



Distributed Data Framework

Quick Start Guide

Version 2.14.1. Copyright (c) Codice Foundation

Table of Contents

- License 1
- 1. Quick Start Tutorial 2
 - 1.1. Installing (Quick Start)..... 2
 - 1.1.1. Quick Install Prerequisites 2
 - 1.1.2. Quick Install of DDF 5
 - 1.1.3. Quick Install of DDF on a remote headless server 6
 - 1.2. Certificates (Quick Start) 6
 - 1.2.1. Demo Certificate Authority (CA) 7
 - 1.2.1.1. Creating New Server Keystore Entry with the CertNew Scripts..... 7
 - 1.2.1.2. Dealing with Lack of DNS 8
 - 1.2.2. Creating Self-Signed Certificates 9
 - 1.2.2.1. Creating a custom CA Key and Certificate 9
 - 1.2.2.2. Sign Certificates Using the custom CA 10
 - 1.2.3. Updating Settings After Changing Certificates 10
 - 1.3. Configuring (Quick Start) 10
 - 1.4. Ingesting (Quick Start) 11
 - 1.4.1. Ingesting Sample Data 11

License

Copyright (c) Codice Foundation.

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

This document last updated: 2019-04-01 18:05:29:143.

1. Quick Start Tutorial

This quick tutorial will enable install, configuring and using 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 will demonstrate:

- ✓ [Prerequisites](#).
- ✓ [Quick Install of DDF](#).
- ✓ [Ingesting Data](#).

1.1. 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 will automatically shut down.

1.1.1. Quick Install Prerequisites

Hardware Requirements (Quick Install)

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

Java Requirements (Quick Install)

Set up Java to run DDF.

- For a runtime system:
 - Install [Oracle JRE 8 x64](#) or [OpenJDK 8 JRE](#)
- For a development system:
 - Install/Upgrade to Java 8 x64 [J2SE 8 SDK](#)
 - The recommended version is [8u60](#) or later.
 - Java Version and Build numbers must contain only number values.
- Microsoft Windows and Linux are supported. For more information about supported versions, see [Installation Prerequisites](#)
- [JRE 8 x64](#) or [OpenJDK 8 JRE](#) must be installed.

- If the JRE was installed, the `JRE_HOME` environment variable must be set to the location where the JRE is installed.
- If the JDK was installed, the `JAVA_HOME` environment variable must be set to the location where the JDK is installed.

1. *Setting `JAVA_HOME` variable (Replace `<JAVA_VERSION>` with the version and build number installed.)*

1. Determine Java Installation Directory (This varies between operating system versions).

Find Java Path in Windows

```
for %i in (java.exe) do @echo. %~$PATH:i
```

*Find Java Path in *nix*

```
which java
```

1. Copy path to Java installation. (example: `/usr/java/<JAVA_VERSION>`)
2. Set `JAVA_HOME` or `JRE_HOME` by replacing `<PATH_TO_JAVA>` with the copied path in this command:

If JDK was installed:

Setting `JAVA_HOME` on Windows

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

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

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

*Setting `JAVA_HOME` on *nix*

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

*Adding `JAVA_HOME` to `PATH` Environment Variable on *nix*

```
export PATH=$JAVA_HOME/bin\: $PATH
```

If JRE was installed:

Setting JRE_HOME on Windows

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

Adding JRE_HOME to PATH Environment Variable on Windows

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

*Setting JRE_HOME on *nix*

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

*Adding JRE_HOME to PATH Environment Variable on *nix*

```
export PATH=$JRE_HOME/bin\: $PATH
```

WARNING

**nix*

Unlink **JAVA_HOME** if it is already linked to a previous version of the JRE: **unlink JAVA_HOME**

TIP

*Verify that the **JAVA_HOME** was set correctly.*

Windows

```
echo %JAVA_HOME%
```

**nix*

```
echo $JAVA_HOME
```

File Descriptor Limit on Linux

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

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

NOTE

- (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
```

Check System Time

WARNING

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

1.1.2. Quick Install of DDF

1. Download the DDF [zip file](#).
2. Install DDF by unzipping the zip file.

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.

WARNING

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-2.14.1.zip
```

3. This will create an installation directory, which is typically created with the name and version of the application. This installation directory will be referred to as `<DDF_HOME>`. (Substitute the actual directory name.)

