Touches`	True if and only if the only common points of the two geometries are in the union of the boundaries of the geometries.
`Within`	True if the first geometry is wholly inside the second geometry.

Temporal
After
Before
During

24.21. Developing Action Components

To provide a service, such as a link to a metacard, the `ActionProvider` interface should be implemented. An `ActionProvider` essentially provides a List of `Actions` when given input that it can recognize and handle. For instance, if a REST endpoint ActionProvider was given a metacard, it could provide a link based on the metacard's ID. An Action Provider performs an action when given a subject that it understands. If it does not understand the subject or does not know how to handle the given input, it returns `Collections.emptyList()`. An Action Provider is required to have an ActionProvider id. The

Action Provider must register itself in the OSGi Service Registry with the `ddf.action.ActionProvider` interface and must also have a service property value for `id`. An action is a URL that, when invoked, provides a resource or executes intended business logic.

24.21.1. Action Component Naming Convention

For each Action, a title and description should be provided to describe what the action does. The recommended naming convention is to use the verb 'Get' when retrieving a portion of a metacard, such as the metadata or thumbnail, or when downloading a resource. The verb 'Export' or the expression 'Export as' is recommended when the metacard is being exported in a different format or presented after going some transformation.

24.21.1.1. Action Component Taxonomy

An Action Provider registers an `id` as a service property in the OGSi Service Registry based on the type of service or action that is provided. Regardless of implementation, if more than one Action Provider provides the same service, such as providing a URL to a thumbnail for a given metacard, they must both register under the same `id`. Therefore, Action Provider implementers must follow an Action Taxonomy.

The following is a sample taxonomy:

1. `catalog.data.metacard` shall be the grouping that represents Actions on a Catalog metacard.
    a. `catalog.data.metacard.view`
    b. `catalog.data.metacard.thumbnail`
    c. `catalog.data.metacard.html`
    d. `catalog.data.metacard.resource`
    e. `catalog.data.metacard.metadata`

Table 88. Action ID Service Descriptions

ID	Required Action	Naming Convention
`catalog.data.metacard.view`	Provides a valid URL to view a metacard. Format of data is not specified, so the representation can be in XML, JSON, or other.	Export as ...
`catalog.data.metacard.thumbnail`	Provides a valid URL to the bytes of a thumbnail (`Metacard.THUMBNAIL`) with MIME type image/jpeg.	Export as Thumbnail
`catalog.data.metacard.map.overlay.thumbnail`	Provides a metacard URL that translates the metacard into a geographically aligned image (suitable for overlaying on a map).	Export as Thumbnail Overlay
`catalog.data.metacard.html`	Provides a valid URL that, when invoked, provides an HTML representation of the metacard.	Export as HTML

ID	Required Action	Naming Convention
catalog.data.metacard.xml	Provides a valid URL that, when invoked, provides an XML representation of the metacard.	Export as XML
catalog.data.metacard.geojson	Provides a valid URL that, when invoked, provides an XML representation of the metacard.	Export as GeoJSON
catalog.data.metacard.resource	Provides a valid URL that, when invoked, provides the underlying resource of the metacard.	Export as Resource
catalog.data.metacard.metadata	Provides a valid URL to the XML metadata in the metacard (Metacard.METADATA).	Export as Metadata

24.22. Developing Query Options

The easiest way to create a Query is to use the ddf.catalog.operation.QueryImpl object. It is first necessary to create an OGC Filter object then set the Query Options after QueryImpl has been constructed.

QueryImpl *Example*

```
/*
  Builds a query that requests a total results count and
  that the first record to be returned is the second record found from
  the requested set of metacards.
 */

 String property = ...;

 String value = ...;

 org.geotools.filter.FilterFactoryImpl filterFactory = new FilterFactoryImpl() ;

 QueryImpl query = new QueryImpl( filterFactory.equals(filterFactory.property(property),

filterFactory.literal(value))) ;

 query.setStartIndex(2) ;

 query.setRequestsTotalResultsCount(true);
```

24.22.1. Evaluating a query

Every Source must be able to evaluate a Query object. Nevertheless, each Source could evaluate the Query differently depending on what that Source supports as to properties and query capabilities. For

instance, a common property all Sources understand is id, but a Source could possibly store frequency values under the property name "frequency." Some Sources may not support frequency property inquiries and throw an error stating it cannot interpret the property. In addition, some Sources might be able to handle spatial operations, while others might not. A developer should consult a Source's documentation for the limitations, capabilities, and properties that a Source can support.

Table 89. Query Options

Option	Description
`StartIndex`	1-based index that states which metacard the Source should return first out of the requested metacards.
`PageSize`	Represents the maximum amount of metacards the Source should return.
`SortBy`	Determines how the results are sorted and on which property.
`RequestsTotalResultsCount`	Determines whether the total number of results should be returned.
`TimeoutMillis`	The amount of time in milliseconds before the query is to be abandoned. If a zero or negative timeout is set, the catalog framework defaults to a value configurable via the Admin UI under Catalog → Configuration → Query Operations.

24.22.2. Commons-DDF Utilities

The `commons-DDF` bundle provides utilities and functionality commonly used across other DDF components, such as the endpoints and providers.

24.22.2.1. FuzzyFunction

`DDF.catalog.impl.filter.FuzzyFunction` class is used to indicate that a `PropertyIsLike` filter should interpret the search as a fuzzy query.

24.22.2.2. XPathHelper

`DDF.util.XPathHelper` provides convenience methods for executing XPath operations on XML. It also provides convenience methods for converting XML as a `String` from a `org.w3c.dom.Document` object and vice versa.

24.23. Configuring Managed Service Factory Bundles

Services that are created using a Managed Service Factory can be configured using `.config` files as well. These configuration files, however, follow a different naming convention than `.cfg` files. The filenames

must start with the Managed Service Factory PID, be followed by a dash and a unique identifier, and have a `.config` extension. For instance, assuming that the Managed Service Factory PID is `org.codice.ddf.factory.pid` and two instances of the service need to be configured, files `org.codice.ddf.factory.pid-<UNIQUE ID 1>.config` and `org.codice.ddf.factory.pid-<UNIQUE ID 2>.config` should be created and added to `<DDF_HOME>/etc`.

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

Any `service.factoryPid` and `service.pid` values in these `.config` files are overridden by the values parsed from the file name, so `.config` files should not contain these properties.

24.23.1. File Format

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

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

Table 90. Property Formats

Type	Format (see details below for variations)	Example
String	`name="value"`	`name="John"`
Boolean	`name=B"true|false"`	`authorized=B"true"`
Integer	`name=I"value"`	`timeout=I"10"`
Long	`name=L"value"`	`diameter=L"100"`
Float	`name=F"value"`	`cost=F"1093140480"`
Double	`name=D"value"`	`latitude=D"4636745974857667812"`
List of Strings	`name=["value1","value2",…]`	`complexStringArray=[ \ "{\"url\"\ \"http://test.sample.com\"\ \"layers\"\ [\"0\"]\ \"VERSION\"\ \"1.1|1.2\"\ \"image/png\"}\ \"beta\"\ 1}", \ "{\"url\"\ \"http://test.sample.com\"\ 0.5}", \ "/security-config\=SAML|basic", \ ]`

Type	Format (see details below for variations)	Example

List of Booleans	`name=B["true|false","true|false",…]`	`authorizedList=B[ \`
		`  "true", \`
		`  "false", \`
		`  ]`

List of Integers	`name=I["value1","value2",…]`	`sizes=I[ \`
		`  "10", \`
		`  "20", \`
		`  "30", \`
		`  ]`

List of Longs	`name=L["value1","value2",…]`	`sizes=L[ \`
		`  "100", \`
		`  "200", \`
		`  "300", \`
		`  ]`

List of Floats	`name=F["value1","value2",…]`	`sizes=F[ \`
		`  "1066192077", \`
		`  "1074580685", \`
		`  "1079194419", \`
		`  ]`

List of Doubles	`name=D["value1","value2",…]`	`sizes=D[ \`
		`  "4607736361554183979", \`
		`  "4612212939583790252", \`
		`  "4614714689176794563", \`
		`  ]`

NOTE

- Values with types other than String must be prefixed with a lower-case or upper-case character. See the examples in the table.
  - Boolean: `B` or `b`
  - Integer: `I` or `i`
  - Long: `L` or `l`
  - Float: `F` or `f`
  - Double: `D` or `d`

- Equal signs (=), double quotes ("), and spaces within values must must be escaped using a backslash (\).
- When properties are split over multiple lines for readability, end of lines must be specified with a backslash (\). See the examples for lists in the table.
- A comma (,) after the last value in a list is optional.
- Surrounding the equal signs (=) with spaces for properties is optional. Because there is a known issue when using OPS4J Pax Exam 4.11.0 and modifying `.config` files that include spaces, all default `.config` files that may be modified in OPS4J Pax Exam 4.11.0 tests should not include spaces.
- Boolean values default to `false` if any value other than `true` is provided.
- Float values must be representated in the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values. For example, `F"1093140480"` corresponds to `F"10.5"`. See the documentation for `java.lang.Integer#parseInt(java.lang.String)` and `java.lang.Float#intBitsToFloat(int)` for more details.
- Double values must be represented in the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values. For example, `D"4636745974857667812"` corresponds to `D"100.1234"`. See the documentation for `java.lang.Long#parseLong(java.lang.String)` and `java.lang.Double#longBitsToDouble` for more details.

Sample configuration file

```
webAuthenticationTypes="basic"
endpointAuthenticationTypes=""
sessionAccess=B"true"
guestAccess=B"true"
realms=[ \
  "/\=karaf", \
  ]
requiredAttributes=[ \
  "/\=", \
  "/admin\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\=admin}", \
  "/system\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\=admin}", \
  "/security-
config\={http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role\=admin}", \
  ]
whiteListContexts=[ \
  "/services/SecurityTokenService", \
  "/services/internal/metrics", \
  "/services/saml", \
  "/proxy", \
  "/services/csw", \
```

```
]
```

24.24. Developing XACML Policies

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

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

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

24.24.1. XACML Policy Attributes

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

24.24.2. XACML Policy Subject

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

24.24.3. XACML Policy Resource

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

24.24.4. Using a XACML Policy

To use a XACML policy, copy the XACML policy into the `<DDF_HOME>/etc/pdp/policies` directory.

24.25. Assuring Authenticity of Bundles and Applications

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

24.25.1. Prerequisites

To work with Java signatures, a keystore/truststore is required. For testing or trial purposes DDF can sign and validate using a self-signed certificate, generated with the keytool utility. In an actual installation, a certificate issued from a trusted Certificate Authority is used.

Additional documentation on keytool can be found at Keytool home.

Using keytool to generate a self-signed certificate keystore

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

24.25.2. Signing a JAR/KAR

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

Additional documentation on jarsigner can be found at Jarsigner.

Using jarsigner to sign a KAR

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

24.25.2.1. Verifying a JAR/KAR

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

Using jarsigner to verify a file

```
~ $ jarsigner -verify -verbose -keystore keystore.jks catalog-app-2.5.1.kar
        9447 Mon Oct 06 17:05:46 MST 2014 META-INF/MANIFEST.MF
        9503 Mon Oct 06 17:05:46 MST 2014 META-INF/SELFSIGN.SF

[... section abbreviated for space]

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

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

24.26. WFS Services

The Web Feature Service (WFS) is an Open Geospatial Consortium (OGC) Specification. DDF supports the ability to integrate WFS 1.1, and 2.0 Web Services.

NOTE	DDF does not include a supported WFS Web Service (Endpoint) implementation. Therefore, federation for 2 DDF instances is not possible via WFS.

WFS Features

When a query is issued to a WFS server, the output of the query is an XML document that contains a collection of feature member elements. Each WFS server can have one or more feature types with each type being defined by a schema that extends the WFS `featureMember` schema. The schema for each type can be discovered by issuing a `DescribeFeatureType` request to the WFS server for the feature type in question. The WFS source handles WFS capability discovery and requests for feature type description when an instance of the WFS source is configured and created.

See the WFS v1.1.0 Source, or WFS v2.0.0 Source for more information about how to configure a WFS source.

Converting a WFS Feature

In order to expose WFS features to DDF clients, the WFS feature must be converted into the common data format of the DDF, a metacard. The OGC package contains a `GenericFeatureConverter` that attempts

to populate mandatory metacard fields with properties from the WFS feature XML. All properties are mapped directly to new attributes in the metacard. However, the `GenericFeatureConverter` may not be able to populate the default metacard fields with properties from the feature XML.

Creating a Custom Converter

To more accurately map WFS feature properties to fields in the metacard, a custom converter can be created. The OGC package contains an interface, `FeatureConverter`, which extends the http://xstream.codehaus.org/javadoc/com/thoughtworks/xstream/converters/Converter.htmlConverter interface provided by the XStream project. XStream is an open source API for serializing XML into Java objects and vice-versa. Additionally, a base class, `AbstractFeatureConverter`, has been created to handle the mapping of many fields to reduce code duplication in the custom converter classes.

1. Create the `CustomConverter` class extending the `ogc.catalog.common.converter.AbstractFeatureConverter` class.

```
public class CustomConverter extends ogc.catalog.common.converter
.AbstractFeatureConverter
```

2. Implement the `FeatureConverterFactory` interface and the `createConverter()` method for the `CustomConverter`.

```
public class CustomConverterFactory implements FeatureConverterFactory {
    private final featureType;
    public CustomConverterFactory(String featureType) {
        this.featureType = featureType;
    }
    public FeatureConverter createConverter() {
        return new CustomConverter();
    }
    public String getFeatureType() {
        return featureType;
    }
}
```

3. Implement the `unmarshal` method required by the `FeatureConverter` interface. The `createMetacardFromFeature(reader, metacardType)` method implemented in the `AbstractFeatureConverter` is recommended.

```
public Metacard unmarshal(HierarchicalStreamReader reader, UnmarshallingContext ctx) {
  MetacardImpl mc = createMetacardFromFeature(reader, metacardType);
  //set your feature specific fields on the metacard object here
  //
  //if you want to map a property called "beginningDate" to the Metacard.createdDate
field
```

```
    //you would do:
    mc.setCreatedDate(mc.getAttribute("beginningDate").getValue());
  }
```

4. Export the `ConverterFactory` to the OSGi registry by creating a `blueprint.xml` file for its bundle. The bean id and argument value must match the WFS Feature type being converted.

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:cm=
"http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.1.0">
  <bean id="custom_type" class="com.example.converter.factory.CustomConverterFactory">
    <argument value="custom_type"/>
  </bean>
  <service ref="custom_type" interface=
"ogc.catalog.common.converter.factory.FeatureConverterFactory"/>
</blueprint>
```

24.27. JSON Definition Files

DDF supports adding new attribute types, metacard types, validators, and more using json-formatted definition files.

The following may be defined in a JSON definition file:

- Attribute Types
- Metacard Types
- Global Attribute Validators
- Metacard Validators
- Default Attribute Values
- Attribute Injections

24.27.1. Definition File Format

A definition file follows the JSON format as specified in ECMA-404. All definition files must be valid JSON in order to be parsed.

A single definition file may define as many of the types as needed. This means that types can be defined across multiple files for grouping or clarity.

24.27.2. Deploying Definition Files

The file must have a `.json` extension in order to be picked up by the deployer. Once the definition file is

ready to be deployed, put the definition file `<filename>.json` into the `etc/definitions` folder.

Definition files can be added, updated, and/or deleted in the `etc/definitions` folder. The changes are applied dynamically and no restart is required.

If a definition file is removed from the `etc/definitions` folder, the changes that were applied by that file are undone.

24.28. Developing Subscriptions

Subscriptions represent "standing queries" in the Catalog. Like a query, subscriptions are based on the OGC Filter specification.

24.28.1. Subscription Lifecycle

A Subscription itself is a series of events during which various plugins or transformers can be called to process the subscription.

24.28.1.1. Creation

- Subscriptions are created directly with theEvent Processor or declaratively through use of the Whiteboard Design Pattern.
- The Event Processor invokes each Pre-Subscription Plugin and, if the subscription is not rejected, the subscription is activated.

24.28.1.2. Evaluation

- When a metacard matching the subscription is created, updated, or deleted in any Source, each Pre-Delivery Plugin is invoked.
- If the delivery is not rejected, the associated Delivery Method callback is invoked.

24.28.1.3. Update Evaluation

Notably, the Catalog allows event evaluation on both the previous value (if available) and new value of a Metacard when an update occurs.

24.28.1.4. Durability

Subscription durability is not provided by the Event Processor. Thus, all subscriptions are transient and not recreated in the event of a system restart.

It is the responsibility of Endpoints using subscriptions to persist and re-establish the subscription on startup.

This decision was made for the sake of simplicity, flexibility, and the inability of the Event Processor to recreate a fully configured Delivery Method without being overly restrictive.

IMPORTANT	**Subscriptions are not persisted by the Catalog itself.** Subscriptions must be explicitly persisted by an endpoint and are not persisted by the Catalog. The Catalog Framework, or more specifically the Event Processor itself, does not persist subscriptions. Certain endpoints, however, can persist the subscriptions on their own and recreate them on system startup.

24.28.2. Creating a Subscription

Currently, the Catalog reference implementation does not contain a subscription endpoint. Therefore, an endpoint that exposes a web service interface to create, update, and delete subscriptions would provide a client's subscription filtering criteria to be used by Catalog's Event Processor to determine which events are of interest to the client. The endpoint client also provides the callback URL of the event consumer to be called when an event matching the subscription's criteria is found. This callback to the event consumer is made by a Delivery Method implementation that the client provides when the subscription is created. Whenever an event occurs in the Catalog matching the subscription, the Delivery Method implementation is called by the Event Processor. The Delivery Method, in turn, sends the event notification out to the event consumer. As part of the subscription creation process, the Catalog verifies that the event consumer at the specified callback URL is available to receive callbacks. Therefore, the client must ensure the event consumer is running prior to creating the subscription. The Catalog completes the subscription creation by executing any pre-subscription Catalog Plugins, and then registering the subscription with the OSGi Service Registry. The Catalog does not persist subscriptions by default.

24.28.2.1. Event Processing and Notification

If an event matches a subscription's criteria, any pre-delivery plugins that are installed are invoked, the subscription's `DeliveryMethod` is retrieved, and its operation corresponding to the type of ingest event is invoked. For example, the `DeliveryMethod created()` function is called when a metacard is created. The `DeliveryMethod` operations subsequently invoke the corresponding operation in the client's event consumer service, which is specified by the callback URL provided when the `DeliveryMethod` was created. An internal subscription tracker monitors the OSGi registry, looking for subscriptions to be added (or deleted). When it detects a subscription being added, it informs the Event Processor, which sets up the subscription's filtering and is responsible for posting event notifications to the subscriber when events satisfying their criteria are met.

The Standard Event Processor is an implementation of the Event Processor and provides the ability to create/delete subscriptions. Events are generated by the CatalogFramework as metacards are created/updated/deleted and the Standard Event Processor is called since it is also a Post-Ingest Plugin. The Standard Event Processor checks each event against each subscription's criteria.

When an event matches a subscription's criteria, the Standard Event Processor invokes:

- each pre-delivery plugin on the metacard in the event.
- the `DeliveryMethod` operation corresponding to the type of event being processed, such as the `created()` operation for the creation of a metacard.

Available Event Processor

- Standard Event Processor

24.28.2.1.1. Using DDF Implementation

If applicable, the implementation of `Subscription` that comes with DDF should be used. It is available at `ddf.catalog.event.impl.SubscriptionImpl` and offers a constructor that takes in all of the necessary objects. Specifically, all that is needed is a `Filter`, `DeliveryMethod`, `Set<String>` of source IDs, and a `boolean` for enterprise.

The following is an example code stub showing how to create a new instance of Subscription using the DDF implementation.

Creating a Subscription

```
// Create a new filter using an imported FilterBuilder
Filter filter = filterBuilder.attribute(Metacard.ANY_TEXT).like().text("*");

// Create a implementation of DeliveryMethod
DeliveryMethod deliveryMethod = new MyCustomDeliveryMethod();

// Create a set of source ids
// This set is empty as the subscription is not specific to any sources
Set<String> sourceIds = new HashSet<String>();

// Set the isEnterprise boolean value
// This subscription example should notifications from all sources (not just local)
boolean isEnterprise = true;

Subscription subscription = new SubscriptionImpl(filter, deliveryMethod, sourceIds
,isEnterprise);
```

24.28.2.2. Delivery Method

A Delivery Method provides the operation (created, updated, deleted) for how an event's metacard can be delivered.

A Delivery Method is associated with a subscription and contains the callback URL of the event consumer to be notified of events. The Delivery Method encapsulates the operations to be invoked by the Event Processor when an event matches the criteria for the subscription. The Delivery Method's operations are responsible for invoking the corresponding operations on the event consumer associated with the callback URL.

24.29. Contributing to Documentation

DDF documentation is included in the source code, so it is edited and maintained in much the same

way.

`src/main/resources`

Table 91. Documentation Directory Structure and Contents

Directory	Contents
`content`	Asciidoctor-formatted files containing documentation contents and the header information needed to organize them.
`images`	Screenshots, icons, and other image files used in documentation.
`templates`	Template files used to compile the documentation for display.
`jbake.properties`	Properties file defining content types and other parameters.

24.29.1. Editing Existing Documentation

Update existing content when code behavior changes, new capabilities are added to features, or the configuration process changes. Content is organized within the `content` directory in sub directories according to the audience and purpose for each document in the documentation library. Use this list to determine placement of new content.

Documentation Sections

Introduction/Core Concepts

This section is intended to be a high-level, executive summary of the features and capabilities of DDF. Content here should be written at a non-technical level.

Quick Start

This section is intended for getting set up with a test, demonstration, or trial instance of DDF. This is the place for non-production shortcuts or workarounds that would not be used in a secured, hardened installation.

Managing

The managing section covers "how-to" instructions to be used to install, configure, and maintain an instance of DDF in a production environment. This content should be aimed at system administrators. Security hardening should be integrated into these sections.

Using

This section is primarily aimed at the final end users who perform tasks with DDF. This content should guide users through common tasks and user interfaces.

Integrating

This section guides developers building other projects looking to connect to new or existing instances of DDF.

Developing

This section provides guidance and best practices on developing custom implementations of DDF components, especially ones that may be contributed into the code baseline.

Architecture

This section is a detailed description of the architectural design of DDF and how components work together.

Reference

This section is a comprehensive list of features and possible configurations.

Metadata Reference

This section details how metadata is extracted and normalized by DDF.

Documentation

This is a collection of all of the individual documentation pages in one html or pdf file.

See the style guide for more guidance on stylistic and formatting concerns.

24.29.2. Adding New Documentation Content

If creating a new section is required, there are some minimal requirements for a new `.adoc` file.

Header content

The templates scan the header information to place it into the correct place within the documentation. Different sections have different headers required, but some common attributes are always required.

- `type`: roughly maps to the section or subSection of the documentation.
- `title`: title of the section or subsection contained in the file.
- `status`: set to `published` to include within the documentation, set to `draft` to hide a work-in-progress section.
- `order`: used in sections where order needs to be enforced.
- `summary`: brief summary of section contents. Some, but not all, summaries are included by templates.

24.29.3. Creating a New Documentation Template

To create a new, standalone documentation page, create a new template in the `templates` directory. Optionally, this template can `include` some of the internal templates in the `templates/build` directory, but this is not required.

For guidance on using the freemarker syntax, see the Freemarker documentation.

24.29.4. Extending Documentation in Downstream Distributions

By mimicking the build and directory structure of the documentation, downstream projects are able to leverage the existing documentation and insert content before and after sections of the DDF documentation.

Documentation Module Directory Structure

```
-docs
  -src
    -main
      -resources
        -content
        -images
        -templates
```

`content`

Contains the .adoc files that make up the content. Sub-directories are organized according to the documents that make up the main library.

