Extending DDF

Table of Contents

License

Overview

This guide discusses the several extension points and components permitted by the Distributed Data Framework (DDF) Catalog API. Using code examples, diagrams, and references to specific instances of the Catalog API, this guide provides details on how to develop and integrate various DDF components.

Building DDF

Prerequisites

- Install J2SE 8 SDK (http://www.oracle.com/technetwork/java/javase/downloads/index.html).
- Verify that the JAVA_HOME environment variable is set to the newly installed JDK location, and that the PATH includes `%JAVA_HOME%\bin` (for Windows) or `$JAVA_HOME$/bin` (*nix). Install Maven 3.1.0 or later (http://maven.apache.org/download.cgi). Verify that the PATH includes the `MVN_HOME/bin` directory. In addition, access to a Maven repository with the latest project artifacts and dependencies is necessary in order for a successful build. The following sample settings.xml (the default settings file) can be used to access the public repositories with the required artifacts. For more help on how to use the settings.xml file, refer to the Maven settings reference page (http://maven.apache.org/settings.html).

Sample settings.xml file

```
<settings>
<!-- If proxy is needed
<proxies>
<proxy>
</proxy>
</proxies>
-->
</settings>
```

TIP

Handy Tip on Encrypting Passwords

See this Maven guide on how to encrypt the passwords in your settings.xml.

Procedures

Run the Build

NOTE

In order to run through a full build, be sure to have a clone for all repositories in the same folder:

ddf (https://github.com/codice/ddf.git)
ddf-admin (https://github.com/codice/ddf-admin.git)
ddf-catalog (https://github.com/codice/ddf-catalog.git)
ddf-content (https://github.com/codice/ddf-content.git)
ddf-parent (https://github.com/codice/ddf-parent.git)
ddf-platform (https://github.com/codice/ddf-platform.git)
ddf-spatial (https://github.com/codice/ddf-spatial.git)
ddf-support (https://github.com/codice/ddf-support.git)
ddf-ui (https://github.com/codice/ddf-ui.git)

- Build command example for one individual repository.

```
# Build is run from the top level of the specified repository in a command line prompt or
terminal.
cd ddf-support
mvn clean install

# At the end of the build, a BUILD SUCCESS will be displayed.
```

- Build command example to build all repositories. This must be performed at the top level folder that contains all the repositories. A command list would look like this:

```
# Build is run from the top level folder that contains all the repositories in a command
line prompt or terminal.

cd ddf-support
mvn clean install

cd ../ddf-parent
mvn clean install

cd ../ddf-platform
mvn clean install

cd ../ddf-admin
mvn clean install

cd ../ddf-catalog
mvn clean install

cd ../ddf-content
mvn clean install

cd ../ddf-spatial
MVN_HOME clean install

cd ../ddf-ui
mvn clean install

cd ../ddf
mvn clean install

# This will fully compile each individual app. From here you may hot deploy the necessary
apps on top of the DDF Kernel.
```

WARNING	To use the updated apps in a DDF distribution, update the versions referenced in the "ddf" repository.

NOTE	[The zip distribution of DDF is contained in the DDF app in the distribution/ddf/target directory after the DDF app is built.

- Optionally, create a reactor pom that will allow you to perform the entire build process by calling a build on one pom rather than all of them. This pom must reside in the top-level folder that holds all the repositories. An example of the file would be:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

     <groupId>org.codice.ddf</groupId>
     <artifactId>reactor</artifactId>
     <version>1.0.0-SNAPSHOT</version>
     <packaging>pom</packaging>
     <name>DDF Reactor</name>
     <description>Distributed Data Framework (DDF) is an open source, modular integration
framework</description>

     <modules>
         <module>ddf-support</module>
         <module>ddf-parent</module>
         <module>ddf-platform</module>
         <module>ddf-admin</module>
         <module>ddf-security</module>
         <module>ddf-catalog</module>
         <module>ddf-content</module>
         <module>ddf-spatial</module>
         <module>ddf-solr</module>
         <module>ddf-ui</module>
         <module>ddf</module>
     </modules>

