Distributed Data Framework Overview

Version 2.22.0. Copyright (c) Codice Foundation

# Table of Contents

License

Copyright (c) Codice Foundation.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

This document last updated: 2020-01-31.

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 will be installing, maintaining, and supporting 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 will use the existing applications to support their external frameworks. This section will provide details for finding, accessing and using the components of DDF.

Developers

Developing | Developers will 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 (e.g., `/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 ⧉, where they will 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) will be returned. If a version is not specified, then all metadata records for the specified datatype(s) regardless of version will be 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 will read the metadata from the ingested file and populate 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, e.g., 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. 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.
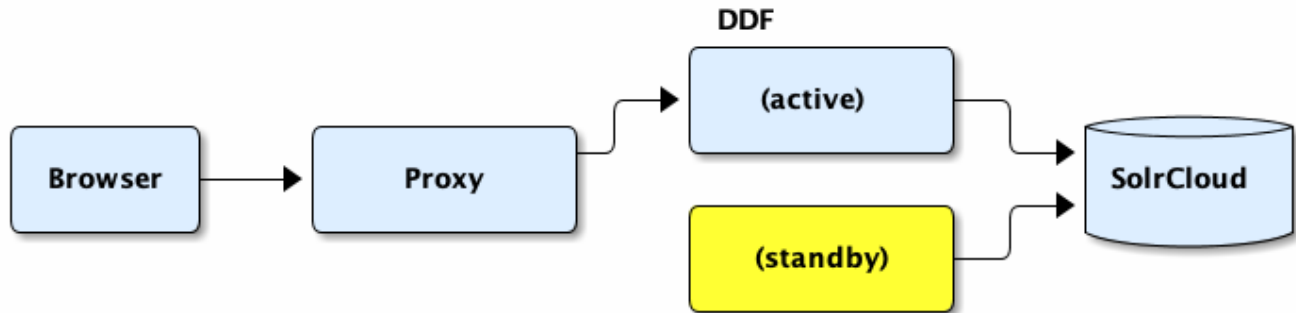
In a Highly Available Cluster, DDF has failover capabilities when a DDF node fails.
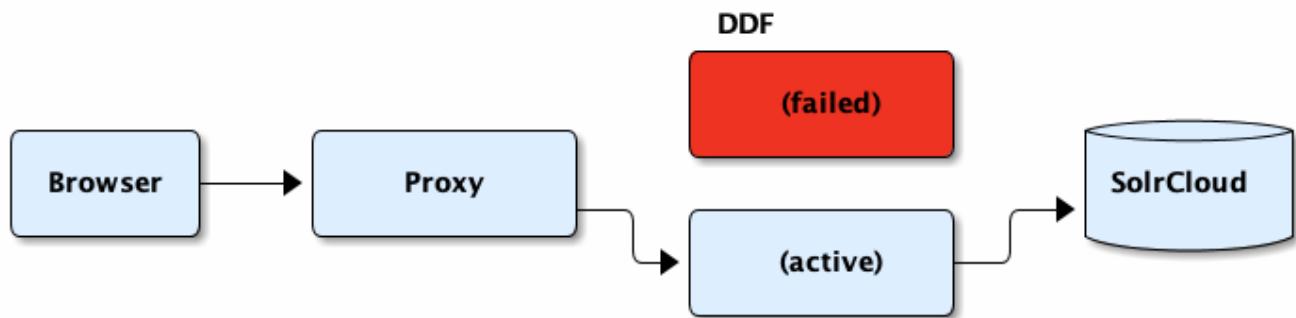
NOTE	The word "node", from a High Availability perspective, is one of the two DDF systems running within the Highly Available Cluster. Though there are multiple systems running with the Highly Available Cluster, it is still considered a single DDF from a user's perspective or from other DDFs' perspectives.

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 will be arbitrarily chosen to be the active node, and the other will be 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 will route all traffic to the active node. If the active node fails for any reason, the standby node will become active and the failover proxy will route all traffic to the new active node. See the below diagrams for more detail.

Highly Available Cluster

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.

3.9.1. 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 Interfaces:
  - Simple
  - Intrigue
- 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

3.10. Standards Supported by DDF

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

3.10.1. 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

Standard (public standards linked where available)	Endpoints	Sources	Status
OGC Web Feature Service WFS 1.0/1.1/2.0 ⌦		WFS 1.0 Source, WFS 1.1 Source, WFS 2.0 Source	Supported
OGC WPS 2.0 ⌦ Web Processing Service	WPS Endpoint		Experimental
OpenSearch ⌦	OpenSearch Endpoint	OpenSearch Source	Supported
File Transfer Protocol (FTP) ⌦	FTP Endpoint		Supported
Atlassian Confluence®		Atlassian Confluence® Federated Source	Supported

3.10.2. 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
AVIs	`avi`	Standard attributes
Keyhole Markup Language (KML) ⌦	`kml`	Standard attributes
Dublin Core ⌦	`n/a`	Standard attributes

3.10.3. 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
File Transfer Protocol (FTP) ☐ / File Transfer Protocol Secure (FTPS) ☐	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
Central Authentication Service (CAS) ☐	Supported

Table 7. Security and SSO Endpoints Provided by DDF

| Standard	Support Status
Security Token Service (STS) ☐	Supported

Standard	Support Status
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