`images`

any pre-existing images, such as screenshots, to be included in the documentation.

`templates`

template files used to create documentation artifacts. A `build` sub-directory holds the templates that are not standalone documents to render specific sections.

25. Development Guidelines

25.1. Contributing

The Distributed Data Framework is free and open-source software offered under the GNU Lesser General Public License. The DDF is managed under the guidance of the Codice Foundation. Contributions are welcomed and encouraged. Please visit the Codice DDF Contributor Guidelines and the DDF source code repository for more information.

25.2. OSGi Basics

DDF runs on top of an OSGi framework, a Java virtual machine (JVM), several choices of operating systems, and the physical hardware infrastructure. The items within the dotted line represent the standard DDF components.

DDF is a customized and branded distribution of Apache Karaf. DDF could also be considered to be a more lightweight OSGi distribution, as compared to Apache ServiceMix, FUSE ESB, or Talend ESB, all of

which are also built upon Apache Karaf. Similar to its peers, DDF incorporates (additional upstream dependencies).

The DDF framework hosts DDF applications, which are extensible by adding components via OSGi. The best example of this is the DDF Catalog (API), which offers extensibility via several types of Catalog Components. The DDF Catalog API serves as the foundation for several applications and resides in the applications tier.

The Catalog Components consist of Endpoints, Plugins, Catalog Frameworks, Sources, and Catalog Providers. Customized components can be added to DDF.

Capability

A general term used to refer to an ability of the system.

Component

Represents a portion of an Application that can be extended.

Bundle

Java Archives (JARs) with special OSGi manifest entries.

Feature

One or more bundles that form an installable unit; defined by Apache Karaf but portable to other OSGi containers.

Application

A JSON file defining a collection of bundles with configurations to be displayed in the Admin Console.

25.2.1. Packaging Capabilities as Bundles

Services and code are physically deployed to DDF using bundles. The bundles within DDF are created using the maven bundle plug-in. Bundles are Java JAR files that have additional metadata in the `MANIFEST.MF` that is relevant to an OSGi container.

The best resource for learning about the structure and headers in the manifest definition is in section 3.6 of the OSGi Core Specification. The bundles within DDF are created using the maven bundle plug-in, which uses the BND tool.

TIP

Alternative Bundle Creation Methods

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

25.2.1.1. Creating a Bundle

25.2.1.1.1. Bundle Development Recommendations

Avoid creating bundles by hand or editing a manifest file

Many tools exist for creating bundles, notably the Maven Bundle plugin, which handle the details of OSGi configuration and automate the bundling process including generation of the manifest file.

Always make a distinction on which imported packages are `optional` or `required`

Requiring every package when not necessary can cause an unnecessary dependency ripple effect among bundles.

Embedding is an implementation detail

Using the `Embed-Dependency` instruction provided by the `maven-bundle-plugin` inserts the specified jars into the target archive and adds them to the `Bundle-ClassPath`. These jars and their contained packages/classes are not for public consumption; they are for the internal implementation of this service implementation only.

Bundles should never be embedded

Bundles expose service implementations; they do not provide arbitrary classes to be used by other bundles.

Bundles should expose service implementations

This is the corollary to the previous rule. Bundles should not be created when arbitrary concrete classes are being extracted to a library. In that case, a library/jar is the appropriate module packaging type.

Bundles should generally *only* export service packages

If there are packages internal to a bundle that comprise its implementation but not its public manifestation of the API, they should be excluded from export and kept as private packages.

Concrete objects that are not loaded by the root classloader should not be passed in or out of a bundle

This is a general rule with some exceptions (JAXB generated classes being the most prominent example). Where complex objects need to be passed in or out of a service method, an interface should be defined in the API bundle.

Bundles separate contract from implementation and allow for modularized development and deployment of functionality. For that to be effective, they must be defined and used correctly so inadvertent coupling does not occur. Good bundle definition and usage leads to a more flexible environment.

25.2.1.1.2. Maven Bundle Plugin

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

Maven `pom.xml`

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

25.2.1.2. Third Party and Utility Bundles

It is recommended to avoid building directly on included third party and utility bundles. These components do provide utility and reuse potential; however, they may be upgraded or even replaced at anytime as bug fixes and new capabilities dictate. For example, web services may be built using CXF. However, the distributions frequently upgrade CXF between releases to take advantage of new features. If building on these components, be aware of the version upgrades with each distribution release.

Instead, component developers should package and deliver their own dependencies to ensure future compatibility. For example, if re-using a bundle, the specific bundle version that you are depending on should be included in your packaged release, and the proper versions should be referenced in your bundles.

25.2.1.3. Deploying a Bundle

A bundle is typically installed in one of two ways:

1. Installed as a feature

2. Hot deployed in the `/deploy` directory

The fastest way to deploy a created bundle during development is to copy it to the `/deploy` directory of a running DDF. This directory checks for new bundles and deploys them immediately. According to Karaf documentation, "Karaf supports hot deployment of OSGi bundles by monitoring JAR files inside the `[home]/deploy` directory. Each time a JAR is copied in this folder, it is installed inside the runtime. It can be updated or deleted and changes are handled automatically. In addition, Karaf also supports exploded bundles and custom deployers (Blueprint and Spring DM are included by default)." Once deployed, the bundle should come up in the Active state, if all of the dependencies were properly met. When this occurs, the service is available to be used.

25.2.1.4. Verifying Bundle State

To verify if a bundle is deployed and running, go to the running command console and view the status.

- Execute the `list` command.
- If the name of the bundle is known, the `list` command can be piped to the `grep` command to quickly find the bundle.

The example below shows how to verify if a Client is deployed and running.

Verifying with grep

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

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

25.3. High Availability Guidance

Capabilities that need to function in a Highly Available Cluster should have one of the two below properties.

Stateless

Stateless capabilities function in a Highly Available Cluster because no synchronization between DDF nodes is necessary.

Common storage

If a capability must store data or share state with another node, then the data or shared state must be accessible to all nodes in the Highly Available Cluster. For example, the Catalog's storage provider must be accessible to all DDF nodes.

Appendices

Appendix A: Application References

26. Application Reference

Installation and configuration details by application.

26.1. Admin Application Reference

The Admin Application contains components that are integral for the configuration of DDF applications. It contains various services and interfaces that allow administrators control over their systems and enhances administrative capabilities.

26.1.1. Admin Application Prerequisites

None.

26.1.2. Installing the Admin Application

Install the Admin application through the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `admin-app` feature.

26.1.3. Configuring the Admin Application

To configure the Admin Application:

1. Navigate to the Admin Console.
2. Select the **Admin** application.
3. Select the **Configuration** tab.

Table 92. Admin Available Configurations

Name	Property	Description
Admin Configuration Policy	org.codice.ddf.admin.config.policy.AdminConfigPolicy	Admin Configuration Policy configurations.

Name	Property	Description
Admin UI	org.codice.admin.ui.configuration	Admin UI configurations.

Table 93. Admin Configuration Policy

Name	Id	Type	Description	Default Value	Required
Feature and App Permissions	`featurePolicies`	String	When enabled, the desired features or apps are only modifiable and viewable to users with the set attributes. The entry should be the format of: `feature name/app name = "user attribute name=user attribute value"`		false
Configuration Permissions	`servicePolicies`	String	When enabled, the desired service is only modifiable and viewable to users with the set attributes. The entry should be the format of: `configuration ID = "user attribute name=user attribute value"`	null	false

Table 94. Admin UI

Name	Id	Type	Description	Default Value	Required
Enable System Usage message	`systemUsageEnabled`	Boolean	Turns on a system usage message, which is shown when the Admin Application is opened.	false	true
System Usage Message Title	`systemUsageTitle`	String	A title for the system usage message when the application is opened.		true
System Usage Message	`systemUsageMessage`	String	A system usage message to be displayed to the user each time the user opens the application.		true
Show System Usage Message once per session	`systemUsageOncePerSession`	Boolean	With this selected,the system usage message is shown once for each browser session. Uncheck this to have the usage message appear every time the admin page is opened or refreshed.	true	true

Name	Id	Type	Description	Default Value	Required
Ignored Installer Applications	`disabledInstallerApps`	String	Comma delimited list (appName, appName2, ...appNameN) of applications that are disabled in the installer.	admin-app,platform-app	null

26.2. Catalog Application Reference

The Catalog provides a framework for storing, searching, processing, and transforming information.

Clients typically perform create, read, update, and delete (CRUD) operations against the Catalog.

At the core of the Catalog functionality is the Catalog Framework, which routes all requests and responses through the system, invoking additional processing per the system configuration.

26.2.1. Catalog Application Prerequisites

To use the Catalog Application, the following applications/features must be installed:

- Platform

26.2.2. Installing the Catalog Application

Install the Catalog application through the Admin Console.

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

26.2.3. Configuring the Catalog Application

To configure the Catalog Application:

1. Navigate to the Admin Console.
2. Select the **Catalog** application.
3. Select the **Configuration** tab.

Table 95. Catalog Available Configurations

Name	Property	Description
Status Source Poller Runner	org.codice.ddf.catalog.sourcepoller.StatusSourcePollerRunner	Status Source Poller Runner.
Security Audit Plugin	org.codice.ddf.catalog.plugin.security.audit.SecurityAuditPlugin	Security Audit Plugin.
Metacard Attribute Security Policy Plugin	org.codice.ddf.catalog.security.policy.metacard.MetacardAttributeSecurityPolicyPlugin	Metacard Attribute Security Policy Plugin.
Confluence Federated Source	Confluence_Federated_Source	Confluence Federated Source.
Historian	ddf.catalog.history.Historian	Enables versioning of both metacards and content.
Expiration Date Pre-Ingest Plugin	org.codice.ddf.catalog.plugin.expiration.ExpirationDatePlugin	Catalog pre-ingest plugin to set an expiration date on metacards.
Catalog Policy Plugin	org.codice.ddf.catalog.security.CatalogPolicy	Catalog Policy Plugin.
Tika Input Transformer	ddf.catalog.transformer.input.tika.TikaInputTransformer	Tika Input Transformer.
Resource Download Settings	ddf.catalog.resource.download.ReliableResourceDownloadManager	Resource Download configurations.
Catalog Metacard Ingest Network Plugin	org.codice.ddf.catalog.plugin.metacard.MetacardIngestNetworkPlugin	Catalog Metacard Ingest Network Plugin.
Catalog OpenSearch Federated Source	OpenSearchSource	Catalog OpenSearch Federated Source.
Catalog Standard Framework	ddf.catalog.CatalogFrameworkImpl	Catalog Standard Framework configurations.
CSW Connected Source	Csw_Connected_Source	CSW Connected Source.
Content File System Storage Provider	org.codice.ddf.catalog.content.impl.FileSystemStorageProvider	Content File System Storage Provider.
URL Resource Reader	ddf.catalog.resource.impl.URLResourceReader	URL Resource Reader
Xml Query Transformer	ddf.catalog.transformer.xml.XmlResponseQueueTransformer	Xml Response Query Transformer.
Resource URI Policy Plugin	org.codice.ddf.catalog.security.ResourceUriPolicy	Resource URI Policy Plugin.

Name	Property	Description
PDF Input Transformer	ddf.catalog.transformer.input.pdf.PdfInputTransformer	PDF Input Transformer configurations.
Video Thumbnail Plugin	org.codice.ddf.catalog.content.plugin.video.VideoThumbnailPlugin	Video Thumbnail Plugin.
Schematron Validation Services	ddf.services.schematron.SchematronValidationService	Schematron Validation Services configurations.
Catalog Backup Plugin	ddf.catalog.backup.CatalogBackupPlugin	Catalog Backup Plugin configurations.
Catalog Preview	org.codice.ddf.transformer.preview	Allow Preview to be Extracted From Metadata.
Content Directory Monitor	org.codice.ddf.catalog.content.monitor.ContentDirectoryMonitor	Content Directory Monitor configurations.
Metacard Validation Marker Plugin	ddf.catalog.metacard.validation.MetacardValidityMarkerPlugin	Metacard Validation Marker Plugin.
XML Attribute Security Policy Plugin	org.codice.ddf.catalog.security.policy.xml.XmlAttributeSecurityPolicyPlugin	XML Attribute Security Policy Plugin.
Metacard Backup File Storage Provider	Metacard_File_Storage_Route	Enable data backup of metacards using a configurable transformer.
Catalog Federation Strategy	ddf.catalog.federation.impl.CachingFederationStrategy	Catalog Federation Strategy.
Metacard Validation Filter Plugin	ddf.catalog.metacard.validation.MetacardValidityFilterPlugin	Metacard Validation Filter Plugin.
Resource Download Settings	Metacard_S3_Storage_Route	Resource Download Configuration.

Table 96. Status Source Poller Runner

Name	Id	Type	Description	Default Value	Required
Poll Interval (minutes)	`pollIntervalMinutes`	Integer	The interval (in minutes) at which to recheck the availability of all sources. Must be at least 1 minute. **WARNING** There is a maximum delay of 2*`pollIntervalMinutes` for the Source Poller to be updated after the availability of a source changes or a source is created/modified/deleted. Currently the Standard Catalog Framework and the Catalog REST Endpoint use the Source Poller to get source availabilities. The `pollIntervalMinutes` should not be set to value a which results in an unacceptable maximum delay.	1	true

Table 97. Security Audit Plugin

Name	Id	Type	Description	Default Value	Required
Security attributes to audit	`auditAttributes`	String	List of security attributes to audit when modified	security.access-groups,security.access-individuals	true

Table 98. Metacard Attribute Security Policy Plugin

Name	Id	Type	Description	Default Value	Required
Metacard Intersect Attributes:	`intersectMetacardAttributes`	List of rules	Each line item in the configuration is a rule. The format of a rule is the name of a single source attribute, followed by an equals sign, followed by the destination attribute. For example: source_attribute1=destination_attribute. The plugin gathers the source attributes that have a common destination. It takes the combined values of the source attributes and makes them the values of a (new) metacard attribute, the destination attribute. The strategy for combining the values is intersection, which means only the values common to all source attribute are added to the destination attribute. Note: Do not use the same destination attributes in both the Intersect and Union rule sets. The plugin behaves unpredictably.	none	false

Name	Id	Type	Description	Default Value	Required
Metacard Union Attributes:	`unionMetacardAttributes`	List of rules	Each line item in the configuration is a rule. The format of a rule is the name of a single source attribute, followed by an equals sign, followed by the destination attribute. For example: source_attribute1=destination_attribute. The plugin gathers the source attributes that have a common destination. It takes the combined values of the source attributes and makes them the values of a (new) metacard attribute, the destination attribute. The strategy for combining the values is union, which means only all the values of the source attribute are added to the destination attribute (excluding duplicates) Note: Do not use the same destination attributes in both the Intersect and Union rule sets. The plugin behaves unpredictably.	none	false

Table 99. Confluence Federated Source

Name	Property	Type	Description	Default Value	Required
Source Name	`shortname`	String			Yes
Confluence Rest URL	`endpointUrl`	String	The Confluence Rest API endpoint URL. Example: https://{FQDN}:{PORT}/rest/api/content		Yes
Authentication Type	`authenticationType`	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	saml	true
Username	`username`	String	Username for WFS Service. Required if basic authentication type is selected.	null	false

Name	Property	Type	Description	Default Value	Required
Password	password	Password	Password for WFS Service. Required if basic authentication type is selected.	null	false
Include Page Contents In Results	includePageContent	Boolean	Flag indicating if Confluence page contents should be included in the returned results.	false	No
Include Archived Spaces	includeArchivedSpaces	Boolean	Flag indicating if archived confluence spaces should be included in search results.	false	No
Exclude Confluence Spaces	excludeSpaces	Boolean	Flag indicating if the list of Confluence Spaces should be excluded from searches instead of included.	false	No
Confluence Spaces	confluenceSpaces	String cardinality =1000	The confluence spaces to include/exclude from searches. If no spaces are specified, all visible spaces are searched.		No
Attribute Overrides	additionalAttributes	String cardinality =100	Attribute Overrides - Optional: Metacard attribute overrides (Key-Value pairs) that can be set on the results comming from this source. If an attribute is specified here, it overwrites the metacard's attribute that was created from the Confluence source. The format should be 'key=value'. The maximum allowed size of an attribute override is 65,535 bytes. All attributes in the catalog taxonomy tables are injected into all metacards by default and can be overridden.		No
Availability Poll Interval	availabilityPollInterval	Long	Availability polling interval in milliseconds.	60000	No

Table 100. Historian

Name	Id	Type	Description	Default Value	Required
Enable Versioning	`historyEnabled`	Boolean	Enables versioning of both metacards and content.	true	true

Table 101. Expiration Date Pre-Ingest Plugin

Name	Id	Type	Description	Default Value	Required
Overwrite If Empty	`overwriteIfBlank`	Boolean	If this is checked, overwrite all blank expiration dates in metacards. If this is not checked, leave metacards with blank expiration dates as-is.	false	true
Overwrite If Exists	`overwriteIfExists`	Boolean	If this is checked, overwrite all existing non-empty expiration dates in metacards with a new date. If this is not checked, leave metacards with an existing expiration date.	false	true
Offset from Created Date (in days)	`offsetFromCreatedDate`	Integer	A metacard's new expiration date is calculated by adding this value (in days) to its created date.	30	true

Table 102. Catalog Policy Plugin

Name	Id	Type	Description	Default Value	Required
Create Required Attributes	`createPermissions`	String	Roles/attributes required for the create operations. Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Update Required Attributes	`updatePermissions`	String	Roles/attributes required for the update operation. Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true

Name	Id	Type	Description	Default Value	Required
Delete Required Attributes	`deletePermissions`	String	Roles/attributes required for the delete operation. Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Read Required Attributes	`readPermissions`	String	Roles/attributes required for the read operations (query and resource). Example: role=role1,role2	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true

Table 103. Tika Input Transformer

Name	Id	Type	Description	Default Value	Required
Use Resource Title	`useResourceTitleAsTitle`	Boolean	Use the resource's metadata to determine the metacard title. If this is not enabled, the metacard title is the file name.	false	true

Table 104. Resource Download Settings

Name	Property	Type	Description	Default Value	Required
Product Cache Directory	productCacheDirectory	String	Directory where retrieved products are cached for faster, future retrieval. If a directory path is specified with directories that do not exist, Product Download feature attempts to create those directories. Without configuration, the product cache directory is <DDF_HOME>/data/product-cache. If a relative path is provided it must be relative to the <DDF_HOME>. It is recommended to enter an absolute directory path such as /opt/product-cache in Linux or C:\product-cache in Windows.		false
Enable Product Caching	cacheEnabled	Boolean	Check to enable caching of retrieved products.	true	false
Delay (in seconds) between product retrieval retry attempts	delayBetweenRetryAttempts	Integer	The time to wait (in seconds) between attempting to retry retrieving a product.	10	false
Max product retrieval retry attempts	maxRetryAttempts	Integer	The maximum number of attempts to retry retrieving a product.	3	false
Product Retrieval Monitor Period	retrievalMonitorPeriod	Integer	How many seconds to wait and not receive product data before retrying to retrieve a product.	5	false
Always Cache Product	cacheWhenCanceled	Boolean	Check to enable caching of retrieved products even if client cancels the download. Note: this has no effect if product caching is disabled.	false	false

Table 105. Catalog Metacard Ingest Network Plugin

Name	Id	Type	Description	Default Value	Required	Criteria
criteriaKey	String	Specifies the criteria for the test of equality; which value is tested? IP Address? Hostname?	remoteAddr	true	Expected Value	expected Value
String	The value that the criteria must equate to for the attribute overrides to occur.		true	New Attributes	newAttributes	String"

Table 106. Catalog OpenSearch Federated Source

Name	Id	Type	Description	Default Value	Required
Source Name	`shortname`	String	null	DDF-OS	true

Name	Id	Type	Description	Default Value	Required
OpenSearch service URL	endpointUrl	String	The OpenSearch endpoint URL or DDF's OpenSearch endpoint (https://{FQDN}:{PORT}/services/catalog/query)	${org.codice.ddf.system.protocol}${org.codice.ddf.system.hostname}:${org.codice.ddf.system.port}${org.codice.ddf.system.rootContext}/catalog/query	true
Authentication Type	authenticationType	String	Authentication type to use when federating.	saml	true
Username	username	String	Username to use with HTTP Basic Authentication. Required if basic authentication type is selected.		false
Password	password	Password	Password to use with HTTP Basic Authentication. Required if basic authentication type is selected.		false
OAuth Discovery Url	oauthDiscoveryUrl	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	https://localhost:8443/auth/realms/master/.well-known/openid-configuration	false
OAuth Client ID	oauthClientId	String	Client ID registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	ddf-client	false
OAuth Client Secret	oauthClientSecret	String	Client Secret registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	secret	false

Name	Id	Type	Description	Default Value	Required
OAuth Flow	`oauthFlow`	String	The OAuth flow to use when federating. Required if OAuth 2.0 authentication type is selected.	code	false
OpenSearch query parameters	`parameters`	String	Query parameters to use with the OpenSearch connection.	q,src,mr,start,count,mt,dn,lat,lon,radius,bbox,geometry,polygon,dtstart,dtend,dateName,filter,sort	true
Always perform local query	`localQueryOnly`	Boolean	When federating with other DDFs, keep this checked. If checked, this source performs a local query on the remote site (by setting src=local in endpoint URL), as opposed to an enterprise search.	true	true
Convert to BBox	shouldConvertToBBox	Boolean	Converts Polygon and Point-Radius searches to a Bounding Box for compatibility with older interfaces. Generated bounding box is a very rough representation of the input geometry.	true	true
Multi Point-Radius polygon approximation vertices	numMultiPointRadiusVertices	Integer	When performing a multi point-radius search, increase this value for more accurate polygon approximation. Minimum value is 4, maximum value is 32.	32	true
Point radius polygon simplification distance tolerance	distanceTolerance	Integer	The maximum distance (in meters) from the original vertices a reduced vertex may lie on a simplified circular polygon.	1	true
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true

Name	Id	Type	Description	Default Value	Required
Connection Timeout	connectionTimeout	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	receiveTimeout	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true
Entry XML Element	markUpSet	String	XML Element from the Response Entry to transform into a Metacard.		false

Table 107. Catalog Standard Framework

Name	Id	Type	Description	Default Value	Required
Enable Fanout Proxy	`fanoutEnabled`	Boolean	When enabled the Framework acts as a proxy, federating requests to all available sources. All requests are executed as federated queries and resource retrievals, allowing the framework to be the sole component exposing the functionality of all of its Federated Sources.	false	true
Enable Notifications	`notificationEnabled`	Boolean	Check to enable notifications.	true	false
Fanout tag blacklist	`fanoutTagBlacklist`	String	Ingest operations with tags in this list are rejected.		true

Table 108. CSW Connected Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source.	CSW	true
CSW URL	`cswUrl`	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec.	null	true
Event Service Address	`eventServiceAddress`	String	DDF Event Service endpoint. Do NOT include .wsdl or ?wsdl.	null	false

Name	Id	Type	Description	Default Value	Required
Register for Events	`registerForEvents`	Boolean	Check to register for events from this connected source.	false	false
Authentication Type	`authenticationType`	String	Authentication type to use when federating.	saml	true
Username	`username`	String	Username for CSW Service. Required if basic authentication type is selected.	null	false
Password	`password`	String	Password for CSW Service. Required if basic authentication type is selected.	null	false
Disable CN Check	`disableCnCheck`	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Force Longitude/ Latitude coordinate order	`isLonLatOrder`	Boolean	Force Longitude/Latitude coordinate order.	false	true
Use posList in LinearRing	`usePosList`	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing.	false	false
Metacard Mappings	`metacardMappings`	String	Mapping of the Metacard Attribute names to their CSW property names. The format should be `'title=dc:title'`.	effective=created, created=dateSubmitted, modified=modified, thumbnail=references, content-type=type, id=identifier, resource-uri=source	false
Poll Interval	`pollInterval`	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true