    <build>
         <plugins>
             <plugin>
                 <groupId>org.apache.maven.plugins</groupId>
                 <artifactId>maven-deploy-plugin</artifactId>
                 <version>2.4</version>
                 <configuration>
                     <!-- Do not deploy the reactor pom -->
                     <skip>true</skip>
                 </configuration>
             </plugin>
         </plugins>
     </build>

</project>
```

NOTE It may take several moments for Maven to download the required dependencies in the first build. Build times may vary based on network speed and machine specifications.

WARNING In certain circumstances, the build may fail due to a 'java.lang.OutOfMemory: Java heap space' error. This error is due to the large number of sub-modules in the DDF build, which causes the heap space to run out in the main Maven JVM. To fix this issue, set the system variable `MAVEN_OPTS` with the value `-Xmx512m` before running the build. Example on Linux system with the bash shell: `export MAVEN_OPTS='-Xmx512m'`

Troubleshoot Build Errors on ddf-admin and ddf-ui on a Windows Platform

Currently, the developers are using the following tools:

Name	Version
bower	1.3.2
node.js	v0.10.26
npm	1.4.3

NOTE

There have been intermittent build issues during the bower install. The error code that shows is an EPERM related to either 'renaming' files or 'unlinking' files. This issue has been tracked multiple times on the bower github page. The following link contains the most recent issue that was tracked: https://github.com/bower/bower/issues/991

This issue will be closely monitored for a full resolution. Until a proper solution is found, there are some options that may solve the issue. . Re-run the build. Occasionally, the issue occurs on first run and will resolve itself on the next. . Clean out the cache. There may be a memory issue, and a cache clean may help solve the issue.

+

```
bower cache clean
npm cache clean
```

+ .Reinstall bower. An occasional reinstall may solve the issue.