4. Start DDF by running the `<DDF_HOME>/bin/ddf` script (or `ddf.bat` on Windows).
5. 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.
6. The Command Console will display.

Command Console Prompt

```
ddf@local>
```

1.1.3. Quick Install of DDF on a remote headless server

If DDF is being installed on a remote server that has no user interface some additional steps must be taken prior to starting the system.

1. Update any references to localhost in the following files. These references to localhost should be updated to match either the hostname or IP of the system.
 - `<DDF_HOME>/etc/custom.system.properties`
 - `<DDF_HOME>/etc/users.properties`
 - `<DDF_HOME>/etc/users.attributes`
2. From the console go to `<DDF_HOME>/etc/certs`.
 - a. If using a hostname run: `sh CertNew.sh -cn <hostname> -san "DNS:<hostname>"` (or `CertNew -cn <hostname> -san "DNS:<hostname>"` on windows).
 - b. If using an IP address run: `sh CertNew.sh -cn <IP> -san "IP:<IP>"` (or `CertNew -cn <IP> -san "IP:<IP>"` on windows).
3. Proceed with starting the system and continue as usual.

1.2. 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 will need 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 will be prompted to upload new trust/key stores.

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

qualified hostname of the server running the DDF instance.

1.2.1. 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.

1.2.1.1. 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 (e.g. "<FQDN>", where <FQDN> could be something like cluster.yoyo.com)
- `<dn>` represents a distinguished name as a comma-delimited string (e.g. "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 (e.g. "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 (e.g. "`<FQDN>`", where `<FQDN>` could be something like `cluster.yoyo.com`)
- `<dn>` represents a distinguished name as a comma-delimited string (e.g. "`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 (e.g. "`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 will be 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.

1.2.1.2. Dealing with Lack of DNS

In some cases DNS may not be available and the system will need to be configured to work with IP

addresses.

Options can be given to the CertNew Scripts to generate certs that will 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.

1.2.2. 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 is preferred over a command line interface.

1.2.2.1. 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
```

1.2.2.2. 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 will be used during system configuration to replace the default certificates.

1.2.3. Updating Settings After Changing Certificates

After changing the certificates it will be 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: <pre>sed -i 's/localhost/<FQDN IP>/g' <DDF_HOME>/etc/users.* <DDF_HOME>/etc/custom.system.properties</pre>
------------	---

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

1.3. 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 will be automatically granted these permissions.**
 - iii. **Guest users will not be 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 will not**

be 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**

1.4. Ingesting (Quick Start)


Now that DDF has been configured, ingest some sample data to demonstrate search capabilities.

This is one way to ingest into the catalog, for a complete list of the different methods, see [Ingesting Data](#).

1.4.1. Ingesting Sample Data

1. Download a sample valid [GeoJson file here](#) .

2. Navigate in the browser to Intrigue at `https://{FQDN}:{PORT}/search/catalog`.

3. Select the Menu icon () in the upper left corner

4. Select **Upload**.

5. Drag and drop the sample file or click to navigate to it.

6. Select **Start** to begin upload.

NOTE XML metadata for text searching is not automatically generated from GeoJson fields.

Querying from Intrigue (`https://{FQDN}:{PORT}/search/catalog`) will return the record for the file ingested:

1. Select the Menu icon () and return to **Workspaces**.

2. Search for the ingested data.

NOTE The sample data was selected as an example of well-formed metadata. Other data can and should be used to test other usage scenarios.