Name	Id	Type	Description	Default Value	Required
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true
Output Schema	`outputSchema`	String	Output Schema	http://www.opengis.net/cat/csw/2.0.2	true
Query Type Name	`queryTypeName`	String	Qualified Name for the Query Type used in the CSW GetRecords request.	csw:Record	true
Query Type Namespace	`queryTypeNamespace`	String	Namespace prefix for the Query Type used in the CSW GetRecords request.	http://www.opengis.net/cat/csw/2.0.2	true
Force CQL Text as the Query Language	`isCqlForced`	Boolean	Force CQL Text.	false	true
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false

Table 109. Content File System Storage Provider

Name	Id	Type	Description	Default Value	Required
Content Repository File Path	`baseContentDirectory`	String	Specifies the directory to use for the content repository. A shutdown of the server is necessary for this property to take effect. If a filepath is provided with directories that don't exist, File System Provider attempts to create them.	<DDF_HOME>/data/content/store	true

Table 110. URL Resource Reader

Name	Property	Type	Description	Default Value
Follow Server Redirects	`followRedirects`	Boolean	Check the box if you want the Resource Reader to automatically follow server issued redirects (HTTP Response Code 300 series).	true
Root Resource Directories	`rootResourceDirectories`	String	List of root resource directories. A relative path is relative to <DDF_HOME>. Specifies the only directories the URLResourceReader has access to when attempting to download resources linked using file-based URLs.	data/products

Table 111. Xml Query Transformer

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

Table 112. Resource URI Policy Plugin

Name	Id	Type	Description	Default Value	Required
Permit Resource URI on Creation	`createPermissions`	String	Allow users to provide a resource URI when creating a metacard	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true
Permit Resource URI on Update	`updatePermissions`	String	Allow users to provide a resource URI when updating a metacard	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role=guest	true

Table 113. PDF Input Transformer

Name	Id	Type	Description	Default Value	Required
Use PDF Title	`usePdfTitleAsTitle`	Boolean	Use the PDF's metadata to determine the metacard title. If this is not enabled, the metacard title is the file name.	false	true
Maximum text extraction length (bytes)	`previewMaxLength`	Integer	The maximum length of text to be extracted.	30000	true
Maximum xml metadata length (bytes)	`metadataMaxLength`	Integer	The maximum length of xml metadata to be extracted.	5000000	true

Table 114. Video Thumbnail Plugin

Name	Property	Type	Description	Default Value	Required
Maximum video file size to process (Megabytes)	`maxFileSizeMB`	Long	Maximum video file size in Megabytes for which to create a thumbnail. Default is 120 Megabytes. Processing large videos may affect system performance.	120	false

Table 115. Schematron Validation Services

Name	Id	Type	Description	Default Value	Required
Ruleset Name	`id`	String	Give this ruleset a name	null	true
Root Namespace	`namespace`	String	The root namespace of the XML	null	true
Schematron File Names	`schematronFileNames`	String	Names of schematron files (*.sch) against which to validate metadata ingested into the Catalog. Absolute paths or relative paths may be specified. Relative paths are assumed to be relative to `<DDF_HOME>/schematron`.	null	true

Table 116. Catalog Backup Plugin

Name	Id	Type	Description	Default Value	Required
Root backup directory path	`rootBackupDir`	String	Root backup directory for Metacards. A relative path is relative to <DDF_HOME>.	data/backup	true
Subdirectory levels	`subDirLevels`	Integer	Number of subdirectory levels to create. Two characters from the ID are used to name each subdirectory level.	2	true

Table 117. Catalog Preview

Name	Id	Type	Description	Default Value	Required
Preview From Metadata	`previewFromMetadata`	Boolean	Allow Preview to be Extracted From Metadata.	false	true
Element Names to Preview	`previewElements`	String	Specify element names to preview from XML. Takes the text content of the first available element for preview. Note: This list is not used unless Preview From Metadata is enabled.	text,TEXT	true

Table 118. Catalog Content Directory Monitor

Name	Id	Type	Description	Default Value	Required
Directory Path	`monitoredDirectoryPath`	String	"Specifies the directory to be monitored, can be a filesystem path or webdav address (only supported for Monitor in place)"	false	true
Maximum Concurrent Files	`numThreads`	Integer	Specifies the maximum number of concurrent files to be processed within a directory (maximum of 8). If this number exceeds 8, 8 is used in order to preserve system resources. Make sure that your system has enough memory to support the number of concurrent processing threads across all directory monitors.	1	true

Name	Id	Type	Description	Default Value	Required
ReadLock Time Interval	`readLockIntervalMilliseconds`	Integer	Specifies the time to wait (in milliseconds) before acquiring a lock on a file in the monitored directory. This interval is used for sleeping between attempts to acquire the read lock on a file to be ingested. The default value of 100 milliseconds is recommended.	100	true
Processing Mechanism	`processingMechanism`	String	Choose what happens to the content item after it is ingested. Delete removes the original file after storing it in the content store. Move stores the item in the content store, and a copy under ./ingested, then removes the original file. (NOTE: this doubles the amount of disk space used.) Monitor in place indexes the file and serves it from its original location. If in place is used, then the URLResourceReader root resource directories configuration must be updated to allow downloading from the monitored directory (See URL Resource Reader).	in_place	false
Attribute Overrides	`attributeOverrides`	String	Optional: Metacard attribute overrides (Key-Value pairs) that can be set on the content monitor. If an attribute is specified here, it overwrites the metacard's attribute that was created from the content directory. The format should be 'key=value'. The maximum allowed size of an attribute override is 65,535 bytes. All attributes in the catalog taxonomy tables are injected into all metacards by default and can be overridden.	null	false

Table 119. Metacard Validation Marker Plugin

Name	Id	Type	Description	Default Value	Required
Enforced Validators	`enforcedMetacardValidators`	String	ID of Metacard Validator to enforce. Metacards that fail these validators are NOT ingested.	false	Enforce errors
enforceErrors	`Boolean`	Sets whether validation errors are enforced. This prevents ingest if errors are present.	true	true	Enforce warnings

Table 120. XML Attribute Security Policy Plugin

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

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

Table 121. Metacard Backup File Storage Provider

Name	Id	Type	Description	Default Value	Required
Keep Deleted Metacard	`keepDeletedMetacards`	Boolean	Should backups for deleted metacards be kept or removed.	false	true
Metacard Transformer Id	`metacardTransformerId`	String	ID of the metacard transformer to use to serialize metacard for backup.	metacard	true
Backup Invalid Metacards	`keepDeletedMetacards`	Boolean	Keep backups for metacards that fail validation with warnings or errors.	true	true
Metacard Backup Output Providers	`metacardOutputProviderIds`	Comma delimited list of metacard output provider IDs.	Metacard Backup Provider IDs to use for this backup plugin.	fileStorageProvider	true

Table 122. Catalog Federation Strategy

Name	Id	Type	Description	Default Value	Required
Maximum start index	maxStartIndex	Integer	Sets a limit on the number of results this sorted federation strategy can handle from each federated source. A large start index in conjunction with several federated sources could yield a large result set, which the sorted federation strategy has a limited ability to do. The admin can make a rough calculation to decide what maximum start index to use based on the amount of memory in the system, the amount of federated sources, the number of threads, and the expected amount of query results requested ((average # of threads) * (maximum # of federated sources) * (maxStartIndex + maximumQueryResults)) must fit into the allocated memory of the running distribution. This field may be deprecated and removed when sorted federation strategy has the ability to sort a larger amount of results.	50000	true
Expiration Interval	expirationIntervalInMinutes	Long	Interval that Solr Cache checks for expired documents to remove.	10	true
Expiration Age	expirationAgeInMinutes	Long	The number of minutes a document remains in the cache before it expires . Default is 7 days.	10080	true
Query Result Cache Strategy	cacheStrategy	String	Strategy for caching query results. Valid entries are ALL, FEDERATED, and NONE.	ALL	true
Cache Remote Ingests	cacheRemoteIngests	Boolean	Cache remote ingest results	false	true

Table 123. Metacard Validation Filter Plugin

Name	Id	Type	Description	Default Value	Required
Attribute map	attributeMap	String	Mapping of Metacard SECURITY attribute to user attribute. Users with this role always receive metacards with errors and/or warnings.	invalid-state=localhost-data-manager	false
Filter errors	filterErrors	Boolean	Sets whether metacards with validation errors are filtered for users without the configured user attribute.	true	false
Filter warnings	filterWarnings	Boolean	Sets whether metacards with validation warnings are filtered for users without the configured user attribute.	false	false

Table 124. Metacard Backup S3 Storage Provider

Name	Id	Type	Description	Default Value	Required
Keep Deleted Metacard	keepDeletedMetacards	Boolean	Should backups for deleted metacards be kept or removed.	false	true
Metacard Transformer Id	metacardTransformerId	String	ID of the metacard transformer to use to serialize metacard for backup.	metacard	true
Backup Invalid Metacards	keepDeletedMetacards	Boolean	Keep backups for metacards that fail validation with warnings or errors.	true	true
Metacard Tags	backupMetacardTags	String	Backup only metacards with one of the tags specified.	resource	true
S3 Access Key	s3AccessKey	String	The access key to use for S3. Leave blank if on an EC2 host with roles assigned.	""	true
S3 Secret Key	s3SecretKey	Password	The secret key to use for S3. Leave blank if on an EC2 host with roles assigned.		true
S3 Bucket	s3Bucket	String	The S3 Bucket in which to store the backed up metacard data.		true
S3 Endpoint	s3Endpoint	String	The endpoint for the region in which the bucket is located.		true

Name	Id	Type	Description	Default Value	Required
Object Template	`objectTemplate`	String	Template specifying the S3 object key for the metacard data. The template uses handlebars syntax. Use [] to reference dotted attributes, as in {{[attribute.name]}}. If you wish to include date, you would use {{dateFormat created yyyy-MM-dd}}	data/backup/metacard/{{substring id 0 3}}/{{substring id 3 6}}/{Metacard_S3_Storage_Route}.xml	true

26.3. Platform Application Reference

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

- Karaf
- CXF
- Camel

A Command Scheduler is also included as part of the Platform application to allow users to schedule Command Line Shell Commands.

26.3.1. Platform Application Prerequisites

None.

26.3.2. Installing Platform

Install the Platform application through the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `platform-app` feature.

26.3.3. Configuring the Platform Application

To configure the Platform Application:

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

Table 125. Platform Available Configurations

Name	Property	Description
MIME Custom Types	DDF_Custom_Mime_Type_Resolver	DDF Custom Mime Types.
Logging Service	org.codice.ddf.platform.logging.LoggingService	Logging Service configurations.
Landing Page	org.codice.ddf.distribution.landingpage.properties	Starting page for users to interact with DDF.
Platform Command Scheduler	ddf.platform.scheduler.Command	Platform Command Scheduler.
Platform UI	ddf.platform.ui.config	Platform UI configurations.
Email Service	org.codice.ddf.platform.email.impl.SmtpClientImpl	Email Service configurations.
HTTP Response Security	org.codice.ddf.security.response.filter.ResponseHeaderConfig	HTTP Response Security response configurations.

Table 126. MIME Custom Types

Name	Id	Type	Description	Default Value	Required
Resolver Name	`name`	String	null	DDF Custom Resolver	false
Priority	`priority`	Integer	null	10	true
File Extensions to Mime Types	`customMimeTypes`	String	List of key/value pairs where key is the file extension and value is the mime type, for example, `nitf=image/nitf`	null	true

Table 127. Logging Service

Name	Id	Type	Description	Default Value	Required
Max Log Events	`maxLogEvents`	Integer	The maximum number of log events stored for display in the Admin Console. This must be greater than 0 and must not exceed 5000.	500	true

Table 128. Landing Page

Name	Id	Type	Description	Default Value	Required
Description	`description`	String	Specifies the description to display on the landing page.	As a common data layer, DDF provides secure enterprise-wide data access for both users and systems.	true
Phone Number	`phone`	String	Specifies the phone number to display on the landing page.		true
Email Address	`email`	String	Specifies the email address to display on the landing page.		true
External Web Site	`externalUrl`	String	Specifies the external web site URL to display on the landing page.		true
Announcements	`announcements`	String	Announcements to display on the landing page.	null	true
Branding Background	`background`	String	Specifies the landing page background color. Use html css colors or #rrggbb.		true
Branding Foreground	`foreground`	String	Specifies the landing page foreground color. Use html css colors or #rrggbb.		true
Branding Logo	`logo`	String	Specifies the landing page logo. Use a base64 encoded image.		true

Name	Id	Type	Description	Default Value	Required
Additional Links	`links`	String	Additional links to be displayed on the landing page. Use the format <text>,<link> (such as `example,http://www.example.com`). Empty entries are ignored.		yes

Table 129. Platform Command Scheduler

Name	Property	Type	Description	Default Value	Required
Command	`command`	String	Shell command to be used within the container. For example, log:set DEBUG">		true
Interval	`intervalString`	String	The Interval String for each execution. Based on the Interval Type, this can be either a Cron String or a Second Interval. (e.x. '0 0 0 1/1 * ? *' or '12')		true
Interval Type	`intervalType`	String	Interval Type	cronString	true

Table 130. Platform UI Configuration

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

Name	Id	Type	Description	Default Value	Required
Show System Usage Message once per session	`systemUsageOncePerSession`	Boolean	With this selected, the system usage message is shown once for each browser session. Uncheck this to have the usage message appear every time the search window is opened or refreshed.	true	true
Header	`header`	String	Specifies the header text to be rendered on all pages.		false
Footer	`footer`	String	Specifies the footer text to be rendered on all pages.		false
Text Color	`color`	String	Specifies the Text Color of the Header and Footer. Use html css colors or `#rrggbb`.		false
Background Color	`background`	String	Specifies the Background Color of the Header and Footer. Use html css colors or `#rrggbb`.		false
Session Timeout	`timeout`	Integer	Specifies the length of inactivity (in minutes) that causes a user to be logged out automatically. This value must be 2 minutes or greater, as users are warned when only 1 minute remains. If a value of less than 2 minutes is used, the timeout is set to the default time of 15 minutes.	15	true

Table 131. Email Service

Name	Property	Type	Description	Default Value	Required
Host	`hostName`	String	Mail server hostname (must be resolvable by DNS) or IP address.		Yes
Port	`portNumber`	Integer	Mail server port number.	25	Yes
User Name	`userName`	String	Mail server user name used only for authenticated connections over TLS.		No
Password	`password`	Password	Mail server password used only for authenticated connections over TLS.		No

Table 132. HTTP Response Security

Name	Id	Type	Description	Default Value	Required
Content Security Policy	`xContentSecurityPolicy`	String	Instructions for the client browser detailing which location and/or which type of resources may be loaded.		true
X-Frame-Options	`xFrameOptions`	String	The X-Frame-Options HTTP response header can be used to indicate whether or not a browser may render a page in a frame, iframe or object.		true
X-XSS-Protection	`xXssProtection`	String	The HTTP X-XSS-Protection response header is a feature that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks.		true

26.4. Security Application Reference

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

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

Features Included in Security

- Security Core
- Security Encryption
- Security PEP
- Security PDP

26.4.1. Security Prerequisites

To use the Security application, the following applications/features must be installed:

- Platform

26.4.2. Installing Security

Install the Security application through the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `security-app` feature.

26.4.3. Configuring the Security Application

To configure the Security Application:

1. Navigate to the Admin Console.
2. Select the **Security** application.
3. Select the **Configuration** tab.

Table 133. Security Available Configurations

Name	Property	Description
Security STS LDAP and Roles Claims Handler	Claims_Handler_Manager	STS Ldap and Roles Claims Handler Configuration.
Guest Claims Configuration	ddf.security.guest.realm	Guest Claims configurations.
Security STS Server	ddf.security.sts	STS configurations.
IdP Client	org.codice.ddf.security.idp.client.IdpMetadata	IdP Client configurations.
SAML Handler	org.codice.ddf.security.idp.client.IdpHandler	IdP Handler configurations.
File Based Claims Handler	org.codice.ddf.security.sts.claims.property.PropertyFileClaimsHandler	File Based Claims Handler.
Web Context Policy Manager	org.codice.ddf.security.policy.context.impl.PolicyManager	Web Context Security Policies.
Security SOAP Guest Interceptor	org.codice.ddf.security.interceptor.GuestInterceptor	Security SOAP Guest Interceptor.
Security STS Client	ddf.security.sts.client.configuration	STS Client configurations.
SAML NameID Policy	ddf.security.service.SecurityManager	SAML NameID Policy.
Security AuthZ Realm	ddf.security.pdp.realm.AuthzRealm	AuthZ Security configurations.
OIDC Handler	org.codice.ddf.security.handler.oidc.OidcHandler	OIDC Handler configurations.

Name	Property	Description
Logout Page	org.codice.ddf.security.idp.client.LogoutRequestService	Logout Page configurations.
Security STS PKI Token Validator	org.codice.ddf.security.validator.pki	STS PKI Token Validator configurations.
Session	org.codice.ddf.security.filter.login.Session	Session configurations.

Table 134. Security STS LDAP and Roles Claims Handler

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

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

Table 135. Guest Claims Configuration

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

Table 136. Security STS Server

Name	Id	Type	Description	Default Value	Required
SAML Assertion Lifetime	lifetime	Long	Set the number of seconds that an issued SAML assertion is valid.	1800	true
Token Issuer	issuer	String	The name of the server issuing tokens. Generally this is unique identifier of this IdP.	https://${org.codice.ddf.system.hostname}:${org.codice.ddf.system.httpsPort}${org.codice.ddf.system.rootContext}/idp/login	true
Signature Username	signatureUsername	String	Alias of the private key in the STS Server's keystore used to sign messages.	${org.codice.ddf.system.hostname}	true

Name	Id	Type	Description	Default Value	Required
Encryption Username	`encryptionUsername`	String	Alias of the private key in the STS Server's keystore used to encrypt messages.	${org.codice.ddf.system.hostname}	true

Table 137. IdP Client

Name	Id	Type	Description	Default Value
IdP Metadata	`metadata`	String	Refer to metadata by HTTPS URL (https://), file URL (file:), or an XML block(<md:EntityDescriptor>... </md:EntityDescriptor>).	https://${org.codice.ddf.system.hostname}:${org.codice.ddf.system.httpsPort}/services/idp/login/metadata
Perform User-Agent Check	`userAgentCheck`	Boolean	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.	true

Table 138. SAML Handler

Name	Id	Type	Description	Default Value
Authentication Context Class	`authContextClasses`	String	Authentication Context Classes that are considered acceptable means of authentication by the SAML handler.	urn:oasis:names:tc:SAML:2.0:ac:classes:Password,urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport,urn:oasis:names:tc:SAML:2.0:ac:classes:X509,urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI,urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI,urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI,urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient

Table 139. File Based Claims Handler

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

Name	Id	Type	Description	Default Value	Required
ID Claim Type	idClaimType	String	ID claim URI.	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier	true
User Role File	propertyFileLocation	String	Location of the file which maps roles to users.	etc/users.properties	true
User Attribute File	usersAttributesFileLocation	String	Location of the file which maps attributes to users.	etc/users.attributes	true

Table 140. Web Context Policy Manager

Name	Id	Type	Description	Default Value	Required
Context Traversal Depth	traversalDepth	Integer	Depth to which paths are traversed. Any value greater than 500 is set to 500.	20	true
Allow Guest Access	guestAccess	Boolean	Allow guest access to all web contexts. Required attributes can be used to restrict access to contexts from guest users.	true	true
Allow Session Storage	sessionAccess	Boolean	Allow for session cookies to be used. Note that the SAML and OIDC authentication types require session storage to be enabled.	true	true
Authentication Types for Web Pages	webAuthenticationTypes	String	List of authentication types required for all web pages (these are all context paths except /services). List of valid authentication types are: BASIC, PKI, SAML, and OIDC. Example: AUTH1	AUTH2	AUTH3
PKI	BASIC	true	Authentication Types for Endpoints	endpointAuthenticationTypes	String

Name	Id	Type	Description	Default Value	Required
List of authentication types required for all endpoints (these are context paths that start with /services). List of valid authentication types are: BASIC, PKI, SAML, and OIDC. Example: AUTH1	AUTH2	AUTH3	PKI	BASIC	true

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

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

Table 141. Security SOAP Guest Interceptor

Name	Id	Type	Description	Default Value	Required
Deny Guest Access	guestAccessDenied	Boolean	If set to true, no guest access is allowed via this guest interceptor. If set to false, this interceptor generates guest tokens for incoming requests that lack a WS-Security header.	false	false

Table 142. Security STS Client

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

Name	Id	Type	Description	Default Value	Required
STS Endpoint Name	endpointName	String	STS Endpoint Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}STS_Port	false
STS Service Name	serviceName	String	STS Service Name.	{http://docs.oasis-open.org/ws-sx/ws-trust/200512/}SecurityTokenService	false
Signature Properties	signatureProperties	String	Path to Signature crypto properties. This path can be part of the classpath, relative to <DDF_HOME>, or an absolute path on the system.	etc/ws-security/server/signature.properties	true
Encryption Properties	encryptionProperties	String	Path to Encryption crypto properties file. This path can be part of the classpath, relative to <DDF_HOME>, or an absolute path on the system.	etc/ws-security/server/encryption.properties	true
STS Properties	tokenProperties	String	Path to STS crypto properties file. This path can be part of the classpath, relative to <DDF_HOME>, or an absolute path on the system.	etc/ws-security/server/signature.properties	true

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

Table 143. SAML NameID Policy

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

Table 144. Security AuthZ Realm

Name	Id	Type	Description	Default Value	Required
Match-All Mappings	`matchAllMappings`	String	List of 'Match-All' subject attribute to Metacard attribute mapping. All values of this metacard key must be present in the corresponding subject key values. Format is subjectAttrName=metacardAttrName.		false
Match-One Mappings	`matchOneMappings`	String	List of 'Match-One' subject attribute to Metacard attribute mapping. One value of this metacard key must be present in the corresponding subject key values. Format is subjectAttrName=metacardAttrName.		false
Environment Attributes	`environmentAttributes`	String	List of environment attributes to pass to the XACML engine. Format is attributeId=attributeValue1,attributeValue2.		false

Table 145. OIDC Handler

Name	Id	Type	Description	Default Value
IdP Type	`idpType`	String	IdP type to use.	Keycloak

Name	Id	Type	Description	Default Value
Client ID	`clientId`	String	Unique ID for the client, this may be provided by the Identity Provider.	ddf-client
Realm/Tenant	`realm`	String	Realm to use for a multi-tenant environment. This is required for Keycloak or Azure.	master
Secret	`secret`	String	This value must match the value set on the Identity Provider.	secret
Discovery URI	`discoveryUri`	String	Discovery URI for fetching OP metadata (http://openid.net/specs/openid-connect-discovery-1_0.html).	http://localhost:8080/auth/realms/master/.well-known/openid-configuration
Base URI	`baseUri`	String	Base URI for IdP. Do not fill out both this and the Discovery URI. Only one is needed depending on the IdP in use.	http://localhost:8080/auth
Logout URI	`logoutUri`	String	URI directing to single logout service of the IdP in use.	http://localhost:8080/auth/realms/master/protocol/openid-connect/logout
Scope	`scope`	String	OIDC scopes.	openid profile email resource.read
Use Nonce	`useNonce`	Boolean	Whether or not to use nonce in JWT.	true
Response Type	`responseType`	String	Response type to use.	code
Mode	`responseMode`	String	Mode. Leave blank if you are unsure of the value to use.	form_post

Table 146. Logout Page

Name	Id	Type	Description	Default Value
Logout Page Time Out	`logOutPageTimeOut`	Long	This is the time limit that the SAML 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."/>	3600000

Table 147. Security STS PKI Token Validator

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

Table 148. Session

Name	Id	Type	Description	Default Value	Required
Session Timeout (in minutes)	`expirationTime`	Integer	Specifies the length of inactivity (in minutes) between client requests before the servlet container invalidates the session (this applies to all client sessions). This value must be 2 minutes or greater, as users are warned when only 1 minute remains. If a value of less than 2 minutes is used, the timeout is set to the default time of 31 minutes. See also Platform UI Config.	31	true

26.5. Solr Catalog Application Reference

DDF uses Solr for data storage, by default.