+

```
npm uninstall -g bower && npm install -g bower
```

+ . Download and use Cygwin to perform the build. This may allow a user to simulate a run on a *nix system, which may not experience these issues.

These options are taken from suggestions provided on github issue tickets. There have been several tickets created and closed, and several workarounds have been suggested. However, it appears that the issue still exists. Once more information develops on the resolution of this issue, this page will be updated.

DDF Development Prerequisites

NOTE Development requires full knowledge of the DDF Catalog.

DDF is written in Java and requires a moderate amount of experience with the Java programming language, along with Java terminology, such as packages, methods, classes, and interfaces. DDF uses a small OSGi runtime to deploy components and applications. Before developing for DDF, it is necessary that developers have general knowledge on OSGi and the concepts used within. This includes, but is not limited to, Catalog Commands and the following topics:

- The Service Registry
  - How services are registered
  - How to retrieve service references

- Bundles
  - Their role in OSGi
  - How they are developed

Documentation on OSGi can be viewed at the OSGi Alliance website (http://www.osgi.org). Helpful literature for beginners includes OSGi and Apache Felix 3.0 Beginner's Guide by Walid Joseph Gédéon and OSGi in Action: Creating Modular Applications in Java by Richard Hall, Karl Pauls, Stuart McCulloch, and David Savage. For specific code examples from DDF, source code can be seen in the OSGi Services section.

Getting Set Up

To develop on DDF, access to the source code via Github is required.

Integrated Development Environments (IDE)

The DDF source code is not tied to any particular IDE. However, if a developer is interested in setting up the Eclipse IDE, they can view the Sonatype guide (http://books.sonatype.com/m2eclipse-book/reference/) on developing with Eclipse.

Additional Documentation

Additional documentation on developing with the core technologies used by DDF can be found on their respective websites.

Notably: * Karaf (http://karaf.apache.org/) * CXF (http://cxf.apache.org/docs/overview.html) * Geotools (http://docs.geotools.org/latest/developer/)

Major Directories

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

Directory Name	Description
bin	Scripts to start and stop DDF
data	The working directory of the system – installed bundles and their data
data/log/ddf.log	Log file for DDF, logging all errors, warnings, and (optionally) debug statements. This log rolls up to 10 times, frequency based on a configurable setting (default=1 MB)

Directory Name	Description
deploy	Hot-deploy directory – KARs and bundles added to this directory will be hot-deployed (Empty upon DDF installation)
docs	The DDF Catalog API Javadoc
etc	Directory monitored for addition/modification/deletion of third party .cfg configuration files
etc/ddf	Directory monitored for addition/modification/deletion of DDF-related .cfg configuration files (e.g., Schematron configuration file)
etc/templates	Template .cfg files for use in configuring DDF sources, settings, etc., by copying to the etc/ddf directory.
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.

Formatting Source Code

A code formatter for the Eclipse IDE that can be used across all DDF projects will allow developers to format code similarly and minimize merge issues in the future.

DDF uses an updated version of the Apache ServiceMix Code Formatter (http://servicemix.apache.org/developers/building.html) for code formatting.

DOWNLOAD THIS FILE: link:ddf-eclipse-code-formatter.xml

1. Follow the link.
2. Right-click on **Raw**.
3. Select **Save As**.

Load the Code Formatter Into the Eclipse IDE

1. In Eclipse, select **Window** **Preferences**. The Preferences window opens.
2. Select **Java** **Code Style** **Formatter**.
3. Select the **Edit...** button and load the attached ddf-eclipse-code-formatter.xml file.
4. Select the **OK** button.

Load the Code Formatter Into IntelliJ IDEA

IntelliJ IDEA 13 is capable of importing Eclipse's Code Formatter directly from within IntelliJ without the use of any plugins.

1. Open **IntelliJ** IDEA.
2. Select **File** **Settings** **Code Style** **Java**.
3. Select **Manage**.
4. Select the **Import** button to Import ddf-eclipse-code-formatter.xml file.

Format Your Source Code Using Eclipse

A developer may write code and format it before saving.

1. Before the file is saved, highlight all of the source code in the IDE editor window.
2. Right-click on the highlighted code.
3. Select **Source** **Format**. The code formatter is applied to the source code and the file can be saved.

Set Up Save Actions in Eclipse

A developer can also set up Save Actions to format the source code automatically.

1. Open Eclipse.
2. Select **Window** **Preferences** (**Eclipse** **Preferences** on Mac). The Preferences window opens.
3. Select **Java** **Editor** **Save Actions**.
4. Select **Perform the selected actions on save**.
5. Select **Format source code**.
6. Select **Format all lines** or **Format edited lines**, as necessary.
7. Optionally, select **Organize imports** (recommended).
8. Select the **Apply** button.
9. Select the **OK** button.

Format Source Code Using IntelliJ

In the toolbar, select **Code** **Reformat Code** or use the keyboard shortcut `Ctrl-Alt-L`.

Ensuring Compatibility

Compatibility Goals

The DDF framework, like all software, will mature over time. Changes will be made to improve efficiency, add features, and fix bugs. To ensure that components built for DDF and its sub-frameworks are compatible, developers must use caution when establishing dependencies from developed components.

Guidelines for Maintaining Compatibility

DDF Framework

For components written at the DDF Framework level (see Developing at the Framework Level), adhere to the following specifications:

Standard/Specification	Version	Current Implementation (subject to change)
OSGi Framework	4.2	Apache Karaf 2.x Eclipse Equinox
OSGi Enterprise Specification	4.2	Apache Aries (Blueprint) Apache Felix FileInstall and ConfigurationAdmin and Web Console (for Metatype support)

IMPORTANT Avoid developing dependencies on the implementations directly, as compatibility in future releases is not guaranteed.

DDF Catalog API

For components written for the DDF Catalog (see Developing Catalog Components), only dependencies on the current major version of the Catalog API should be used. Detailed documentation of the Catalog API can be found in the Catalog API Javadocs.

Dependency	Version Interval	Notes

DDF Catalog API	[2.0, 3.0)	Major version will be incremented (to 3.0) if/when compatibility is broken with the 2.x API.

DDF Software Versioning

DDF follows the Semantic Versioning White Paper for bundle versioning (see Software Versioning).

Third Party and Utility Bundles

It is recommended to avoid building directly on included third party and utility bundles. These components do provide utility (e.g., JScience) 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 like `commons-geospatial`, 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 bundle(s).

Best Practices

- Always use a version number when exporting a package. In the following example, the `docs` represents the project and artifactId of the package being exported. The `2.8.2` represents the version of the project.

Export Example