26.5.1. Solr Catalog Prerequisites

To use the Solr Catalog Application, the following apps/features must be installed:

- Platform
- Catalog

26.5.2. Installing Solr Catalog

Install the Solr Catalog application through the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `catalog-solr-app` feature.

26.5.3. Configuring the Solr Catalog Application

To configure the Solr Catalog Application:

1. Navigate to the Admin Console.
2. Select the **Solr Catalog** application.
3. Select the **Configuration** tab.

Table 149. Solr Catalog Available Configurations

Name	Property	Description
Solr Catalog Provider	ddf.catalog.solr.provider.SolrCatalogProvider	Solr Catalog Provider.

Table 150. Solr Catalog Provider

Name	Property	Type	Description	Default Value	Required
Force Auto Commit	`forceAutoCommit`	Boolean	WARNING: Performance Impact. Only in special cases should auto-commit be forced. Forcing auto-commit makes the search results visible immediately.	false	true

26.6. Spatial Application Reference

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

26.6.1. Offline Gazetteer Service

In the Spatial Application, the `offline-gazetteer` is installed by default. This feature enables you to use an offline source of GeoNames data (as an alternative to the GeoNames Web service enabled by the `webservice-gazetteer` feature) to perform searches via the gazetteer search box in the Search UI.

Installing the `offline-gazetteer-index` feature provides a small set of GeoNames data to use with the offline gazetteer. The GeoNames data is stored as metacards in the core catalog and are tagged with `geonames` and `gazetteer`. This collection of GeoNames metacards can be expanded or updated by using the `gazetteer:update` command.

26.6.1.1. Spatial Gazetteer Console Commands

The `gazetteer` commands provide the ability to interact with the local GeoNames metacard collection in the core catalog. These GeoNames metacards are used by the `offline-gazetteer` feature, which is an optional feature available in this application and is explained above. Note that these commands are only available if the `offline-gazetteer` feature is installed.

Table 151. Gazetteer Command Descriptions

Command	Description
`gazetteer:update`	Adds new gazetteer metacards to the core catalog from a resource. The resource argument can be one of three types: • a local file path to a `.txt`, `.zip`, or `.geo.json` GeoNames data file. If a path to a file ends in .geo.json, it is processed as a geoJSON feature collection and imported as supplementary shape data for GeoNames entries. • a URL to a `.txt` or `.zip` GeoNames data file. GeoJSON URLs are not supported. • a keyword to automatically process a GeoNames file from from http://download.geonames.org/export/dump. Valid keywords include ◦ a country code, which adds the country as GeoNames metacards in the core catalog. The full list of country codes available can be found in http://download.geonames.org/export/dump/countryInfo.txt. ◦ `cities1000`, `cities5000`, and `cities15000`, which adds cities to the index that have at least 1000, 5000, or 15000 people, respectively. ◦ `all`, which downloads all of the current country codes. This process may take some time. The `-c` or `--create` flag can be used to clear out the existing gazetteer metacards before adding new entries.

Command	Description
`build-suggester-index`	Builds the Solr suggester index used for placename autocompletion when using the offline gazetteer. This index is built automatically whenever gazetteer metacards are created, updated, or deleted, but if those builds fail then this command can be used to attempt to build the index again.

26.6.2. Standalone Solr Offline Gazetteer Feature

The Standalone Solr Offline Gazetteer is a reflection of the gazetteer data stored in the catalog but separated from other data. This allows the suggester to be separately rebuilt much quicker than for the entire catalog collection.

26.6.2.1. Installing Standalone Solr Offline Gazetteer

The Standalone Solr Offline Gazetteer is not installed by default but can be installed by running `feature:install catalog-solr-offline-gazetteer`. This installs:

- The plugin that intercepts gazetteer metacard creates/updates/deletes to the catalog collection (and stores them in the standalone solr gazetteer collection)
- The gazetteer query service which registers the Query component that responds to requests from the UI gazetteer search box
- The `offline-solr-gazetteer:removeall` command which deletes all records in the gazetteer collection.
- The `offline-solr-gazetteer:synccatalog` command which syncs with the catalog and updates all records in the gazetteer collection to reflect it (or add them if they are not yet created)

26.6.2.2. Special Note Regarding Installation

If the Spatial Geocoding Offline Catalog bundle and the bundle using the Gazetteer query service (for example, Intrigue) are both already running, you must restart the bundles consuming the service (for example, Intrigue) in order to pick up the correct one.

26.6.2.3. Building the suggester index on Standalone Solr Offline Gazetteer

The suggester index is built when any of the other methods of building the catalog gazetteer are run, along with on initial install. So you can run `gazetteer:build-suggester-index` or `offline-solr-gazetteer:build-suggester-index` to build the suggester.

Another option is to send a build request via a solr client. An example of this is done through the Solr Dashboard after selecting the `gazetteer` collection and clicking `query`. Ensure that the Request-Handler (qt) is set to `/suggest` and then in the 'Extra Parameters' field add `suggest.build=true&suggest.dictionary=suggestPlace&suggest.q=bla`.

26.6.3. Spatial Prerequisites

To use the Spatial Application, the following apps/features must be installed:

- Platform
- Catalog

26.6.4. Installing Spatial

Install the Spatial application through the Admin Console.

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

26.6.5. Configuring the Spatial Application

To configure the Spatial Application:

1. Navigate to the Admin Console.
2. Select the **Spatial** application.
3. Select the **Configuration** tab.

Table 152. Spatial Available Configurations

Name	Property	Description
WFS 2.0.0 Connected Source	Wfs_v2_0_0_Connected_Source	WFS 2.0.0 Connected Source.
CSW Specification Profile Federated Source	Csw_Federated_Source	CSW Specification Profile Federated Source should be used when federating to an external CSW service.
WFS 1.1.0 Federated Source	Wfs_v1_1_0_Federated_Source	WFS 1.1.0 Federated Source.
CSW Federation Profile Source	Csw_Federation_Profile_Source	DDF's full-fidelity CSW Federation Profile. Use this when federating to a DDF-based system.
Metacard to WFS Feature Map	org.codice.ddf.spatial.ogc.wfs.catalog.mapper.MetacardMapper	Metacard to WFS Feature Map.
GMD CSW ISO Federated Source	Gmd_Csw_Federated_Source	CSW Federated Source using the Geographic MetaData (GMD) format (ISO 19115:2003).

Name	Property	Description
GeoCoder Plugin	org.codice.ddf.spatial.geocoding.plugin.GeoCoderPlugin	GeoCoder Plugin.
WFS 2.0.0 Federated Source	Wfs_v2_0_0_Federated_Source	WFS 2.0.0 Federated Source.
CSW Transactional Profile Federated Source	Csw_Transactional_Federated_Source	CSW Federated Source that supports transactions (create, update, delete).
Spatial KML Style Map Entry	org.codice.ddf.spatial.kml.style	Spatial KML Style Map Entry.
Spatial KML Endpoint	org.codice.ddf.spatial.kml.endpoint.KmlEndpoint	Spatial KML Endpoint.

Table 153. WFS 2.0.0 Connected Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source	WFS	true
WFS URL	`wfsUrl`	String	URL to the endpoint implementing the Web Feature Service (WFS) 2.0.0 spec	null	true
Disable CN Check	`disableCnCheck`	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Force Longitude/ Latitude coordinate order	`isLonLatOrder`	Boolean	Force Longitude/Latitude coordinate order	false	true
Disable Sorting	`disableSorting`	Boolean	When selected, the system does not specify sort criteria with the query. This should only be used if the remote source is unable to handle sorting even when the capabilities states 'ImplementsSorting' is supported.	false	true
Authentication Type	`authenticationType`	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	saml	true

Name	Id	Type	Description	Default Value	Required
Username	`username`	String	Username for WFS Service. Required if basic authentication type is selected.	null	false
Password	`password`	Password	Password for WFS Service. Required if basic authentication type is selected.	null	false
Non Queryable Properties	`nonQueryableProperties`	String	Properties listed here are NOT queryable and any attempt to filter on these properties results in an exception.	null	false
Poll Interval	`pollInterval`	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

Table 154. CSW Specification Profile Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source	null	true

Name	Id	Type	Description	Default Value	Required
CSW URL	cswUrl	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec	${org.codice.ddf.external.protocol}${org.codice.ddf.external.hostname}:${org.codice.ddf.external.port}${org.codice.ddf.external.context}${org.codice.ddf.system.rootContext}/csw	true
Event Service Address	eventServiceAddress	String	DDF Event Service endpoint.	${org.codice.ddf.external.protocol}${org.codice.ddf.external.hostname}:${org.codice.ddf.external.port}${org.codice.ddf.external.context}${org.codice.ddf.system.rootContext}/csw/subscription	false
Register for Events	registerForEvents	Boolean	Check to register for events from this source.	false	false
Authentication Type	authenticationType	String	Authentication type to use when federating.	saml	true
Username	username	String	Username for CSW Service. Required if basic authentication type is selected.	null	false

Name	Id	Type	Description	Default Value	Required
Password	password	Password	Password for CSW Service. Required if basic authentication type is selected.	null	false
OAuth Discovery Url	oauthDiscoveryUrl	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	https://localhost:8443/auth/realms/master/.well-known/openid-configuration	false
OAuth Client ID	oauthClientId	String	Client ID registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	ddf-client	false
OAuth Client Secret	oauthClientSecret	String	Client Secret registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	secret	false
OAuth Flow	oauthFlow	String	The OAuth flow to use when federating. Required if OAuth 2.0 authentication type is selected.	code	false
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	coordinateOrder	String	Coordinate order that remote source expects and returns spatial data in	LON_LAT	true
Use posList in LinearRing	usePosList	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	false

Name	Id	Type	Description	Default Value	Required
Metacard Mappings	`metacardMappings`	String	Mapping of the Metacard Attribute names to their CSW property names. The format should be 'title=dc:title'.	effective=created,created=dateSubmitted,modified=modified,thumbnail=references,content-type=type,id=identifier,resource-uri=source	false
Poll Interval	`pollInterval`	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out,in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out,in milliseconds.	60000	true
Output Schema	`outputSchema`	String	Output Schema	http://www.opengis.net/cat/csw/2.0.2	true
Query Type Name	`queryTypeName`	String	Qualified Name for the Query Type used in the CSW GetRecords request	csw:Record	true
Query Type Namespace	`queryTypeNamespace`	String	Namespace for the Query Type used in the CSW GetRecords request	http://www.opengis.net/cat/csw/2.0.2	true
Force CQL Text as the Query Language	`isCqlForced`	Boolean	Force CQL Text	false	true
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false

Name	Id	Type	Description	Default Value	Required
Security Attributes	securityAttributeStrings	String	Security attributes for this source	null	true

Table 155. WFS v1.1.0 Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	id	String	The unique name of the Source	WFS	true
WFS URL	wfsUrl	String	URL to the endpoint implementing the Web Feature Service (WFS) spec	null	true
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	coordinateOrder	String	Coordinate order that remote source expects and returns spatial data in	LAT_LON	true
Forced Feature Type	forcedFeatureType	String	Force only a specific FeatureType to be queried instead of all featureTypes	null	false
Authentication Type	authenticationType	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	saml	true
Username	username	String	Username for WFS Service. Required if basic authentication type is selected.	null	false
Password	password	Password	Password for WFS Service. Required if basic authentication type is selected.	null	false
Non Queryable Properties	nonQueryableProperties	String	Properties listed here are NOT queryable and any attempt to filter on these properties results in an exception.	null	false
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	forceSpatialFilter	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false

Name	Id	Type	Description	Default Value	Required
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true
SRS Name	`srsName`	String	SRS Name to use in outbound GetFeature requests. The SRS Name parameter is used to assert the specific CRS transformation to be applied to the geometries of the features returned in a response document.	EPSG:4326	false

Table 156. CSW Federation Profile Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source	CSW	true
CSW URL	`cswUrl`	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec	${org.codice.ddf.external.protocol}${org.codice.ddf.external.hostname}:${org.codice.ddf.external.port}${org.codice.ddf.external.context}${org.codice.ddf.system.rootContext}/csw	true

Name	Id	Type	Description	Default Value	Required
CSW Event Service Address	eventServiceAddress	String	CSW Event Service endpoint.	${org.codice.ddf.external.protocol}${org.codice.ddf.external.hostname}:${org.codice.ddf.external.port}${org.codice.ddf.external.context}${org.codice.ddf.system.rootContext}/csw/subscription	false
Register for Events	registerForEvents	Boolean	Check to register for events from this connected source.	false	false
Authentication Type	authenticationType	String	Authentication type to use when federating.	saml	true
Username	username	String	Username for CSW Service. Required if basic authentication type is selected.	null	false
Password	password	String	Password for CSW Service. Required if basic authentication type is selected.	null	false
OAuth Discovery Url	oauthDiscoveryUrl	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	https://localhost:8443/auth/realms/master/.well-known/openid-configuration	false

Name	Id	Type	Description	Default Value	Required
OAuth Client ID	`oauthClientId`	String	Client ID registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	ddf-client	false
OAuth Client Secret	`oauthClientSecret`	String	Client Secret registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	secret	false
OAuth Flow	`oauthFlow`	String	The OAuth flow to use when federating. Required if OAuth 2.0 authentication type is selected.	code	false
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out,in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out,in milliseconds.	60000	true

Table 157. Metacard to WFS Feature Map

Name	Id	Type	Description	Default Value	Required
Feature Type	`featureType`	String	Feature Type. Format is {URI}local-name		true
Metacard Title to WFS Feature Property Mapping	`titleMapping`	String	Metacard Title to WFS Feature Property Mapping		false
Metacard Created Date to WFS Feature Property Mapping	`createdDateMapping`	String	Metacard Created Date to WFS Feature Property Mapping		false

Name	Id	Type	Description	Default Value	Required
Metacard Modified Date to WFS Feature Property Mapping	`modifiedDateMapping`	String	Metacard Modified Date to WFS Feature Property Mapping		false
Metacard Effective Date to WFS Feature Property Mapping	`effectiveDateMapping`	String	Metacard Effective Date to WFS Feature Property Mapping		false
Metacard Expiration Date to WFS Feature Property Mapping	`expirationDateMapping`	String	Metacard Expiration Date to WFS Feature Property Mapping		false
Metacard Resource URI to WFS Feature Property Mapping	`resourceUriMapping`	String	Metacard Resource URI to WFS Feature Property Mapping		false
Metacard Resource Size to WFS Feature Property Mapping	`resourceSizeMapping`	String	Metacard Resource Size to WFS Feature Property Mapping		false

Name	Id	Type	Description	Default Value	Required
The Units of the Feature Property that corresponds to the Metacard Resource Size	`dataUnit`	String	The Units of the Feature Property that corresponds to the Metacard Resource Size	B	true
Metacard Thumbnail to WFS Feature Property Mapping	`thumbnailMapping`	String	Metacard Thumbnail to WFS Feature Property Mapping		false
Metacard Geography to WFS Feature Property Mapping	`geographyMapping`	String	Metacard Geography to WFS Feature Property Mapping		false
Temporal Sort By Feature Property	`sortByTemporalFeatureProperty`	String	When Sorting Temporally, Sort By This Feature Property.		false
Relevance Sort By Feature Property	`sortByRelevanceFeatureProperty`	String	When Sorting By Relevance, Sort By This Feature Property.		false
Distance Sort By Feature Property	`sortByDistanceFeatureProperty`	String	When Sorting By Distance, Sort By This Feature Property.		false

Table 158. GMD CSW ISO Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source		true

Name	Id	Type	Description	Default Value	Required
CSW URL	cswUrl	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec		true
Authentication Type	authenticationType	String	Authentication type to use when federating.	saml	true
Username	username	String	Username for CSW Service. Required if basic authentication type is selected.		false
Password	password	Password	Password for CSW Service. Required if basic authentication type is selected.		false
OAuth Discovery Url	oauthDiscoveryUrl	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	https://localhost:8443/auth/realms/master/.well-known/openid-configuration	false
OAuth Client ID	oauthClientId	String	Client ID registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	ddf-client	false
OAuth Client Secret	oauthClientSecret	String	Client Secret registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	secret	false
OAuth Flow	oauthFlow	String	The OAuth flow to use when federating. Required if OAuth 2.0 authentication type is selected.	code	false
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	coordinateOrder	String	Coordinate order expected and returned by remote source	LON_LAT	true

Name	Id	Type	Description	Default Value	Required
Use posList in LinearRing	`usePosList`	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	false

Name	Id	Type	Description	Default Value	Required
Metacard Mappings	`metacardMappings`	String	Mapping of the Metacard Attribute names to their CSW property names. The format should be 'title=dc:title'.	id=apiso:Identifier,effective=apiso:PublicationDate,created=apiso:CreationDate,modified=apiso:RevisionDate,title=apiso:AlternateTitle,AnyText=apiso:AnyText,ows:BoundingBox=apiso:BoundingBox,language=apiso:Language,language=apiso:ResourceLanguage,datatype=apiso:Type,description=apiso:Abstract,contact.point-of-contact-name=apiso:OrganisationName,topic.keyword=apiso:Subject,media.format=apiso:Format,modified=apiso:Modified	false

Name	Id	Type	Description	Default Value	Required
Poll Interval	`pollInterval`	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true
Output Schema	`outputSchema`	String	Output Schema	http://www.isotc211.org/2005/gmd	true
Query Type Name	`queryTypeName`	String	Qualified Name for the Query Type used in the CSW GetRecords request	gmd:MD_Metadata	true
Query Type Namespace	`queryTypeNamespace`	String	Namespace for the Query Type used in the CSW GetRecords request	http://www.isotc211.org/2005/gmd	true
Force CQL Text	`isCqlForced`	Boolean	Force CQL Text as the Query Language	false	true
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Security Attributes	`securityAttributeStrings`	String	Security attributes for this source		true

Table 159. GeoCoder Plugin

Title	Property	Type	Description	Default Value
Radius	`radiusInKm`	Integer	The search radius from a Point in kilometers.	10

Table 160. WFS 2.0.0 Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source	WFS_v2_0_0	true

Name	Id	Type	Description	Default Value	Required
WFS URL	wfsUrl	String	URL to the endpoint implementing the Web Feature Service (WFS) 2.0.0 spec	null	true
Disable CN Check	disableCnCheck	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	coordinateOrder	String	Coordinate order that remote source expects and returns spatial data in	LAT_LON	true
Forced Feature Type	forcedFeatureType	String	Force only a specific FeatureType to be queried instead of all featureTypes	null	false
Disable Sorting	disableSorting	Boolean	When selected, the system does not specify sort criteria with the query. This should only be used if the remote source is unable to handle sorting even when the capabilities states 'ImplementsSorting' is supported.	false	true
Authentication Type	authenticationType	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	saml	true
Username	username	String	Username for the WFS Service. Required if basic authentication type is selected.	null	false
Password	password	Password	Password for the WFS Service. Required if basic authentication type is selected.	null	false
Non Queryable Properties	nonQueryableProperties	String	Properties listed here are NOT queryable and any attempt to filter on these properties results in an exception.	null	false
Poll Interval	pollInterval	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Forced Spatial Filter Type	forceSpatialFilter	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false

Name	Id	Type	Description	Default Value	Required
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true

Table 161. CSW Transactional Profile Federated Source

Name	Id	Type	Description	Default Value	Required
Source ID	`id`	String	The unique name of the Source		true
CSW URL	`cswUrl`	String	URL to the endpoint implementing the Catalogue Service for Web (CSW) spec	${variable-name}org.codice.ddf.system.protocol}${variable-name}org.codice.ddf.system.hostname}:${variable-name}org.codice.ddf.system.port}${variable-name}org.codice.ddf.system.rootContext}/csw	true

Name	Id	Type	Description	Default Value	Required
Event Service Address	`eventServiceAddress`	String	Event Service endpoint.	${variable-name}org.codice.ddf.system.protocol}${variable-name}org.codice.ddf.system.hostname}:${variable-name}org.codice.ddf.system.port}${variable-name}org.codice.ddf.system.rootContext}/csw/subscription	false
Register for Events	`registerForEvents`	Boolean	Check to register for events from this source.	false	false
Authentication Type	`authenticationType`	String	Authentication type to use when federating.	saml	true
Username	`username`	String	Username for CSW Service. Required if basic authentication type is selected.		false
Password	`password`	Password	Password for CSW Service. Required if basic authentication type is selected.		false

Name	Id	Type	Description	Default Value	Required
OAuth Discovery Url	`oauthDiscoveryUrl`	String	The Discovery URL where the metadata of the OAuth Provider protecting the source is hosted. Required if OAuth 2.0 authentication type is selected.	https://localhost:8443/auth/realms/master/.well-known/openid-configuration	false
OAuth Client ID	`oauthClientId`	String	Client ID registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	ddf-client	false
OAuth Client Secret	`oauthClientSecret`	String	Client Secret registered with the OAuth provider. Required if OAuth 2.0 authentication type is selected.	secret	false
OAuth Flow	`oauthFlow`	String	The OAuth flow to use when federating. Required if OAuth 2.0 authentication type is selected.	code	false
Disable CN Check	`disableCnCheck`	Boolean	Disable CN check for the server certificate. This should only be used when testing.	false	true
Coordinate Order	`coordinateOrder`	String	Coordinate order expected and returned by remote source	LON_LAT	true
Use posList in LinearRing	`usePosList`	Boolean	Use a <posList> element rather than a series of <pos> elements when issuing geospatial queries containing a LinearRing	false	false

Name	Id	Type	Description	Default Value	Required
Metacard Mappings	`metacardMappings`	String	Mapping of the Metacard Attribute names to their CSW property names. The format should be 'title=dc:title'.	effective=created,created=dateSubmitted,modified=modified,thumbnail=references,content-type=type,id=identifier,resource-uri=source	false
Poll Interval	`pollInterval`	Integer	Poll Interval to Check if the Source is available (in minutes - minimum 1).	5	true
Connection Timeout	`connectionTimeout`	Integer	Amount of time to attempt to establish a connection before timing out, in milliseconds.	30000	true
Receive Timeout	`receiveTimeout`	Integer	Amount of time to wait for a response before timing out, in milliseconds.	60000	true
Output Schema	`outputSchema`	String	Output Schema	urn:catalog:metacard	true
Query Type Name	`queryTypeName`	String	Qualified Name for the Query Type used in the CSW GetRecords request	csw:Record	true
Query Type Namespace	`queryTypeNamespace`	String	Namespace for the Query Type used in the CSW GetRecords request	http://www.opengis.net/cat/csw/2.0.2	true
Force CQL Text	`isCqlForced`	Boolean	Force CQL Text as the Query Language	false	true
Forced Spatial Filter Type	`forceSpatialFilter`	String	Force only the selected Spatial Filter Type as the only available Spatial Filter.	NO_FILTER	false
Security Attributes	`securityAttributeStrings`	String	Security attributes for this source		true

Table 162. Spatial KML Style Map Entry

Name	Id	Type	Description	Default Value	Required
Attribute Name	attributeName	String	The name of the Metacard Attribute to match against. For example, title, metadata-content-type, etc	null	true
Attribute Value	attributeValue	String	The value of the Metacard Attribute.	null	true
Style URL	styleUrl	String	The full qualified URL to the KML Style, for example, http://example.com/styles#myStyle	null	true

Table 163. Spatial KML Endpoint

Name	Id	Type	Description	Default Value	Required
Style Document	styleUrl	String	KML Document containing custom styling. This is served up by the KmlEndpoint. (for example, file:///path/to/kml/style/doc.kml)		false
Icons Location	iconLoc	String	Location of icons for the KML endpoint		false
Description	description	String	Description of this NetworkLink. Enter a short description of what this NetworkLink provides.		false
Web Site	webSite	String	URL of the web site to be displayed in the description.		false
Logo	logo	String	URL to the logo to be displayed in the description.		false
Visible By Default	visibleByDefault	Boolean	Check if the source NetworkLinks should be visible by default.	false	false
Max Number of Results	maxResults	Integer	The maximum number of results that should be returned from each layer.	100	false

26.7. Search UI Application Reference

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

26.7.1. Search UI Prerequisites

To use the Search UI application, the following applications/features must be installed:

- Platform
- Catalog

26.7.2. Installing Search UI

Install the Search UI application through the Admin Console.

1. Navigate to the **Admin Console**.
2. Select the **System** tab.
3. Select the **Features** tab.
4. Install the `search-ui-app` feature.

26.7.3. Configuring the Search UI Application

To configure the Search UI Application:

1. Navigate to the Admin Console.
2. Select the **Search UI** application.
3. Select the **Configuration** tab.

Table 164. Search UI Available Configurations

Name	Property	Description
Search UI Redirect	org.codice.ddf.ui.searchui.filter.RedirectServlet	Search UI redirect.
Facet Attribute Whitelist	org.codice.ddf.catalog.plugin.facetattributeaccess.facetwhitelist	Facet Attribute Whitelist
Catalog UI Search Workspace Security	org.codice.ddf.catalog.ui.security	Catalog UI Search Workspace Security.
Catalog UI Search Workspace Service	org.codice.ddf.catalog.ui.query.monitor.impl.WorkspaceServiceImpl	Catalog UI Search Workspace Service.
Catalog UI Search Workspace Query Monitor	org.codice.ddf.catalog.ui.query.monitor.impl.WorkspaceQueryService	Catalog UI Search Workspace Query Monitor.
Email Notifier	org.codice.ddf.catalog.ui.query.monitor.email.EmailNotifier	Email Notifier.

Name	Property	Description
Catalog UI Search Transformer Blacklists	org.codice.ddf.catalog.ui.transformer.TransformerDescriptors	Catalog UI Search Transformer Blacklists.

Table 165. Search UI Redirect

Name	Id	Type	Description	Default Value	Required
Redirect URI	`defaultUri`	String	Specifies the redirect URI to use when accessing the /search URI.	`${org.codice.ddf.external.context}/search/simple/`	true

Table 166. Facet Attribute Whitelist

Name	Id	Type	Description	Default Value	Required
Facet Attribute Whitelist	`facetAttributeWhitelist`	String	Attributes that can be faceted against through the catalog framework. Caution: Suggestion values are not protected by any security. Only choose attributes whose values are safe for all users to view.		false

Table 167. Catalog UI Search Workspace Security

Name	Id	Type	Description	Default Value	Required
System User Attribute	`systemUserAttribute`	String	The name of the attribute to determine the system user.	`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role`	true
System User Attribute Value	`systemUserAttributeValue`	String	The value of the attribute to determine the system user.	`system-user`	true

Table 168. Catalog UI Search Workspace Service

Name	Id	Type	Description	Default Value	Required
Maximum Subscriptions	`maxSubscriptions`	Integer	Specifies the maximum number of workspace subscriptions that may be queried for email notifications.	`100`	true

Table 169. Catalog UI Search Workspace Query Monitor

Name	Id	Type	Description	Default Value	Required
Query Timeout	`queryTimeoutMinutes`	Long	Set the number of minutes to wait for query to complete.	`5`	true
Notification Time Interval	`queryTimeInterval`	Integer	Set the Relative Time Search (past X minutes up to 24 hours). Note: This queries for results from the interval to the time the query is sent out.	`1440`	true

Table 170. Catalog UI Search Email Notifier

Name	Id	Type	Description	Default Value	Required
Subject	`subjectTemplate`	String	Set the subject line template.	`Workspace '%[attribute=title]' notification`	true
Body	`bodyTemplate`	String	Set the body template.	`The workspace '%[attribute=title]' contains up to %[hitCount] results. Log in to see results https://{FQDN}:{PORT}/search/catalog/#workspaces/%attribute=id.`	true
From Address	`fromEmail`	String	Set the 'from' email address.	`donotreply@example.com`	true

Table 171. Catalog UI Search Transformer Blacklists

Name	Id	Type	Description	Default Value	Required
Metacard Transformer Blacklist	`blackListedMetacardTransformerIds`	String	The IDs of all Metacard Transformers services that do not show up as export actions in the UI. Every ID in this set removes that transformer as an export option in the UI.	`[]`	false
Query Response Transformer Blacklist	`blackListedQueryResponseTransformerIds`	String	The IDs of all Query Response Transformers services that do not show up as export actions in the UI. Every ID in this set removes that transformer as an export option in the UI.	`[zipCompression]`	false

Appendix B: Application Whitelists

Within each DDF application, certain packages are exported for use by third parties.

B.1. Packages Removed From Whitelist

In the transition of the whitelist from the ambiguous package listing to the new class listing several errors were found. The packages originally listed that were removed either did not exist, contained experimental interfaces, or contained only internal implementations and should have never been included in the whitelist. The following is a list of packages that were listed in error and have been removed from the whitelist.

NOTE	None of the packages in this list have been removed from the distribution. They may however be changed or removed in the future.

Admin

- `org.codice.ddf.ui.admin.api.plugin`
- `org.codice.ddf.admin.configuration.plugin`

Catalog

- `org.codice.ddf.admin.configuration.plugin`
- `ddf.catalog.data.metacardtype`

- ddf.catalog.federation.impl
- ddf.catalog.plugin.groomer
- ddf.catalog.pubsub
- ddf.catalog.pubsub.tracker
- ddf.catalog.resource.data
- ddf.catalog.resource.impl
- ddf.catalog.resourceretriever
- ddf.catalog.transformer.metacard.geojson
- ddf.common
- org.codice.ddf.endpoints
- org.codice.ddf.endpoints.rest
- org.codice.ddf.endpoints.rest.action
- org.codice.ddf.opensearch.query
- org.codice.ddf.opensearch.query.filter

Platform

- org.codice.ddf.configuration.admin
- org.codice.ddf.configuration.migration
- org.codice.ddf.configuration.persistence
- org.codice.ddf.configuration.persistence.felix
- org.codice.ddf.configuration.status
- org.codice.ddf.parser
- org.codice.ddf.parser.xml
- org.codice.ddf.platform.error.handler
- org.codice.ddf.platform.util

Security

- ddf.security.assertion.impl
- ddf.security.common.audit
- ddf.security.http.impl
- ddf.security.impl
- ddf.security.pdp.realm
- ddf.security.permission

- ddf.security.principal
- ddf.security.realm.sts
- ddf.security.samlp.impl
- ddf.security.service.impl
- ddf.security.settings
- ddf.security.soap.impl
- ddf.security.sts
- ddf.security.ws.policy.impl
- org.codice.ddf.security.certificate.generator
- org.codice.ddf.security.certificate.keystore.editor
- org.codice.ddf.security.common
- org.codice.ddf.security.filter.authorization
- org.codice.ddf.security.filter.login
- org.codice.ddf.security.filter.websso
- org.codice.ddf.security.handler.basic
- org.codice.ddf.security.handler.guest.configuration
- org.codice.ddf.security.handler.guest
- org.codice.ddf.security.handler.pki
- org.codice.ddf.security.handler.saml
- org.codice.ddf.security.interceptor
- org.codice.ddf.security.interceptor
- org.codice.ddf.security.policy.context.impl
- org.codice.ddf.security.servlet.logout
- org.codice.ddf.security.validator.username

Spatial

- org.codice.ddf.spatial.geocoder
- org.codice.ddf.spatial.geocoder.geonames
- org.codice.ddf.spatial.geocoding
- org.codice.ddf.spatial.geocoding.context
- org.codice.ddf.spatial.kml.endpoint
- org.codice.ddf.spatial.ogc.catalog.resource.impl

B.2. Catalog Whitelist

The following classes have been exported by the Catalog application and are approved for use by third parties:

In package `ddf.catalog`

- `CatalogFramework`
- `Constants`

In package `ddf.catalog.cache`

- `ResourceCacheInterface` Deprecated

In package `ddf.catalog.data`

- `Attribute`
- `AttributeDescriptor`
- `AttributeType`
- `BinaryContent`
- `ContentType`
- `Metacard`
- `MetacardCreationException`
- `MetacardType`
- `MetacardTypeUnregistrationException`
- `Result`

In package `ddf.catalog.event`

- `DeliveryException`
- `DeliveryMethod`
- `EventException`
- `EventProcessor`
- `InvalidSubscriptionException`
- `Subscriber`
- `Subscription`
- `SubscriptionExistsException`

- SubscriptionNotFoundException

In package ddf.catalog.federation

- Federatable
- FederationException
- FederationStrategy

In package ddf.catalog.filter

- AttributeBuilder
- BufferedSpatialExpressionBuilder
- ContextualExpressionBuilder
- EqualityExpressionBuilder
- ExpressionBuilder
- FilterAdapter
- FilterBuilder
- FilterDelegate
- NumericalExpressionBuilder
- NumericalRangeExpressionBuilder
- SpatialExpressionBuilder
- TemporalInstantExpressionBuilder
- TemporalRangeExpressionBuilder
- XPathBasicBuilder
- XPathBuilder

In package ddf.catalog.filter.delegate

- CopyFilterDelegate
- FilterToTextDelegate

In package ddf.catalog.operation

- CreateRequest
- CreateResponse
- DeleteRequest
- DeleteResponse
- Operation

- OperationTransaction
- Pingable
- ProcessingDetails
- Query
- QueryRequest
- QueryResponse
- Request
- ResourceRequest
- ResourceResponse
- Response
- SourceInfoRequest
- SourceInfoResponse
- SourceProcessingDetails
- SourceResponse
- Update
- UpdateRequest
- UpdateResponse

In package ddf.catalog.plugin

- AccessPlugin
- PluginExecutionException
- PolicyPlugin
- PolicyResponse
- PostFederatedQueryPlugin
- PostIngestPlugin
- PostQueryPlugin
- PostResourcePlugin
- PreDeliveryPlugin
- PreFederatedQueryPlugin
- PreIngestPlugin
- PreQueryPlugin
- PreResourcePlugin
- PreSubscriptionPlugin

- StopProcessingException

In package ddf.catalog.resource

- DataUsageLimitExceededException
- Resource
- ResourceNotFoundException
- ResourceNotSupportedException
- ResourceReader
- ResourceWriter

In package ddf.catalog.service

- ConfiguredService

In package ddf.catalog.source

- CatalogProvider
- ConnectedSource
- FederatedSource
- IngestException
- InternalIngestException
- RemoteSource
- Source
- SourceDescriptor
- SourceMonitor
- SourceUnavailableException
- UnsupportedQueryException

In package ddf.catalog.transform

- CatalogTransformerException
- InputCollectionTransformer
- InputTransformer
- MetacardTransformer
- QueryResponseTransformer

In package ddf.catalog.transformer.api

- MetacardMarshaller

- `PrintWriter`
- `PrintWriterProvider`

In package `ddf.catalog.util`

- `Describable` Deprecated
- `Maskable`

In package `ddf.catalog.validation`

- `MetacardValidator`
- `ValidationException`

In package `ddf.geo.formatter`

- `CompositeGeometry`
- `GeometryCollection`
- `LineString`
- `MultiLineString`
- `MultiPoint`
- `MultiPolygon`
- `Point`
- `Polygon`

In package `ddf.util`

- `InetAddressUtil`
- `NamespaceMapImpl`
- `NamespaceResolver`
- `WktStandard`
- `XPathCache`
- `XPathHelper`
- `XSLTUtil`

B.3. Platform Whitelist

The following classes have been exported by the Platform application and are approved for use by third parties:

In package `ddf.action`

- `Action`
- `ActionProvider`
- `ActionRegistry`

In package `org.codice.ddf.branding`

- `BrandingPlugin`
- `BrandingRegistry`

In package `org.codice.ddf.configuration`

- `ConfigurationWatcher` Deprecated

B.4. Security Whitelist

The following classes have been exported by the Security application and are approved for use by third parties:

In package `ddf.security`

- `SecurityConstants`
- `Subject`

In package `ddf.security.assertion`

- `SecurityAssertion`

In package `ddf.security.common.util`

- `Security` Deprecated
- `SecurityProperties`
- `ServiceComparator`
- `SortedServiceList` Deprecated

In package `ddf.security.encryption`

- `EncryptionService`

In package `ddf.security.expansion`

- `Expansion`

In package `ddf.security.http`

- `SessionFactory`

In package `ddf.security.service`

- `SecurityManager`
- `SecurityServiceException`
- `TokenRequestHandler`

In package `ddf.security.sts.client.configuration`

- `STSClientConfiguration`

In package `ddf.security.ws.policy`

- `AbstractOverrideInterceptor`
- `PolicyLoader`

In package `org.codice.ddf.security.handler.api`

- `AuthenticationHandler`

In package `org.codice.ddf.security.policy.context.attributes`

- `ContextAttributeMapping`

In package `org.codice.ddf.security.policy.context`

- `ContextPolicy`
- `ContextPolicyManager`

B.5. Solr Catalog Whitelist

The following classes have been exported by the Solr Catalog application and are approved for use by third parties:

None.

B.6. Search UI Whitelist

The following classes have been exported by the Search UI application and are approved for use by

third parties:

None.

27. DDF Dependency List

This list of DDF dependencies is automatically generated:

27.1. DDF 2.30.1 Dependency List

- ca.juliusdavies:not-yet-commons-ssl:jar:0.3.11
- ch.qos.logback:logback-access:jar:1.2.3
- ch.qos.logback:logback-classic:jar:1.2.3
- ch.qos.logback:logback-core:jar:1.2.3
- com.connexta.arbitro:arbitro-core:jar:1.0.0
- com.fasterxml.jackson.core:jackson-annotations:jar:2.12.5
- com.fasterxml.jackson.core:jackson-core:jar:2.12.5
- com.fasterxml.jackson.core:jackson-databind:jar:2.12.5
- com.fasterxml.woodstox:woodstox-core:jar:6.2.6
- com.github.drapostolos:type-parser:jar:0.5.0
- com.github.jai-imageio:jai-imageio-core:jar:1.4.0
- com.github.jai-imageio:jai-imageio-jpeg2000:jar:1.3.1_CODICE_3
- com.github.jknack:handlebars:jar:2.0.0
- com.github.lookfirst:sardine:jar:5.7
- com.github.luben:zstd-jni:jar:1.4.9-1
- com.github.openjson:openjson:jar:1.0.12
- com.google.code.findbugs:jsr305:jar:3.0.2
- com.google.code.gson:gson:jar:2.8.5
- com.google.crypto.tink:tink:jar:1.4.0
- com.google.guava:guava:jar:29.0-jre
- com.google.http-client:google-http-client:jar:1.22.0
- com.google.protobuf:protobuf-java:jar:3.12.4
- com.googlecode.json-simple:json-simple:jar:1.1.1

- com.googlecode.owasp-java-html-sanitizer:owasp-java-html-sanitizer:jar:20191001.1
- com.hazelcast:hazelcast:jar:3.12.10
- com.jhlabs:filters:jar:2.0.235-1
- com.mchange:c3p0:jar:0.9.5.5
- com.nimbusds:lang-tag:jar:1.4.4
- com.nimbusds:nimbus-jose-jwt:jar:8.14.1
- com.nimbusds:oauth2-oidc-sdk:jar:8.3
- com.rometools:rome-utils:jar:1.9.0
- com.rometools:rome:jar:1.9.0
- com.sparkjava:spark-core:jar:2.9.2
- com.thoughtworks.xstream:xstream:jar:1.4.18
- com.unboundid:unboundid-ldapsdk:jar:4.0.14
- com.xebialabs.restito:restito:jar:0.9.4
- com.zensols.jrtf:tutego.jrtf:jar:0.1.0
- commons-beanutils:commons-beanutils:jar:1.9.4
- commons-codec:commons-codec:jar:1.11
- commons-codec:commons-codec:jar:1.15
- commons-collections:commons-collections:jar:3.2.2
- commons-configuration:commons-configuration:jar:1.10
- commons-digester:commons-digester:jar:1.8.1
- commons-fileupload:commons-fileupload:jar:1.3.3
- commons-io:commons-io:jar:2.11.0
- commons-lang:commons-lang:jar:2.6
- commons-logging:commons-logging:jar:1.2
- commons-net:commons-net:jar:3.5
- commons-validator:commons-validator:jar:1.6
- de.micromata.jak:JavaAPIforKml:jar:2.2.0
- de.micromata.jak:JavaAPIforKml:jar:2.2.1_CODICE_1
- io.github.soc:ogc-tools-gml-jts:jar:1.1.90
- io.micrometer:micrometer-registry-prometheus:jar:1.5.1
- io.rest-assured:rest-assured:jar:3.3.0
- jakarta.activation:jakarta.activation-api:jar:1.2.2

- jakarta.mail:jakarta.mail-api:jar:1.6.5
- jakarta.xml.bind:jakarta.xml.bind-api:jar:2.3.3
- jakarta.xml.soap:jakarta.xml.soap-api:jar:1.4.2
- javax.activation:activation:jar:1.1.1
- javax.annotation:javax.annotation-api:jar:1.2
- javax.cache:cache-api:jar:1.1.1
- javax.inject:javax.inject:jar:1
- javax.measure:unit-api:jar:1.0
- javax.servlet:javax.servlet-api:jar:3.1.0
- javax.servlet:servlet-api:jar:2.5
- javax.validation:validation-api:jar:1.1.0.Final
- javax.ws.rs:javax.ws.rs-api:jar:2.1
- javax.xml.bind:jaxb-api:jar:2.2.11
- javax.xml.stream:stax-api:jar:1.0-2
- joda-time:joda-time:jar:2.10.6
- junit:junit:jar:4.13.2
- net.bytebuddy:byte-buddy-agent:jar:1.10.18
- net.bytebuddy:byte-buddy:jar:1.10.18
- net.jodah:failsafe:jar:0.9.3
- net.jodah:failsafe:jar:0.9.5
- net.jodah:failsafe:jar:1.0.0
- net.minidev:accessors-smart:jar:2.4.7
- net.minidev:asm:jar:1.0.2
- net.minidev:json-smart:jar:2.3
- net.minidev:json-smart:jar:2.4.7
- net.sf.jwordnet:jwnl:jar:1.3.3
- net.sf.saxon:Saxon-HE:jar:9.6.0-4
- org.antlr:antlr4-runtime:jar:4.3
- org.apache.abdera:abdera-extensions-geo:jar:1.1.3
- org.apache.abdera:abdera-extensions-opensearch:jar:1.1.3
- org.apache.ant:ant-launcher:jar:1.9.7
- org.apache.ant:ant:jar:1.9.7

- org.apache.aries.jmx:org.apache.aries.jmx.api:jar:1.1.5
- org.apache.aries.jmx:org.apache.aries.jmx.core:jar:1.1.8
- org.apache.aries.proxy:org.apache.aries.proxy:jar:1.1.10
- org.apache.aries:org.apache.aries.util:jar:1.1.3
- org.apache.calcite:calcite-core:jar:1.26.0
- org.apache.camel:camel-aws-s3:jar:3.8.0
- org.apache.camel:camel-base:jar:3.8.0
- org.apache.camel:camel-core:jar:3.8.0
- org.apache.camel:camel-http-common:jar:3.8.0
- org.apache.camel:camel-servlet:jar:3.8.0
- org.apache.camel:camel-support:jar:3.8.0
- org.apache.commons:commons-collections4:jar:4.1
- org.apache.commons:commons-compress:jar:1.21
- org.apache.commons:commons-csv:jar:1.4
- org.apache.commons:commons-exec:jar:1.3
- org.apache.commons:commons-lang3:jar:3.11
- org.apache.commons:commons-math3:jar:3.6.1
- org.apache.commons:commons-pool2:jar:2.5.0
- org.apache.commons:commons-text:jar:1.6
- org.apache.cxf:cxf-core:jar:3.4.4
- org.apache.cxf:cxf-rt-bindings-soap:jar:3.4.4
- org.apache.cxf:cxf-rt-features-logging:jar:3.4.4
- org.apache.cxf:cxf-rt-frontend-jaxrs:jar:3.4.4
- org.apache.cxf:cxf-rt-frontend-jaxws:jar:3.4.4
- org.apache.cxf:cxf-rt-rs-client:jar:3.4.4
- org.apache.cxf:cxf-rt-rs-security-jose-jaxrs:jar:3.4.4
- org.apache.cxf:cxf-rt-rs-security-jose:jar:3.4.4
- org.apache.cxf:cxf-rt-rs-security-oauth2:jar:3.4.4
- org.apache.cxf:cxf-rt-rs-security-sso-saml:jar:3.4.4
- org.apache.cxf:cxf-rt-rs-security-xml:jar:3.4.4
- org.apache.cxf:cxf-rt-transports-http:jar:3.4.1
- org.apache.cxf:cxf-rt-transports-http:jar:3.4.4

- org.apache.cxf:cxf-rt-ws-security:jar:3.4.4
- org.apache.felix:org.apache.felix.configadmin:jar:1.9.22
- org.apache.felix:org.apache.felix.fileinstall:jar:3.7.0
- org.apache.felix:org.apache.felix.framework:jar:6.0.5
- org.apache.felix:org.apache.felix.scr:jar:2.1.28
- org.apache.felix:org.apache.felix.utils:jar:1.11.8
- org.apache.httpcomponents:httpclient:jar:4.5.12
- org.apache.httpcomponents:httpcore:jar:4.4.13
- org.apache.httpcomponents:httpmime:jar:4.5.12
- org.apache.karaf.bundle:org.apache.karaf.bundle.core:jar:4.3.3
- org.apache.karaf.features:enterprise:xml:features:4.3.3
- org.apache.karaf.features:org.apache.karaf.features.core:jar:4.3.3
- org.apache.karaf.features:standard:xml:features:4.3.3
- org.apache.karaf.itests:common:jar:4.3.3
- org.apache.karaf.jaas:org.apache.karaf.jaas.boot:jar:4.3.3
- org.apache.karaf.jaas:org.apache.karaf.jaas.config:jar:4.3.3
- org.apache.karaf.jaas:org.apache.karaf.jaas.modules:jar:4.3.3
- org.apache.karaf.log:org.apache.karaf.log.core:jar:4.3.3
- org.apache.karaf.shell:org.apache.karaf.shell.console:jar:4.3.3
- org.apache.karaf.shell:org.apache.karaf.shell.core:jar:4.3.3
- org.apache.karaf.system:org.apache.karaf.system.core:jar:4.3.3
- org.apache.karaf:apache-karaf:tar.gz:4.3.3
- org.apache.karaf:apache-karaf:zip:4.3.3
- org.apache.karaf:karaf:pom:4.3.3
- org.apache.karaf:org.apache.karaf.client:jar:4.3.3
- org.apache.karaf:org.apache.karaf.util:jar:4.3.3
- org.apache.logging.log4j:log4j-1.2-api:jar:2.11.2
- org.apache.logging.log4j:log4j-api:jar:2.11.2
- org.apache.logging.log4j:log4j-api:jar:2.8.2
- org.apache.logging.log4j:log4j-core:jar:2.11.2
- org.apache.logging.log4j:log4j-slf4j-impl:jar:2.11.2
- org.apache.lucene:lucene-analyzers-common:jar:8.9.0