```
<Export-Package>
  docs*;version=2.8.2
</Export-Package>
```

- Try to avoid deploying multiple versions of a bundle. Although OSGi is designed to support multiple versions, other developers may not include the versions of the packages that are being imported. If the bundle is versioned and designed appropriately, typically, having multiple versions of the bundle will not be an issue. However, if each bundle is competing for a specific resource, race conditions may occur. Third party and utility bundles (often denoted by commons in the bundle name) are the general exception to this rule, as these bundles will likely function as expected with multiple versions deployed.

Development Recommendations

Javascript

Avoid using **console.log**

Package Names

Use singular package names.

Author Tags

Author tags are discouraged from being placed in the source code, as they can be a barrier to collaboration and have potential legal ramifications.

Unit Testing

All code should contain unit tests that are able to test out any localized functionality within that class. When working with OSGi, code may have references to various services and other areas that are not available at compile-time. One way to work around the issue of these external dependencies is to use a mocking framework.

NOTE

Recommended Framework

The recommended framework to use with DDF is Mockito: https://github.com/mockito/mockito. This test-level dependency is managed by the ddf-parent pom and is used to standardize the version being used across DDF.

Logging

There are many logging frameworks available for Java.

NOTE

Recommended Framework

To maintain the best compatibility, the recommended logging framework is Simple Logging Facade for Java (SLF4J) (http://www.slf4j.org/), specifically the slf4j-api. SLF4J allows a very robust logging API while letting the backend implementation be switched out seamlessly. Additionally, it is compatible with pax logging and natively implemented by logback.

DDF code uses the first five SLF4J log levels:

1. trace (the least serious)
2. debug

3. info

4. warn

5. error (the most serious)

Examples:

```
//Check if trace is enabled before executing expense XML processing
if (LOGGER.isTraceEnabled()) {
      LOGGER.trace("XML returned: {}", XMLUtils.toString(xml));
}
//It is not necessary to wrap with LOGGER.isTraceEnabled() since slf4j will not construct
the String unless
//trace level is enabled
LOGGER.trace("Executing search: {}", search);
```

Dependency Injection Frameworks

It is highly recommended to use a dependency injection framework, such as Blueprint, Spring-DM, or iPojo for non-advanced OSGi tasks. Dependency injection frameworks allow for more modularity in code, keep the code's business logic clean of OSGi implementation details, and take the complexity out of the dynamic nature of OSGi. In OSGi, services can be added and removed at any time, and dependency injection frameworks are better suited to handle these types of situations. Allowing the code to be clean of OSGi packages also makes code easier to reuse outside of OSGi. These frameworks provide code conveniences of service registration, service tracking, configuration property management, and other OSGi core principles.

Basic Security

(Provided by Pierre Parrend (http://www.slideshare.net/kaihackbarth/security-in-osgi-applications-robust-osgi-platforms-secure-bundles))

- Bundles should
  - Never use synchronized statements that rely on third party code. Keep multi-threaded code in mind when using synchronized statements in general, as they can lead to performance issues.
  - Only have dependencies on bundles that are trusted.
- Shared Code
  - Provide only final static non-mutable fields.
  - Set security manager calls during creation in all required places at the beginning of methods.

        - All Constructors
        - clone() method if a class implements Cloneable
        - readObject(ObjectInputStream) if the class implements Serializable
    - Have security check in final methods only.
- Shared Objects (OSGi services)
    - Only have basic types and serializable final types as parameters.
    - Perform copy and validation (e.g., null checks) of parameters prior to using them.
    - Do not use Exception objects that carry any configuration information.

OSGi Services

DDF uses resource injection to retrieve and register services to the OSGi registry. There are many resource injection frameworks that are used to complete these operations. Blueprint and Spring DM are both used by DDF. There are many tutorials and guides available on the Internet for both of these frameworks. Refer to the Additional Resources section for details not covered in this guide. Links to some of these guides are given in the External Links section of this page.

Spring DM - Retrieving a Service Instance

Spring DM example of retrieving and injecting services

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:osgi="http://www.springframework.org/schema/osgi">

  <osgi:reference id="ddfCatalogFramework" interface="ddf.catalog.CatalogFramework" />

  <bean class="my.sample.NiftyEndpoint">
     <constructor-arg ref="ddfCatalogFramework" />
  </bean>
</beans>
```