- org.apache.lucene:lucene-core:jar:3.0.2
- org.apache.lucene:lucene-core:jar:8.9.0
- org.apache.lucene:lucene-queryparser:jar:8.9.0
- org.apache.pdfbox:fontbox:jar:2.0.23
- org.apache.pdfbox:pdfbox-tools:jar:2.0.23
- org.apache.pdfbox:pdfbox:jar:2.0.23
- org.apache.poi:poi-ooxml-schemas:jar:4.1.2
- org.apache.poi:poi-ooxml:jar:4.1.2
- org.apache.poi:poi-scratchpad:jar:4.1.2
- org.apache.poi:poi:jar:4.1.2
- org.apache.servicemix.bundles:org.apache.servicemix.bundles.jaxb-runtime:jar:2.3.2_2
- org.apache.servicemix.bundles:org.apache.servicemix.bundles.jdom:jar:2.0.6_1
- org.apache.servicemix.bundles:org.apache.servicemix.bundles.poi:jar:4.1.2_1
- org.apache.servicemix.specs:org.apache.servicemix.specs.jaxb-api-2.3:jar:2.3_3
- org.apache.servicemix.specs:org.apache.servicemix.specs.jsr339-api-2.0:jar:2.6.0
- org.apache.shiro:shiro-core:jar:1.7.1
- org.apache.solr:solr-core:jar:8.9.0
- org.apache.solr:solr-solrj:jar:8.9.0
- org.apache.tika:tika-core:jar:1.26
- org.apache.tika:tika-parsers:jar:1.26
- org.apache.ws.commons.axiom:axiom-api:jar:1.2.14
- org.apache.ws.xmlschema:xmlschema-core:jar:2.2.3
- org.apache.wss4j:wss4j-ws-security-common:jar:2.3.0
- org.apache.wss4j:wss4j-ws-security-dom:jar:2.3.0
- org.apache.xmlbeans:xmlbeans:jar:3.1.0
- org.apache.zookeeper:zookeeper-jute:jar:3.6.2
- org.apache.zookeeper:zookeeper:jar:3.6.2
- org.assertj:assertj-core:jar:3.18.1
- org.awaitility:awaitility:jar:3.1.6
- org.bouncycastle:bcmail-jdk15on:jar:1.69
- org.bouncycastle:bcpkix-jdk15on:jar:1.69
- org.bouncycastle:bcprov-jdk15on:jar:1.69

- org.codehaus.woodstox:stax2-api:jar:4.2.1
- org.codice.countrycode:converter:jar:0.1.8
- org.codice.opendj.embedded:opendj-embedded-app:xml:features:1.3.3
- org.codice.pro-grade:pro-grade:jar:1.1.3
- org.codice.thirdparty:commons-httpclient:jar:3.1.0_1
- org.codice.thirdparty:ffmpeg:zip:bin:4.3.1_1
- org.codice.thirdparty:geotools-suite:jar:20.1_2
- org.codice.thirdparty:gt-api:jar:20.1_3
- org.codice.thirdparty:jts:jar:1.16.0_2
- org.codice.thirdparty:lucene-core:jar:3.0.2_1
- org.codice.thirdparty:ogc-filter-v_1_1_0-schema:jar:1.1.0_5
- org.codice.thirdparty:picocontainer:jar:1.3_1
- org.codice.thirdparty:tika-bundle:jar:1.26.0_1
- org.codice.usng4j:usng4j-api:jar:0.4
- org.codice.usng4j:usng4j-impl:jar:0.4
- org.cryptomator:siv-mode:jar:1.2.2
- org.eclipse.jetty.websocket:websocket-servlet:jar:9.4.42.v20210604
- org.eclipse.jetty:jetty-http:jar:9.4.42.v20210604
- org.eclipse.jetty:jetty-security:jar:9.4.42.v20210604
- org.eclipse.jetty:jetty-server:jar:9.4.42.v20210604
- org.eclipse.jetty:jetty-servlet:jar:9.4.42.v20210604
- org.eclipse.jetty:jetty-util:jar:9.4.42.v20210604
- org.eclipse.platform:org.eclipse.osgi:jar:3.16.300
- org.forgerock.commons:forgerock-util:jar:3.0.2
- org.forgerock.commons:i18n-core:jar:1.4.2
- org.forgerock.commons:i18n-slf4j:jar:1.4.2
- org.forgerock.opendj:opendj-core:jar:3.0.0
- org.forgerock.opendj:opendj-grizzly:jar:3.0.0
- org.fusesource.jansi:jansi:jar:1.18
- org.geotools.xsd:gt-xsd-gml3:jar:28.2
- org.geotools:gt-cql:jar:20.2
- org.geotools:gt-epsg-hsql:jar:28.2

- org.geotools:gt-main:jar:28.2
- org.geotools:gt-api:jar:28.2
- org.geotools:gt-referencing:jar:28.2
- org.geotools:gt-shapefile:jar:28.2
- org.geotools:gt-xml:jar:28.2
- org.glassfish.grizzly:grizzly-framework:jar:2.3.30
- org.glassfish.grizzly:grizzly-http-server:jar:2.3.25
- org.glassfish.jaxb:jaxb-runtime:jar:2.3.3
- org.hamcrest:hamcrest-all:jar:1.3
- org.hibernate:hibernate-core:jar:5.4.24.Final
- org.hisrc.w3c:xlink-v_1_0:jar:1.4.0
- org.imgscalr:imgscalr-lib:jar:4.2
- org.jasypt:jasypt:jar:1.9.3
- org.jcodec:jcodec:jar:0.2.0_1
- org.joda:joda-convert:jar:1.2
- org.jolokia:jolokia-osgi:jar:1.6.2
- org.json:json:jar:20170516
- org.jsoup:jsoup:jar:1.11.3
- org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.10.0
- org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.11.0
- org.jvnet.jaxb2_commons:jaxb2-basics-runtime:jar:0.6.0
- org.jvnet.ogc:filter-v_1_1_0:jar:2.6.1
- org.jvnet.ogc:filter-v_2_0_0-schema:jar:1.1.0
- org.jvnet.ogc:gml-v_3_1_1-schema:jar:1.1.0
- org.jvnet.ogc:gml-v_3_1_1:jar:2.6.1
- org.jvnet.ogc:gml-v_3_2_1-schema:jar:1.1.0
- org.jvnet.ogc:gml-v_3_2_1:pom:1.1.0
- org.jvnet.ogc:ows-v_1_0_0-schema:jar:1.1.0
- org.jvnet.ogc:ows-v_1_0_0:jar:2.6.1
- org.jvnet.ogc:ows-v_1_1_0-schema:jar:1.1.0
- org.jvnet.ogc:wcs-v_1_0_0-schema:jar:1.1.0
- org.jvnet.ogc:wfs-v_1_1_0:jar:2.6.1

- org.la4j:la4j:jar:0.6.0
- org.locationtech.jts:jts-core:jar:1.19.0
- org.locationtech.spatial4j:spatial4j:jar:0.8
- org.mindrot:jbcrypt:jar:0.4
- org.mockito:mockito-core:jar:3.6.28
- org.objenesis:objenesis:jar:3.1
- org.opengis:geoapi:jar:3.0.1
- org.opensaml:opensaml-core:jar:3.4.6
- org.opensaml:opensaml-soap-impl:jar:3.4.6
- org.opensaml:opensaml-xmlsec-api:jar:3.4.6
- org.opensaml:opensaml-xmlsec-impl:jar:3.4.6
- org.ops4j.pax.exam:pax-exam-container-karaf:jar:4.13.4
- org.ops4j.pax.exam:pax-exam-features:xml:4.13.4
- org.ops4j.pax.exam:pax-exam-junit4:jar:4.13.4
- org.ops4j.pax.exam:pax-exam-link-mvn:jar:4.13.4
- org.ops4j.pax.exam:pax-exam:jar:4.13.4
- org.ops4j.pax.swissbox:pax-swissbox-extender:jar:1.8.4
- org.ops4j.pax.tinybundles:tinybundles:jar:3.0.0
- org.ops4j.pax.url:pax-url-aether:jar:2.6.7
- org.ops4j.pax.url:pax-url-wrap:jar:2.6.7
- org.ops4j.pax.web:pax-web-api:jar:7.3.19
- org.ops4j.pax.web:pax-web-jsp:jar:7.3.19
- org.osgi:org.osgi.enterprise:jar:5.0.0
- org.osgi:osgi.cmpn:jar:7.0.0
- org.osgi:osgi.core:jar:7.0.0
- org.ow2.asm:asm-analysis:jar:7.1
- org.ow2.asm:asm-tree:jar:7.1
- org.ow2.asm:asm:jar:7.1
- org.ow2.asm:asm:jar:9.2
- org.pac4j:pac4j-core:jar:4.0.1
- org.pac4j:pac4j-jwt:jar:4.0.1
- org.pac4j:pac4j-oauth:jar:4.0.1

- org.pac4j:pac4j-oidc:jar:4.0.1
- org.parboiled:parboiled-core:jar:1.3.1
- org.parboiled:parboiled-java:jar:1.3.1
- org.quartz-scheduler:quartz:jar:2.3.2
- org.slf4j:jcl-over-slf4j:jar:1.7.32
- org.slf4j:jul-to-slf4j:jar:1.7.24
- org.slf4j:slf4j-api:jar:1.7.32
- org.slf4j:slf4j-ext:jar:1.7.29
- org.slf4j:slf4j-log4j12:jar:1.7.32
- org.slf4j:slf4j-simple:jar:1.7.32
- org.springframework.ldap:spring-ldap-core:jar:2.3.3.RELEASE
- org.springframework.osgi:spring-osgi-core:jar:1.2.1
- org.springframework:spring-core:jar:5.2.13.RELEASE
- org.taktik:mpegts-streamer:jar:0.1.0_2
- org.xmlunit:xmlunit-matchers:jar:2.8.2
- xalan:serializer:jar:2.7.2
- xalan:xalan:jar:2.7.2
- xerces:xercesImpl:jar:2.12.0
- xml-apis:xml-apis:jar:1.4.01
- xpp3:xpp3:jar:1.1.4c

28. Hardening Checklist

The following list enumerates the required mitigations needed for hardening. It is not intended to be a step-by-step procedure. To harden a new system, perform configuration as documented.

- ☐ Configure Auditing
- ☐ Set Directory Permissions
- ☐ Configure Keystore and Certificates
- ☐ Disallow Login Without Certificates
- ☐ Configure Certificate Revocation
  - ☐ Deny Guest User Access (if denying Guest users)
  - ☐ Allow Guest User Access (if allowing Guest users)

- ☐ Configure Guest Claim Attributes (if allowing Guest users)
- ☐ Configure Guest User Authentication
- ☐ Create unique user role
- ☐ Restricting Access to Admin Console
- ☐ Restricting Access to Features, Apps, Services, and Configurations
- ☐ Remove Default Users
- ☐ Harden Solr
- ☐ Environment Hardening

Appendix C: Metadata Reference

DDF extracts basic metadata from the resources ingested. Many file types contain additional file format-specific metadata attributes. A neutral Catalog Taxonomy enables transformation of metadata to other formats. See also a list of all formats supported for ingest.

C.1. Common Metadata Attributes

DDF supports a wide variety of file types and data types for ingest. The DDF's internal Input Transformers extract the necessary data into a generalized format. DDF supports ingest of many datatypes and commonly used file formats, such as Microsoft office products: including Word documents, Excel spreadsheets, and PowerPoint presentations, as well as .pdf files, GeoJson, and others. See complete list. Many of these file types support additional file format-specific attributes from which additional metadata can be extracted.

NOTE	These attributes are available in all the specified file formats; however, values are only present if present in the original document/resource.

These attributes are supported by any file type ingested into DDF:

Common Attributes in All Supported File Types

- metadata
- id
- modified (date)
- title (filename)
- metadata content type (mime type)
- effective (date)
- created (date)

These 'media' file types have support for additional attributes to be available when ingested into DDF:

File Types Supporting Additional Attributes

- Video Types
  - WMV
  - AVI
  - MP4
  - MOV
  - h.264 MPEG2
- Image Types
  - JPEG-2000
- Document Types
  - .DOC, .DOCX, .DOTX, .DOCM
  - .PPT, .PPTX
  - .XLS, .XLSX
  - .PDF

These are the attributes common to any of the media file types which support additional attributes:

Additional Possible Attributes Common to 'Media' File Types

- `media.format-version`
- `media.format`
- `media.bit-rate`
- `media.bits-per-sample`
- `media.compression`
- `media.encoding`
- `media.frame-center`
- `media.frame-rate`
- `media.height-pixels`
- `media.number-of-bands`
- `media.scanning-mode`
- `media.type`
- `media.duration`
- `media.page-count`
- `datatype`

- description
- contact.point-of-contact-name
- contact.contributor-name
- contact.creator-name
- contact.publisher-name
- contact.point-of-contact-phone
- topic.keyword

C.2. File Format-specific Attributes

Many file formats support additional metadata attributes that DDF is able to extract and make discoverable.

C.2.1. Mp4 Additional Attribute

Mp4 files have an additional attribute:

- ext.mp4.audio-sample-rate

C.2.2. All File Formats Supported

Supported File Types

Using the various input transformers, DDF supports ingest of the following MIME types. While ingest is possible for these files, metadata is limited unless otherwise noted.

Table 172. Application File Types

activemessage	andrew-inset	applefile
applixware	atom+xml	atomcat+xml
atomicmail	atomsvc+xml	auth-policy+xml
batch-smtp	beep+xml	bizagi-modeler
cals-1840	cbor	ccxml+xml
cea-2018+xml	cellml+xml	cnrp+xml
commonground	conference-info+xml	cpl+xml
csta+xml	cstadata+xml	cu-seeme
cybercash	davmount+xml	dca-rft
dec-dx	dialog-info+xml	dicom
dif+xml	dita+xml	dita+xml
dita+xml	dita+xml	dita+xml
dita+xml	dns	dvcs

ecmascript	edi-consent	edi-x12
edifact	emma+xml	epp+xml
epub+zip	eshop	example
fastinfoset	fastsoap	fits
font-tdpfr	gzip	h224
http	hyperstudio	ibe-key-request+xml
ibe-pkg-reply+xml	ibe-pp-data	iges
illustrator	im-iscomposing+xml	index
index.cmd	index.obj	index.response
index.vnd	inf	iotp
ipp	isup	java-archive
java-serialized-object	java-vm	javascript
json	kate	kpml-request+xml
kpml-response+xml	lost+xml	mac-binhex40
mac-compactpro	macwriteii	marc
mathematica	mathml+xml	mbms-associated-procedure-description+xml
mbms-deregister+xml	mbms-envelope+xml	mbms-msk+xml
mbms-msk-response+xml	mbms-protection-description+xml	mbms-reception-report+xml
mbms-register+xml	mbms-register-response+xml	mbms-user-service-description+xml
mbox	media_control+xml	mediaservercontrol+xml
mikey	moss-keys	moss-signature
mosskey-data	mosskey-request	mp4
mpeg4-generic	mpeg4-iod	mpeg4-iod-xmt
msword	msword2	msword5
mxf	nasdata	news-checkgroups
news-groupinfo	news-transmission	nss
ocsp-request	ocsp-response	octet-stream
oda	oebps-package+xml	ogg
onenote	parityfec	patch-ops-error+xml
pdf	pgp-encrypted	pgp-keys
pgp-signature	pics-rules	pidf+xml
pidf-diff+xml	pkcs10	pkcs7-mime
pkcs7-signature	pkix-cert	pkix-crl
pkix-pkipath	pkixcmp	pls+xml

poc-settings+xml	postscript	prs.alvestrand.titrax-sheet
prs.cww	prs.nprend	prs.plucker
qsig	quicktime	rdf+xml
reginfo+xml	relax-ng-compact-syntax	remote-printing
resource-lists+xml	resource-lists-diff+xml	riscos
rlmi+xml	rls-services+xml	rsd+xml
rss+xml	rtf	rtx
samlassertion+xml	samlmetadata+xml	sbml+xml
scvp-cv-request	scvp-cv-response	scvp-vp-request
scvp-vp-response	sdp	sereal
sereal	sereal	sereal
set-payment	set-payment-initiation	set-registration
set-registration-initiation	sgml	sgml-open-catalog
shf+xml	sieve	simple-filter+xml
simple-message-summary	simplesymbolcontainer	slate
sldworks	smil+xml	soap+fastinfoset
soap+xml	sparql-query	sparql-results+xml
spirits-event+xml	srgs	srgs+xml
ssml+xml	timestamp-query	timestamp-reply
tve-trigger	ulpfec	vemmi
vividence.scriptfile	vnd.3gpp.bsf+xml	vnd.3gpp.pic-bw-large
vnd.3gpp.pic-bw-small	vnd.3gpp.pic-bw-var	vnd.3gpp.sms
vnd.3gpp2.bcmcsinfo+xml	vnd.3gpp2.sms	vnd.3gpp2.tcap
vnd.3m.post-it-notes	vnd.accpac.simply.aso	vnd.accpac.simply.imp
vnd.acucobol	vnd.acucorp	vnd.adobe.aftereffects.project
vnd.adobe.aftereffects.template	vnd.adobe.air-application-installer-package+zip	vnd.adobe.xdp+xml
vnd.adobe.xfdf	vnd.aether.imp	vnd.airzip.filesecure.azf
vnd.airzip.filesecure.azs	vnd.amazon.ebook	vnd.americandynamics.acc
vnd.amiga.ami	vnd.android.package-archive	vnd.anser-web-certificate-issue-initiation
vnd.anser-web-funds-transfer-initiation	vnd.antix.game-component	vnd.apple.installer+xml
vnd.apple.iwork	vnd.apple.keynote	vnd.apple.numbers
vnd.apple.pages	vnd.arastra.swi	vnd.audiograph
vnd.autopackage	vnd.avistar+xml	vnd.blueice.multipass

vnd.bluetooth.ep.oob	vnd.bmi	vnd.businessobjects
vnd.cab-jscript	vnd.canon-cpdl	vnd.canon-lips
vnd.cendio.thinlinc.clientconf	vnd.chemdraw+xml	vnd.chipnuts.karaoke-mmd
vnd.cinderella	vnd.cirpack.isdn-ext	vnd.claymore
vnd.clonk.c4group	vnd.commerce-battelle	vnd.commonspace
vnd.contact.cmsg	vnd.cosmocaller	vnd.crick.clicker
vnd.crick.clicker.keyboard	vnd.crick.clicker.palette	vnd.crick.clicker.template
vnd.crick.clicker.wordbank	vnd.criticaltools.wbs+xml	vnd.ctc-posml
vnd.ctct.ws+xml	vnd.cups-pdf	vnd.cups-postscript
vnd.cups-ppd	vnd.cups-raster	vnd.cups-raw
vnd.curl.car	vnd.curl.pcurl	vnd.cybank
vnd.data-vision.rdz	vnd.denovo.fcselayout-link	vnd.dir-bi.plate-dl-nosuffix
vnd.dna	vnd.dolby.mlp	vnd.dolby.mobile.1
vnd.dolby.mobile.2	vnd.dpgraph	vnd.dreamfactory
vnd.dvb.esgcontainer	vnd.dvb.ipdcdftnotifaccess	vnd.dvb.ipdcesgaccess
vnd.dvb.ipdcroaming	vnd.dvb.iptv.alfec-base	vnd.dvb.iptv.alfec-enhancement
vnd.dvb.notif-aggregate-root+xml	vnd.dvb.notif-container+xml	vnd.dvb.notif-generic+xml
vnd.dvb.notif-ia-msglist+xml	vnd.dvb.notif-ia-registration-request+xml	vnd.dvb.notif-ia-registration-response+xml
vnd.dvb.notif-init+xml	vnd.dxr	vnd.dynageo
vnd.ecdis-update	vnd.ecowin.chart	vnd.ecowin.filerequest
vnd.ecowin.fileupdate	vnd.ecowin.series	vnd.ecowin.seriesrequest
vnd.ecowin.seriesupdate	vnd.emclient.accessrequest+xml	vnd.enliven
vnd.epson.esf	vnd.epson.msf	vnd.epson.quickanime
vnd.epson.salt	vnd.epson.ssf	vnd.ericsson.quickcall
vnd.eszigno3+xml	vnd.etsi.aoc+xml	vnd.etsi.asic-e+zip
vnd.etsi.asic-s+zip	vnd.etsi.cug+xml	vnd.etsi.iptvcommand+xml
vnd.etsi.iptvdiscovery+xml	vnd.etsi.iptvprofile+xml	vnd.etsi.iptvsad-bc+xml
vnd.etsi.iptvsad-cod+xml	vnd.etsi.iptvsad-npvr+xml	vnd.etsi.iptvueprofile+xml
vnd.etsi.mcid+xml	vnd.etsi.sci+xml	vnd.etsi.simservs+xml
vnd.eudora.data	vnd.ezpix-album	vnd.ezpix-package
vnd.f-secure.mobile	vnd.fdf	vnd.fdsn.mseed
vnd.fdsn.seed	vnd.ffsns	vnd.fints
vnd.flographit	vnd.fluxtime.clip	vnd.font-fontforge-sfd
vnd.framemaker	vnd.frogans.fnc	vnd.frogans.ltf

vnd.fsc.weblaunch	vnd.fujitsu.oasys	vnd.fujitsu.oasys2
vnd.fujitsu.oasys3	vnd.fujitsu.oasysgp	vnd.fujitsu.oasysprs
vnd.fujixerox.art-ex	vnd.fujixerox.art4	vnd.fujixerox.ddd
vnd.fujixerox.docuworks	vnd.fujixerox.docuworks.binder	vnd.fujixerox.hbpl
vnd.fut-misnet	vnd.fuzzysheet	vnd.genomatix.tuxedo
vnd.geogebra.file	vnd.geogebra.tool	vnd.geometry-explorer
vnd.gmx	vnd.google-earth.kml+xml	vnd.google-earth.kmz
vnd.grafeq	vnd.gridmp	vnd.groove-account
vnd.groove-help	vnd.groove-identity-message	vnd.groove-injector
vnd.groove-tool-message	vnd.groove-tool-template	vnd.groove-vcard
vnd.handheld-entertainment+xml	vnd.hbci	vnd.hcl-bireports
vnd.hhe.lesson-player	vnd.hp-hpgl	vnd.hp-hpid
vnd.hp-hps	vnd.hp-jlyt	vnd.hp-pcl
vnd.hp-pclxl	vnd.httphone	vnd.hydrostatix.sof-data
vnd.hzn-3d-crossword	vnd.ibm.afplinedata	vnd.ibm.electronic-media
vnd.ibm.minipay	vnd.ibm.modcap	vnd.ibm.rights-management
vnd.ibm.secure-container	vnd.iccprofile	vnd.igloader
vnd.immervision-ivp	vnd.immervision-ivu	vnd.informedcontrol.rms+xml
vnd.informix-visionary	vnd.intercon.formnet	vnd.intertrust.digibox
vnd.intertrust.nncp	vnd.intu.qbo	vnd.intu.qfx
vnd.iptc.g2.conceptitem+xml	vnd.iptc.g2.knowledgeitem+xml	vnd.iptc.g2.newsitem+xml
vnd.iptc.g2.packageitem+xml	vnd.ipunplugged.rcprofile	vnd.irepository.package+xml
vnd.is-xpr	vnd.jam	vnd.japannet-directory-service
vnd.japannet-jpnstore-wakeup	vnd.japannet-payment-wakeup	vnd.japannet-registration
vnd.japannet-registration-wakeup	vnd.japannet-setstore-wakeup	vnd.japannet-verification
vnd.japannet-verification-wakeup	vnd.jcp.javame.midlet-rms	vnd.jisp
vnd.joost.joda-archive	vnd.kahootz	vnd.kde.karbon
vnd.kde.kchart	vnd.kde.kformula	vnd.kde.kivio
vnd.kde.kontour	vnd.kde.kpresenter	vnd.kde.kspread
vnd.kde.kword	vnd.kenameaapp	vnd.kidspiration
vnd.kinar	vnd.koan	vnd.kodak-descriptor
vnd.liberty-request+xml	vnd.llamagraphics.life-balance.desktop	vnd.llamagraphics.life-balance.exchange+xml
vnd.lotus-1-2-3	vnd.lotus-approach	vnd.lotus-freelance

vnd.lotus-notes	vnd.lotus-organizer	vnd.lotus-screencam
vnd.lotus-wordpro	vnd.macports.portpkg	vnd.marlin.drm.actiontoken+xml
vnd.marlin.drm.conftoken+xml	vnd.marlin.drm.license+xml	vnd.marlin.drm.mdcf
vnd.mcd	vnd.medcalcdata	vnd.mediastation.cdkey
vnd.meridian-slingshot	vnd.mfer	vnd.mfmp
vnd.micrografx.flo	vnd.micrografx.igx	vnd.mif
vnd.mindjet.mindmanager	vnd.minisoft-hp3000-save	vnd.mitsubishi.misty-guard.trustweb
vnd.mobius.daf	vnd.mobius.dis	vnd.mobius.mbk
vnd.mobius.mqy	vnd.mobius.msl	vnd.mobius.plc
vnd.mobius.txf	vnd.mophun.application	vnd.mophun.certificate
vnd.motorola.flexsuite	vnd.motorola.flexsuite.adsi	vnd.motorola.flexsuite.fis
vnd.motorola.flexsuite.gotap	vnd.motorola.flexsuite.kmr	vnd.motorola.flexsuite.ttc
vnd.motorola.flexsuite.wem	vnd.motorola.iprm	vnd.mozilla.xul+xml
vnd.ms-artgalry	vnd.ms-asf	vnd.ms-cab-compressed
vnd.ms-excel	vnd.ms-excel.addin.macroenabled.12	vnd.ms-excel.sheet.2
vnd.ms-excel.sheet.3	vnd.ms-excel.sheet.4	vnd.ms-excel.sheet.binary.macroenabled.12
vnd.ms-excel.sheet.macroenabled.12	vnd.ms-excel.template.macroenabled.12	vnd.ms-excel.workspace.3
vnd.ms-excel.workspace.4	vnd.ms-fontobject	vnd.ms-htmlhelp
vnd.ms-ims	vnd.ms-lrm	vnd.ms-outlook
vnd.ms-outlook-pst	vnd.ms-pki.seccat	vnd.ms-pki.stl
vnd.ms-playready.initiator+xml	vnd.ms-powerpoint	vnd.ms-powerpoint.addin.macroenabled.12
vnd.ms-powerpoint.presentation.macroenabled.12	vnd.ms-powerpoint.slide.macroenabled.12	vnd.ms-powerpoint.slideshow.macroenabled.12
vnd.ms-powerpoint.template.macroenabled.12	vnd.ms-project	vnd.ms-tnef
vnd.ms-visio.drawing	vnd.ms-visio.drawing.macroenabled.12	vnd.ms-visio.stencil
vnd.ms-visio.stencil.macroenabled.12	vnd.ms-visio.template	vnd.ms-visio.template.macroenabled.12
vnd.ms-visio.viewer	vnd.ms-wmdrm.lic-chlg-req	vnd.ms-wmdrm.lic-resp

vnd.ms-wmdrm.meter-chlg-req	vnd.ms-wmdrm.meter-resp	vnd.ms-word.document.macroenabled.12
vnd.ms-word.template.macroenabled.12	vnd.ms-works	vnd.ms-wpl
vnd.ms-xpsdocument	vnd.mseq	vnd.msign
vnd.multiad.creator	vnd.multiad.creator.cif	vnd.music-niff
vnd.musician	vnd.muvee.style	vnd.ncd.control
vnd.ncd.reference	vnd.nervana	vnd.netfpx
vnd.neurolanguage.nlu	vnd.noblenet-directory	vnd.noblenet-sealer
vnd.noblenet-web	vnd.nokia.catalogs	vnd.nokia.conml+wbxml
vnd.nokia.conml+xml	vnd.nokia.iptv.config+xml	vnd.nokia.isds-radio-presets
vnd.nokia.landmark+wbxml	vnd.nokia.landmark+xml	vnd.nokia.landmarkcollection+xml
vnd.nokia.n-gage.ac+xml	vnd.nokia.n-gage.data	vnd.nokia.n-gage.symbian.install
vnd.nokia.ncd	vnd.nokia.pcd+wbxml	vnd.nokia.pcd+xml
vnd.nokia.radio-preset	vnd.nokia.radio-presets	vnd.novadigm.edm
vnd.novadigm.edx	vnd.novadigm.ext	vnd.oasis.opendocument.chart
vnd.oasis.opendocument.chart-template	vnd.oasis.opendocument.database	vnd.oasis.opendocument.formula
vnd.oasis.opendocument.formula-template	vnd.oasis.opendocument.graphics	vnd.oasis.opendocument.graphics-template
vnd.oasis.opendocument.image	vnd.oasis.opendocument.image-template	vnd.oasis.opendocument.presentation
vnd.oasis.opendocument.presentation-template	vnd.oasis.opendocument.spreadsheet	vnd.oasis.opendocument.spreadsheet-template
vnd.oasis.opendocument.text	vnd.oasis.opendocument.text-master	vnd.oasis.opendocument.text-template
vnd.oasis.opendocument.text-web	vnd.obn	vnd.olpc-sugar
vnd.oma-scws-config	vnd.oma-scws-http-request	vnd.oma-scws-http-response
vnd.oma.bcast.associated-procedure-parameter+xml	vnd.oma.bcast.drm-trigger+xml	vnd.oma.bcast.imd+xml
vnd.oma.bcast.ltkm	vnd.oma.bcast.notification+xml	vnd.oma.bcast.provisioningtrigger
vnd.oma.bcast.sgboot	vnd.oma.bcast.sgdd+xml	vnd.oma.bcast.sgdu
vnd.oma.bcast.simple-symbol-container	vnd.oma.bcast.smartcard-trigger+xml	vnd.oma.bcast.sprov+xml
vnd.oma.bcast.stkm	vnd.oma.dcd	vnd.oma.dcdc
vnd.oma.dd2+xml	vnd.oma.drm.risd+xml	vnd.oma.group-usage-list+xml

vnd.oma.poc.detailed-progress-report+xml	vnd.oma.poc.final-report+xml	vnd.oma.poc.groups+xml
vnd.oma.poc.invocation-descriptor+xml	vnd.oma.poc.optimized-progress-report+xml	vnd.oma.xcap-directory+xml
vnd.omads-email+xml	vnd.omads-file+xml	vnd.omads-folder+xml
vnd.omaloc-supl-init	vnd.openofficeorg.extension	vnd.openxmlformats-officedocument.presentationml.presentation
vnd.openxmlformats-officedocument.presentationml.slide	vnd.openxmlformats-officedocument.presentationml.slideshow	vnd.openxmlformats-officedocument.presentationml.template
vnd.openxmlformats-officedocument.spreadsheetml.sheet	vnd.openxmlformats-officedocument.spreadsheetml.template	vnd.openxmlformats-officedocument.wordprocessingml.document
vnd.openxmlformats-officedocument.wordprocessingml.template	vnd.osa.netdeploy	vnd.osgi.bundle
vnd.osgi.dp	vnd.otps.ct-kip+xml	vnd.palm
vnd.paos.xml	vnd.pg.format	vnd.pg.osasli
vnd.piaccess.application-licence	vnd.picsel	vnd.poc.group-advertisement+xml
vnd.pocketlearn	vnd.powerbuilder6	vnd.powerbuilder6-s
vnd.powerbuilder7	vnd.powerbuilder7-s	vnd.powerbuilder75
vnd.powerbuilder75-s	vnd.preminet	vnd.previewsystems.box
vnd.proteus.magazine	vnd.publishare-delta-tree	vnd.pvi.ptid1
vnd.pwg-multiplexed	vnd.pwg-xhtml-print+xml	vnd.qualcomm.brew-app-res
vnd.quark.quarkxpress	vnd.rapid	vnd.recordare.musicxml
vnd.recordare.musicxml+xml	vnd.renlearn.rlprint	vnd.rim.cod
vnd.rn-realmedia	vnd.route66.link66+xml	vnd.ruckus.download
vnd.s3sms	vnd.sbm.cid	vnd.sbm.mid2
vnd.scribus	vnd.sealed.3df	vnd.sealed.csf
vnd.sealed.doc	vnd.sealed.eml	vnd.sealed.mht
vnd.sealed.net	vnd.sealed.ppt	vnd.sealed.tiff
vnd.sealed.xls	vnd.sealedmedia.softseal.html	vnd.sealedmedia.softseal.pdf
vnd.seemail	vnd.sema	vnd.semd
vnd.semf	vnd.shana.informed.formdata	vnd.shana.informed.formtemplate
vnd.shana.informed.interchange	vnd.shana.informed.package	vnd.simtech-mindmapper
vnd.smaf	vnd.smart.teacher	vnd.software602.filler.form+xml

vnd.software602.filler.form-xml-zip	vnd.solent.sdkm+xml	vnd.spotfire.dxp
vnd.spotfire.sfs	vnd.sss-cod	vnd.sss-dtf
vnd.sss-ntf	vnd.stardivision.calc	vnd.stardivision.draw
vnd.stardivision.impress	vnd.stardivision.math	vnd.stardivision.writer
vnd.stardivision.writer-global	vnd.street-stream	vnd.sun.wadl+xml
vnd.sun.xml.calc	vnd.sun.xml.calc.template	vnd.sun.xml.draw
vnd.sun.xml.draw.template	vnd.sun.xml.impress	vnd.sun.xml.impress.template
vnd.sun.xml.math	vnd.sun.xml.writer	vnd.sun.xml.writer.global
vnd.sun.xml.writer.template	vnd.sus-calendar	vnd.svd
vnd.swiftview-ics	vnd.symbian.install	vnd.syncml+xml
vnd.syncml.dm+wbxml	vnd.syncml.dm+xml	vnd.syncml.dm.notification
vnd.syncml.ds.notification	vnd.tao.intent-module-archive	vnd.tcpdump.pcap
vnd.tmobile-livetv	vnd.trid.tpt	vnd.triscape.mxs
vnd.trueapp	vnd.truedoc	vnd.ufdl
vnd.uiq.theme	vnd.umajin	vnd.unity
vnd.uoml+xml	vnd.uplanet.alert	vnd.uplanet.alert-wbxml
vnd.uplanet.bearer-choice	vnd.uplanet.bearer-choice-wbxml	vnd.uplanet.cacheop
vnd.uplanet.cacheop-wbxml	vnd.uplanet.channel	vnd.uplanet.channel-wbxml
vnd.uplanet.list	vnd.uplanet.list-wbxml	vnd.uplanet.listcmd
vnd.uplanet.listcmd-wbxml	vnd.uplanet.signal	vnd.vcx
vnd.vd-study	vnd.vectorworks	vnd.vidsoft.vidconference
vnd.visio	vnd.visionary	vnd.vividence.scriptfile
vnd.vsf	vnd.wap.sic	vnd.wap.slc
vnd.wap.wbxml	vnd.wap.wmlc	vnd.wap.wmlscriptc
vnd.webturbo	vnd.wfa.wsc	vnd.wmc
vnd.wmf.bootstrap	vnd.wordperfect	vnd.wqd
vnd.wrq-hp3000-labelled	vnd.wt.stf	vnd.wv.csp+wbxml
vnd.wv.csp+xml	vnd.wv.ssp+xml	vnd.xara
vnd.xfdl	vnd.xfdl.webform	vnd.xmi+xml
vnd.xmpie.cpkg	vnd.xmpie.dpkg	vnd.xmpie.plan
vnd.xmpie.ppkg	vnd.xmpie.xlim	vnd.yamaha.hv-dic
vnd.yamaha.hv-script	vnd.yamaha.hv-voice	vnd.yamaha.openscoreformat
vnd.yamaha.openscoreformat.osfpvg+xml	vnd.yamaha.smaf-audio	vnd.yamaha.smaf-phrase
vnd.yellowriver-custom-menu	vnd.zul	vnd.zzazz.deck+xml

voicexml+xml	watcherinfo+xml	whoispp-query
whoispp-response	winhlp	wita
wordperfect5.1	wsdl+xml	wspolicy+xml
x-123	x-7z-compressed	x-abiword
x-ace-compressed	x-adobe-indesign	x-adobe-indesign-interchange
x-apple-diskimage	x-appleworks	x-archive
x-arj	x-authorware-bin	x-authorware-map
x-authorware-seg	x-axcrypt	x-bcpio
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-berkeley-db	x-berkeley-db	x-berkeley-db
x-bibtex-text-file	x-bittorrent	x-bplist
x-bzip	x-bzip2	x-cdlink
x-chat	x-chess-pgn	x-chrome-package
x-compress	x-coredump	x-corelpresentations
x-cpio	x-csh	x-debian-package
x-dex	x-director	x-doom
x-dosexec	x-dtbncx+xml	x-dtbook+xml
x-dtbresource+xml	x-dvi	x-elc
x-elf	x-emf	x-erdas-hfa
x-executable	x-fictionbook+xml	x-filemaker
x-font-adobe-metric	x-font-bdf	x-font-dos
x-font-framemaker	x-font-ghostscript	x-font-libgrx
x-font-linux-psf	x-font-otf	x-font-pcf
x-font-printer-metric	x-font-snf	x-font-speedo
x-font-sunos-news	x-font-ttf	x-font-type1
x-font-vfont	x-foxmail	x-futuresplash
x-gnucash	x-gnumeric	x-grib
x-gtar	x-hdf	x-hwp
x-hwp-v5	x-ibooks+zip	x-isatab
x-isatab-assay	x-isatab-investigation	x-iso9660-image
x-itunes-ipa	x-java-jnilib	x-java-jnlp-file
x-java-pack200	x-kdelnk	x-killustrator
x-latex	x-lha	x-lharc

x-matlab-data	x-matroska	x-mobipocket-ebook
x-ms-application	x-ms-installer	x-ms-wmd
x-ms-wmz	x-ms-xbap	x-msaccess
x-msbinder	x-mscardfile	x-msclip
x-msdownload	x-msdownload	x-msdownload
x-msdownload	x-msdownload	x-msdownload
x-msdownload	x-msmediaview	x-msmetafile
x-msmoney	x-mspublisher	x-msschedule
x-msterminal	x-mswrite	x-mysql-db
x-mysql-misam-compressed-index	x-mysql-misam-data	x-mysql-misam-index
x-mysql-table-definition	x-netcdf	x-object
x-pkcs12	x-pkcs7-certificates	x-pkcs7-certreqresp
x-project	x-prt	x-quattro-pro
x-rar-compressed	x-roxio-toast	x-rpm
x-sas	x-sas-access	x-sas-audit
x-sas-backup	x-sas-catalog	x-sas-data
x-sas-data-index	x-sas-dmdb	x-sas-fdb
x-sas-itemstor	x-sas-mddb	x-sas-program-data
x-sas-putility	x-sas-transport	x-sas-utility
x-sas-view	x-sc	x-sfdu
x-sh	x-shapefile	x-shar
x-sharedlib	x-shockwave-flash	x-silverlight-app
x-snappy-framed	x-sqlite3	x-staroffice-template
x-stuffit	x-stuffitx	x-sv4cpio
x-sv4crc	x-tar	x-tex
x-tex-tfm	x-texinfo	x-tika-iworks-protected
x-tika-java-enterprise-archive	x-tika-java-web-archive	x-tika-msoffice
x-tika-msoffice-embedded	x-tika-msoffice-embedded	x-tika-msoffice-embedded
x-tika-msworks-spreadsheet	x-tika-old-excel	x-tika-ooxml
x-tika-ooxml-protected	x-tika-staroffice	x-tika-unix-dump
x-tika-visio-ooxml	x-uc2-compressed	x-ustar
x-vhd	x-vmdk	x-wais-source
x-webarchive	x-x509-ca-cert	x-xfig
x-xmind	x-xpinstall	x-xz
x-zoo	x400-bp	xcap-att+xml

xcap-caps+xml	xcap-el+xml	xcap-error+xml
xcap-ns+xml	xcon-conference-info+xml	xcon-conference-info-diff+xml
xenc+xml	xhtml+xml	xhtml-voice+xml
xml	xml-dtd	xml-external-parsed-entity
xmpp+xml	xop+xml	xquery
xslfo+xml	xslt+xml	xspf+xml
xv+xml	zip	zlib

Table 173. Audio File Types

32kadpcm	3gpp	3gpp2
ac3	adpcm	amr
amr-wb	amr-wb+	asc
basic	bv16	bv32
clearmode	cn	dat12
dls	dsr-es201108	dsr-es202050
dsr-es202211	dsr-es202212	dvi4
eac3	evrc	evrc-qcp
evrc0	evrc1	evrcb
evrcb0	evrcb1	evrcwb
evrcwb0	evrcwb1	example
g719	g722	g7221
g723	g726-16	g726-24
g726-32	g726-40	g728
g729	g7291	g729d
g729e	gsm	gsm-efr
ilbc	l16	l20
l24	l8	lpc
midi	mobile-xmf	mp4
mp4a-latm	mpa	mpa-robust
mpeg	mpeg4-generic	ogg
opus	parityfec	pcma
pcma-wb	pcmu	pcmu-wb
prs.sid	qcelp	red
rtp-enc-aescm128	rtp-midi	rtx
smv	smv-qcp	smv0
sp-midi	speex	t140c

t38	telephone-event	tone
ulpfec	vdvi	vmr-wb
vnd.3gpp.iufp	vnd.4sb	vnd.adobe.soundbooth
vnd.audiokoz	vnd.celp	vnd.cisco.nse
vnd.cmles.radio-events	vnd.cns.anp1	vnd.cns.inf1
vnd.digital-winds	vnd.dlna.adts	vnd.dolby.heaac.1
vnd.dolby.heaac.2	vnd.dolby.mlp	vnd.dolby.mps
vnd.dolby.pl2	vnd.dolby.pl2x	vnd.dolby.pl2z
vnd.dts	vnd.dts.hd	vnd.everad.plj
vnd.hns.audio	vnd.lucent.voice	vnd.ms-playready.media.pya
vnd.nokia.mobile-xmf	vnd.nortel.vbk	vnd.nuera.ecelp4800
vnd.nuera.ecelp7470	vnd.nuera.ecelp9600	vnd.octel.sbc
vnd.qcelp	vnd.rhetorex.32kadpcm	vnd.sealedmedia.softseal.mpeg
vnd.vmx.cvsd	vorbis	vorbis-config
x-aac	x-adbcm	x-aiff
x-dec-adbcm	x-dec-basic	x-flac
x-matroska	x-mod	x-mpegurl
x-ms-wax	x-ms-wma	x-oggflac
x-oggpcm	x-pn-realaudio	x-pn-realaudio-plugin
x-wav		

Table 174. Chemical File Types

x-cdx	x-cif	x-cmdf
x-cml	x-csml	x-pdb
x-xyz		

Table 175. Image File Types

bmp	cgm	example
fits	g3fax	gif
icns	ief	jp2
jpeg	jpm	jpx
naplps	nitf	png
prs.btif	prs.pti	svg+xml
t38	tiff	tiff-fx
vnd.adobe.photoshop	vnd.adobe.premiere	vnd.cns.inf2
vnd.djvu	vnd.dwg	vnd.dxb

vnd.dxf	vnd.dxf	vnd.dxf
vnd.fastbidsheet	vnd.fpx	vnd.fst
vnd.fujixerox.edmics-mmr	vnd.fujixerox.edmics-rlc	vnd.globalgraphics.pgb
vnd.microsoft.icon	vnd.mix	vnd.ms-modi
vnd.net-fpx	vnd.radiance	vnd.sealed.png
vnd.sealedmedia.softseal.gif	vnd.sealedmedia.softseal.jpg	vnd.svf
vnd.wap.wbmp	vnd.xiff	webp
x-bpg	x-cmu-raster	x-cmx
x-freehand	x-jp2-codestream	x-jp2-container
x-ms-bmp	x-niff	x-pcx
x-pict	x-portable-anymap	x-portable-bitmap
x-portable-graymap	x-portable-pixmap	x-raw-adobe
x-raw-canon	x-raw-casio	x-raw-epson
x-raw-fuji	x-raw-hasselblad	x-raw-imacon
x-raw-kodak	x-raw-leaf	x-raw-logitech
x-raw-mamiya	x-raw-minolta	x-raw-nikon
x-raw-olympus	x-raw-panasonic	x-raw-pentax
x-raw-phaseone	x-raw-rawzor	x-raw-red
x-raw-sigma	x-raw-sony	x-rgb
x-xbitmap	x-xcf	x-xpixmap
x-xwindowdump		

Table 176. Message File Types

cpim	delivery-status	disposition-notification
example	external-body	global
global-delivery-status	global-disposition-notification	global-headers
http	imdn+xml	news
partial	rfc822	s-http
sip	sipfrag	tracking-status
vnd.si.simp	x-emlx	

Table 177. Model File Types

example	iges	mesh
vnd.dwf	vnd.dwf	vnd.dwf
vnd.dwf	vnd.dwfx+xps	vnd.flatland.3dml
vnd.gdl	vnd.gs-gdl	vnd.gs.gdl

vnd.gtw	vnd.moml+xml	vnd.mts
vnd.parasolid.transmit.binary	vnd.parasolid.transmit.text	vnd.vtu
vrml		

Table 178. Multipart File Types

alternative	appledouble	byteranges
digest	encrypted	example
form-data	header-set	mixed
parallel	related	report
signed	voice-message	

Table 179. Text File Types

asp	aspdotnet	calendar
css	csv	directory
dns	ecmascript	enriched
example	html	iso19139+xml
parityfec	plain	prs.fallenstein.rst
prs.lines.tag	red	rfc822-headers
richtext	rtp-enc-aescm128	rtx
sgml	t140	tab-separated-values
troff	ulpfec	uri-list
vnd.abc	vnd.curl	vnd.curl.dcurl
vnd.curl.mcurl	vnd.curl.scurl	vnd.dmclientscript
vnd.esmertec.theme-descriptor	vnd.fly	vnd.fmi.flexstor
vnd.graphviz	vnd.in3d.3dml	vnd.in3d.spot
vnd.iptc.anpa	vnd.iptc.newsml	vnd.iptc.nitf
vnd.latex-z	vnd.motorola.reflex	vnd.ms-mediapackage
vnd.net2phone.commcenter.command	vnd.si.uricatalogue	vnd.sun.j2me.app-descriptor
vnd.trolltech.linguist	vnd.wap.si	vnd.wap.sl
vnd.wap.wml	vnd.wap.wmlscript	vtt
x-actionscript	x-ada	x-applescript
x-asciidoc	x-aspectj	x-assembly
x-awk	x-basic	x-c++hdr
x-c++src	x-cgi	x-chdr
x-clojure	x-cobol	x-coffeescript

x-coldfusion	x-common-lisp	x-csharp
x-csrc	x-d	x-diff
x-eiffel	x-emacs-lisp	x-erlang
x-expect	x-forth	x-fortran
x-go	x-groovy	x-haml
x-haskell	x-haxe	x-idl
x-ini	x-java-properties	x-java-source
x-jsp	x-less	x-lex
x-log	x-lua	x-matlab
x-ml	x-modula	x-objcsrc
x-ocaml	x-pascal	x-perl
x-php	x-prolog	x-python
x-rexx	x-rsrc	x-rst
x-ruby	x-scala	x-scheme
x-sed	x-setext	x-sql
x-stsrc	x-tcl	x-tika-text-based-message
x-uuencode	x-vbasic	x-vbdotnet
x-vbscript	x-vcalendar	x-vcard
x-verilog	x-vhdl	x-web-markdown
x-yacc	x-yaml	

Table 180. Video File Types

3gpp	3gpp-tt	3gpp2
bmpeg	bt656	celb
daala	dv	example
h261	h263	h263-1998
h263-2000	h264	jpeg
jpeg2000	mj2	mp1s
mp2p	mp2t	mp4
mp4v-es	mpeg	mpeg4-generic
mpv	nv	ogg
parityfec	pointer	quicktime
raw	rtp-enc-aescm128	rtx
smpte292m	theora	ulpfec
vc1	vnd.cctv	vnd.dlna.mpeg-tts
vnd.fvt	vnd.hns.video	vnd.iptvforum.1dparityfec-1010

vnd.iptvforum.1dparityfec-2005	vnd.iptvforum.2dparityfec-1010	vnd.iptvforum.2dparityfec-2005
vnd.iptvforum.ttsavc	vnd.iptvforum.ttsmpeg2	vnd.motorola.video
vnd.motorola.videop	vnd.mpegurl	vnd.ms-playready.media.pyv
vnd.nokia.interleaved-multimedia	vnd.nokia.videovoip	vnd.objectvideo
vnd.sealed.mpeg1	vnd.sealed.mpeg4	vnd.sealed.swf
vnd.sealedmedia.softseal.mov	vnd.vivo	webm
x-dirac	x-f4v	x-flc
x-fli	x-flv	x-jng
x-m4v	x-matroska	x-mng
x-ms-asf	x-ms-wm	x-ms-wmv
x-ms-wmx	x-ms-wvx	x-msvideo
x-oggrgb	x-ogguvs	x-oggyuv
x-ogm	x-sgi-movie	

Table 181. x-conference File Types

x-cooltalk		

C.3. Catalog Taxonomy Definitions

To facilitate data sharing while maximizing the usefulness of metadata, the attributes on resources are normalized into a common taxonomy that maps to attributes in the desired output format.

NOTE The taxonomy is presented here for reference only.

C.3.1. Core Attributes

Table 182. Core Attributes. **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
title	A name for the resource. Dublin Core elements-title.	String	< 1024 characters	

Term	Definition	Datatype	Constraints	Example Value
source-id	ID of the source where the Metacard is cataloged. While this cannot be moved or renamed for legacy reasons, it should be treated as non-mappable, since this field is overwritten by the system when federated results are retrieved.	String	< 1024 characters	
metadata-content-type **[deprecated]** *see Media Attributes*	Content type of the resource.	String	< 1024 characters	
metadata-content-type-version **[deprecated]** *see Media Attributes*	Version of the metadata content type of the resource.	String	< 1024 characters	
metadata-target-namespace **[deprecated]** *see Media Attributes*	Target namespace of the metadata.	String	< 1024 characters	
metadata	Additional XML metadata describing the resource.	XML	A valid XML string per RFC 4825 (must be well-formed but not necessarily schema-compliant).	

Term	Definition	Datatype	Constraints	Example Value
location	The primary geospatial location of the resource.	Geometry	Valid Well Known Text (WKT) per http://www.opengeospatial.org/standards/wkt-crs *Coordinates must be in **lon-lat** coordinate order*	POINT(150 30)
expiration	The expiration date of the resource.	Date		
effective **[deprecated]**	The `effective` time of the event or resource represented by the metacard. Deprecated in favor of created and modified.	Date		
point-of-contact **[deprecated]**	The name of the point of contact for the resource. This is set internally to the user's subject and should be considered read-only to other DDF components.	String	< 1024 characters	
resource-uri	Catalog-specific location of the resource for the metacard. This URI is used for internal catalog requests.	String	Valid URI per RFC 2396	
resource-download-url	URL location of the resource for the metacard. This attributes provides a client-resolvable URL to the download location of the resource. Clients should use this URL for download requests.	String	Valid URL per RFC 2396	
resource-size	Size in bytes of resource.	String	Although this type cannot be changed for legacy reasons, its value should always be a parsable whole number.	

Term	Definition	Datatype	Constraints	Example Value
thumbnail	The thumbnail for the resource in JPEG format.	Base 64 encoded binary string per RFC 4648	⇐ 128 KB	
description	An account of the resource. Dublin Core elements-description.	String		
checksum	Checksum value for the primary resource for the metacard.	String	< 1024 characters	
checksum-algorithm	Algorithm used to calculate the checksum on the primary resource of the metacard.	String	< 1024 characters	
created	The creation date of the resource Dublin Core terms-created.	Date		
modified	The modification date of the resource Dublin Core terms-modified.	Date		
language	The languages of the resource. Dublin Core language.	List of Strings	Alpha-3 language codes per ISO_639-2	
resource.derived-uri	Catalog-specific Locations for accessing the resources derived from another source (for example, an overlay of a larger image). This URI is used for internal catalog requests.	List of Strings	Valid URI per RFC 2396	
resource.derived-download-url	Download URLs for accessing the resources derived from another source (for example, an overlay of a larger image). Clients should use this URL for download requests.	List of Strings	Valid URLs per RFC 2396	

Term	Definition	Datatype	Constraints	Example Value
datatype	The generic types of the resource including the Dublin Core terms-type. DCMI Type term labels are expected here as opposed to term names.	List of Strings	`Collection`, `Dataset`, `Event`, `Image`, `Interactive Resource`, `Moving Image`, `Physical Object`, `Service`, `Software`, `Sound`, `Still Image`, and/or `Text`	

C.3.2. Associations Attributes

Table 183. Associations (Attributes in this group represent associations between resources.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
metacard.associations.derived	ID of one or more metacards derived from this metacard.	List of Strings	A valid metacard ID (conventionally, a type 4 random UUID with hyphens removed).	70809f17782c42b8ba15747b86b50ebf
metacard.associations.related	ID of one or more metacards related to this metacard.	List of Strings	A valid metacard ID (conventionally, a type 4 random UUID with hyphens removed).	70809f17782c42b8ba15747b86b50ebf
associations.external	One or more URI's identifying external associated resources.	List of Strings	A valid URI.	https://infocorp.org/wikia/reference

C.3.3. Contact Attributes

Table 184. Contact (Attributes in this group reflect metadata about different kinds of people/groups/units/organizations that can be associated with a metacard.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
contact.creator-name	The names of this metacard's creators.	List of Strings	< 1024 characters per entry	
contact.creator-address	The physical addresses of this metacard's creators.	List of Strings	< 1024 characters per entry	
contact.creator-email	The email addresses of this metacard's creators.	List of Strings	A valid email address per RFC 5322.	
contact.creator-phone	The phone numbers of this metacard's creators.	List of Strings	< 1024 characters per entry	
contact.publisher-name	The names of this metacard's publishers.	List of Strings	< 1024 characters per entry	
contact.publisher-address	The physical addresses of this metacard's publishers.	List of Strings	< 1024 characters per entry	
contact.publisher-email	The email addresses of this metacard's publishers.	List of Strings	A valid email address per RFC 5322.	
contact.publisher-phone	The phone numbers of this metacard's publishers.	List of Strings	< 1024 characters per entry	
contact.contributor-name	The name of the contributors to this metacard.	List of Strings	< 1024 characters per entry	
contact.contributor-address	The physical addresses of the contributors to this metacard.	List of Strings	< 1024 characters per entry	
contact.contributor-email	The email addresses of the contributors to this metacard.	List of Strings	A valid email address per RFC 5322.	
contact.contributor-phone	The phone numbers of the contributors to this metacard.	List of Strings	< 1024 characters per entry	

Term	Definition	Datatype	Constraints	Example Value
contact.point-of-contact-name	The names of the points of contact for this metacard.	List of Strings	< 1024 characters per entry	
contact.point-of-contact-address	The physical addresses of the points of contact for this metacard.	List of Strings	< 1024 characters per entry	
contact.point-of-contact-email	The email addresses of the points of contact for this metacard.	List of Strings	A valid email address per RFC 5322.	
contact.point-of-contact-phone	The phone numbers of the points of contact for this metacard.	List of Strings	< 1024 characters per entry	

C.3.4. DateTime Attributes

Table 185. DateTime (Attributes in this group reflect temporal aspects about the resource.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
datetime.start	Start times for the resource.	List of Dates		
datetime.end	End times for the resource.	List of Dates		
datetime.name	A descriptive name for the corresponding temporal attributes. See datetime.start and datetime.end.	List of Strings	< 1024 characters per entry	

C.3.5. History Attributes

Table 186. History (Attributes in this group describe the history/versioning of the metacard.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
metacard.version.id	Internal attribute identifier for which metacard this version is representing	String	A valid metacard ID (conventionally, a type 4 random UUID with hyphens removed).	70809f17782c42b8ba15747b86b50ebf
metacard.version.edited-by	Internal attribute identifying the editor of a history metacard.	String	A valid email address per RFC 5322	
metacard.version.versioned-on	Internal attribute for the versioned date of a metacard version.	Date		
metacard.version.action	Internal attribute for the action associated with a history metacard.	String	One of `Deleted`, `Deleted-Content`, `Versioned`, `Versioned-Content`	
metacard.version.tags	Internal attribute for the tags that were on the original metacard.	String		
metacard.version.type	Internal attribute for the metacard type of the original metacard.	String		
metacard.version.type-binary	Internal attribute for the serialized metacard type of the original metacard.	Binary		
metacard.version.resource-uri	Internal attribute for the original resource uri.	URI		

C.3.6. Location Attributes

Table 187. Location (Attributes in this group reflect location aspects about the resource.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
location.altitude-meters	Altitude of the resource in meters.	List of Doubles	> 0	
location.country-code	One or more country codes associated with the resource.	List of Strings	ISO_3166-1 alpha-3 codes	

Term	Definition	Datatype	Constraints	Example Value
location.crs-code	Coordinate reference system code of the resource.	List of Strings	< 1024 characters per entry	EPSG:4326
location.crs-name	Coordinate reference system name of the resource.	List of Strings	< 1024 characters per entry	WGS 84

C.3.7. Media Attributes

Table 188. Media (Attributes in this group reflect metadata about media in general.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
media.format	The file format, physical medium, or dimensions of the resource. Dublin Core elements-format	String	< 1024 characters	txt, docx, xml - typically the extension or a more complete name for such, note that this is not the mime type
media.format-version	The file format version of the resource. Note that the syntax can vary widely from format to format.	String	< 1024 characters	POSIX, 2016, 1.0
media.bit-rate	The bit rate of the media, in bits per second.	Double		
media.frame-rate	The frame rate of the video, in frames per second.	Double		
media.frame-center	The center of the video frame.	Geometry	Valid Well Known Text (WKT)	
media.height-pixels	The height of the media resource in pixels.	Integer		
media.width-pixels	The width of the media resource in pixels.	Integer		

Term	Definition	Datatype	Constraints	Example Value
media.compression	The type of compression this media uses. EXIF STANAG 4559 NC, NM, C1, M1, I1, C3, M3, C4, M4, C5, M5, C8, M8	String	One of the values defined for EXIF Compression tag.	
media.bits-per-sample	The number of bits per image component.	Integer		
media.type (RFC 2046)	A two-part identifier for file formats and format content.	String	A valid mime-type per https://www.ietf.org/rfc/rfc2046.txt	application/json
media.encoding	The encoding format of the media.	List of Strings	< 1024 characters per entry	MPEG-2, RGB
media.number-of-bands	The number of spectral bands in the media.	Integer	The significance of this number is instrumentation-specific, but there are eight commonly recognized bands. https://en.wikipedia.org/wiki/Multispectral_image	
media.scanning-mode (MPEG2)	Indicate if progressive or interlaced scans are being applied.	String	PROGRESSIVE, INTERLACED	

C.3.8. Metacard Attributes

Table 189. Metacard (Attributes in this group describe the metacard itself.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
metacard.created	The creation date of the metacard.	Date		

Term	Definition	Datatype	Constraints	Example Value
metacard.modified	The modified date of the metacard.	Date		
metacard.owner	The email address of the metacard owner.	String	A valid email address per RFC 5322	
metacard-tags	Collections of data that go together, used for filtering. query results. NOTE: these are system tags. For descriptive tags, Topic Attributes.	List of Strings	< 1024 characters per entry	

C.3.9. Security Attributes

Table 190. Security (Attributes in this group relate to security of the resource and metadata.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
security.access-groups	Attribute name for storing groups to enforce access controls upon that enable a user to read and write a metacard.	List of Strings	< 1024 characters per entry	
security.access-individuals	Attribute name for storing the email addresses of users to enforce access controls upon that enable the ability to read and write a metacard.	List of Strings	A valid email address per RFC 5322.	
security.access-individuals-read	Attribute name for storing the email addresses of users to enforce access controls upon that can read, but not explicitly write to a metacard.	List of Strings	A valid email address per RFC 5322.	
security.access-groups-read	Attribute name for storing groups to enforce access controls upon that enable a user to read, but not necessarily write to a metacard.	List of Strings	< 1024 characters per entry	

Term	Definition	Datatype	Constraints	Example Value
security.access-administrators	Attribute name for explicitly stating who has the permissions to modify the access control values of a metacard. These values include changing the security.access-groups, security.access-individuals and the security.access-administrators values.	List of Strings	A valid email address per RFC 5322.	

C.3.10. Topic Attributes

Table 191. Topic (Attributes in this group describe the topic of the resource.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
topic.category	A category code from a given vocabulary.	List of Strings	A valid entry from the corresponding controlled vocabulary.	
topic.keyword	One or more keywords describing the subject matter of the metacard or resource.	List of Strings	< 1024 characters per entry	
topic.vocabulary	An identifier of a controlled vocabulary from which the topic category is derived.	List of Strings	Valid URI per RFC 2396.	

C.3.11. Validation Attributes

Table 192. Validation (Attributes in this group identify validation issues with the metacard and/or resource.) **Injected by default.**

Term	Definition	Datatype	Constraints	Example Value
validation-warnings	Textual description of validation warnings on the resource.	List of Strings	< 1024 characters per entry	
validation-errors	Textual description of validation errors on the resource.	List of Strings	< 1024 characters per entry	

Index

A

Access plugins, 276
Action Framework, 316
Action Providers, 316
Actions, 316
Adding New User Roles, 43
Admin, 447
Admin Application, 25
Admin Console, 71
Alerts, 140
Apache Shiro, 322
Application Whitelists, 523
Assertion Consumer Service, 337
Asynchronous Processing Framework, 317
Atom Query Response Transformer, 240
Attribute injections, 360
Attribute Type Definition File, 356
Audit logging, 45
Authentication Types, 111

B

Bundles, 443

C

Catalog, 449
Catalog API, 223, 226
Catalog API implementations, 345
Catalog Application, 25
Catalog Backup Plugin, 166, 282
Catalog Framework, 222
Catalog Framework Camel Component, 231
Catalog Plugins, 263, 373
Catalog Policy Plugin, 283
Catalog Provider, 370
Catalog Providers, 101, 122
Catalog REST Endpoint, 214
Catalog REST endpoint, 135
Catalog sources, 121
Catalog Store, 371
Catalog Stores, 122
Catalog Taxonomy, 543
Catalog Taxonomy Definitions, 561
Catalog Whitelist, 526
Certificate Revocation List, 96
Checksum plugin, 284
Client Info plugin, 284
Command Scheduler, 155
Common storage, 446
Connected Source, 371
Connected Sources, 122
Console command, 72
Console Commands, 146
Content Directory Monitor, 104, 163
Content Item, 309
Content Retrieval Endpoints, 176
Content URI Access plugin, 284
Contributions, 442
CQL syntax, 147
CRUD Endpoints, 175
CSW endpoint, 135, 188
CSW Query Response Transformer, 252
Custom endpoints, 362
Custom Mime Type Resolver, 262
`custom.system.properties`, 74

D

Data Access Plugins, 110
Data attributes, 111
Data Management, 100
DDF Mime Type Mapper, 261
Default Attribute Values, 358
Demo Certificate Authority, 39
Disallowing Login Without Certificates, 96
Download Resource ActionProvider, 316
Duplication Validator, 167

E

Embedded LDAP Server, 328
Encrypting Passwords, 141
Enumeration Validator, 167
Environment Hardening, 114
Event Processor, 285
Eventing, 321
Expansion Service, 340

Expiration Date plugin, 285

F

Failover Proxy, 68
Fanout Proxy, 137
Feature Commands, 72
Federated Identity, 344
Federated Source, 370
Federated Sources, 121
Federation, 118, 175
Filter, 412
Filter Delegates, 423
Filter Plugin, 286
FilterBuilder API, 315
Filtering, 113
Filters, 314
Firewall Port Configuration, 67
Future Date Validator, 167

G

GeoCoder plugin, 287
GeoJSON input transformer, 253
GeoJSON Metacard Transformer, 247
GeoJSON Query Response Transformer, 237
Global Attribute Validators, 353
Global configuration settings, 74
Guest Access, 87
Guest access, 111
Guest Interceptor, 335

H

Hardening Checklist, 542
Hardware Requirements, 47
High Availability, 67
Highly Available Cluster, 138
Historian Policy Plugin, 288
Hostname, 102

I

IdP Logout Action Provider, 316
Ingest Command, 163
Ingest Endpoints, 175
Ingesting, 162
Initial Startup, 61
Input Transformers, 363
Insecure Defaults Service, 44
Integrating, 174
Internet Explorer 11 Enhanced Security Configuration, 66
ISO3 Country Code Validator, 167

J

Java Dependency List, 533
Java Requirements, 48
Jetty Authenticator, 336
Jetty ThreadPool settings, 140
JPEG2000 Thumbnail converter, 288
JSON Definition Files, 436

K

Karaf Logout Action, 317
Keycloak, 91
Keyhole Markup Language, 135
Keystores, 57
KML, 135
KML Metacard Transformer, 248
KML Network Link endpoint, 135
KML Query Response Transformer, 256
KML Style Mapper, 238

L

Landing Page, 138
LDAP, 91
LDAP Logout Action, 317
Logout Page, 139
LogViewer, 159

M

Managed Services, 142
Match-All Mapping, 335
Match-One Mapping, 335
Metacard, 301
Metacard Attribute Security Policy Plugin, 108, 289
Metacard Backup File Storage Provider, 289
Metacard Backup S3 Storage Provider, 290
Metacard filtering, 339
Metacard Groomer Pre-Ingest plugin, 291
Metacard Ingest Network Plugin, 291
Metacard Resource Size post-query plugin, 293

Metacard Transformer ActionProvider, 317
Metacard Transformers, 367
Metacard Type, 301
Metacard Type definition files, 350
Metacard Validation Marker Plugin, 109
Metacard Validator definitions, 352
Metacard Validators, 352
Metacard Validity Filter Plugin, 109, 294
Metacard Validity Marker Pre-Ingest plugin, 294
Metacard WKT Validator, 167
Metadata, 301
Metadata Exchange, 337
Metadata Metacard Transformer, 236
Metrics Reporting, 157
MFA, 91
Microsoft PowerPoint, 237
Mime Type Mapper, 260
Mime Type Resolver, 261
Monitoring, 157
Multi-factor authentication, 91

O

OAuth 2.0, 88
OGC Filter, 314
Online Certificate Status Protocol, 100
Online Certificate Status Protocol (OCSP) server, 96
OpenID Connect
  OIDC, 88
OpenSearch Endpoint, 182
OpenSearch endpoint, 136
Operation plugin, 295
OSGi, 221, 442
Overlay ActionProvider, 317

P

Packages Removed From Whitelist, 523
Past Date Validator, 167
Pattern Validator, 167
PDF Input Transformer, 252
PDP, 334
PEP, 339
Persistence Store Commands, 155
Platform, 473
Platform Application, 25
Platform Commands, 154
Platform UI Themes, 139
Platform Whitelist, 530
Plugins, 263
Point of Contact Policy Plugin, 295
Policy Plugins, 114
Policy plugins, 275
Post-Create storage plugins, 281
Post-federated-query plugins, 280
Post-ingest plugins, 277
Post-Process Plugins, 278
Post-query plugins, 279
Post-resource plugins, 280
Post-Update storage plugins, 281
PPTX Input Transformer, 237
Pre-Create storage plugins, 281
Pre-delivery plugins, 275, 282
Pre-federated-query plugins, 279
Pre-ingest plugins, 277
Pre-query plugins, 278
Pre-Resource plugins, 280
Pre-subscription plugins, 282
Pre-Update storage plugins, 281
Processing Post Ingest Plugin, 296
Prometheus endpoint, 181
Pub-Sub Endpoints, 176

Q

Queries, 314
Queries Endpoint, 177
Query, 427
Query Endpoints, 176
Query Response Transformer Consumer, 256
Query Response Transformers, 368
Quick Install of DDF, 36
Quick Install of DDF on a remote headless server, 37
Quick Install of SolrCloud, 36
Quick Install Prerequisites, 35
Quick Start Certificates, 38
Quick start configuration, 42
Quick start guide, 35

R

Range Validator, 167
Realms, 322
Registry Clients, 407
Relationship Validator, 167
Remote Source, 121
Removing Default Users, 95
Required Attributes Metacard Validator, 167
Resource components, 311
Resource Metacard Transformer, 246
Resource Reader, 312, 408
Resource URI Policy Plugin, 296
Resource Writer, 314, 410
Resources, 311
REST services, 88
REST sources, 88
Restricting Access to Admin Console, 93
Restricting Access to Features
    Apps
        Services, 94
Run DDF as a Managed Service, 142

S

S3, 290
SAML ECP, 338
SAML Handler, 337
SAML Identity Provider, 90
Schematron, 168
Search UI, 26, 519
Search UI Whitelist, 532
Security, 478
Security Application, 25
Security Audit Plugin, 110, 296
Security Core API, 325
Security Core application, 325
Security Core Commons, 326
Security Core Services, 326
Security Encryption API, 326
Security Encryption application, 326
Security Encryption Commands, 327
Security Encryption Implementation, 327
Security Framework, 321, 336
Security Hardening, 45
Security LDAP, 328
Security Logging Plugin, 297
Security Manager, 322
Security PDP, 334
Security PDP AuthZ Realm, 334
Security PEP, 339
Security PEP Interceptor, 339
Security Plugin, 297
Security policies, 111
Security Policy Decision Point, 334
Security Policy Enforcement Point, 339
Security Services, 336
Security Whitelist, 531
Self-Signed Certificates, 41
Setting Directory Permissions, 53
Setting Internal Policies, 113
Simple Search UI, 172
Single Sign On
    SSO, 64
Size Validator, 167
Solr Admin UI, 101
Solr Catalog, 494
Solr Catalog Application, 26
Solr Catalog Whitelist, 532
Solr Commands, 151
Sources, 121, 370
Spatial, 495
Spatial Application, 26
SSH Search Interface, 226
SSL, 118
Standard Catalog Framework, 229
Startup, 144
Stateless, 446
STS Claims Handlers, 380
Subject, 322
Subscriptions, 437
Subscriptions Commands, 151
System search, 165
System Usage Message, 108
System users, 92

T

Tags Filter Plugin, 297
Thumbnail Metacard Transformer, 237
Tika Input Transformer, 260

Tika Mime Type Resolver, 262
Token validators, 380
Transformers, 233
Troubleshooting, 159
Truststore, 58
Two-factor authentication, 91

U

URL Resource Reader, 312
User attributes, 111

V

Validator Plugins, 167
Video Input transformer, 252
Video Thumbnail Plugin, 298
View Metacard ActionProvider, 317

W

Web Context Policy Manager, 111
Web Service Security, 336
Workspace Access Plugin, 299
Workspace Pre-Ingest Plugin, 299
Workspace Sharing Policy Plugin, 299
WSS, 336

X

XACML Policies, 432
XACML policy, 114
XML Attribute Security Policy Plugin, 110, 300
XML Input Transformer, 251
XML metacard transformer, 243
XML Query Response Transformer, 245

Z

Zookeeper, 137