Line #	Action
5	Retrieves a Service from the Registry
8	Instantiates a new object, injecting the retrieved Service as a constructor argument

Blueprint - Retrieving a Service Instance

Blueprint example of retrieving and injecting services

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

 <reference id="ddfCatalogFramework" interface="ddf.catalog.CatalogFramework" />

 <bean class="my.sample.NiftyEndpoint" >
    <argument ref="ddfCatalogFramework" />
 </bean>

</blueprint>
```

Line #	Action
3	Retrieves a Service from the Registry
6	Instantiates a new object, injecting the retrieved Service as a constructor argument

Blueprint - Registering a Service into the Registry

Creating a bean and registering it into the Service Registry

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

  <bean id="transformer" class="my.sample.NiftyTransformer"/>

  <service ref="transformer" interface="ddf.catalog.transform.QueryResponseTransformer"
/>

</blueprint>
```

Line #	Action
3	Instantiates a new object
5	Registers the object instance created in Line 3 as a service that implements the `ddf.catalog.transform.QueryResponseTransformer` interface

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. Alternative Bundle Creation Methods

TIP	Using Maven is not necessary to create bundles. Alternative tools exist, and OSGi manifest files can also be created by hand, although hand editing should be avoided by most developers.

See external links (below) for resources that give in-depth guides on creating bundles.

Creating a Bundle

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.

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>DDF DOCS</Bundle-Name>
        <Export-Package />
        <Bundle-SymbolicName>ddf.docs</Bundle-SymbolicName>
        <Import-Package>
          ddf.catalog,
          ddf.catalog.*
        </Import-Package>
      </instructions>
    </configuration>
  </plugin>
...
</build>
...
```

Deploying a Bundle

A bundle is typically installed in two ways:

1. 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 will be installed inside the runtime. It can be updated or deleted and changes will be 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.

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 the CAB Client is deployed and running.

Verifying with grep

```
ddf@local>list | grep -i cab
[ 162] [Active    ] [       ] [  ] [ 80] DDF :: Registry :: CAB Client (2.0.0)
```

The state is `Active`, indicating that the bundle is ready for program execution.

Additional Resources

- Blueprint
  - http://aries.apache.org/modules/blueprint.html
  - http://www.ibm.com/developerworks/opensource/library/os-osgiblueprint/
  - http://static.springsource.org/osgi/docs/2.0.0.M1/reference/html/blueprint.html
- Spring DM
  - http://www.springsource.org/osgi

- Lessons Learned from it-agile (PDF)
- http://www.martinlippert.org/events/OOP2010-OSGiLessonsLearned.pdf

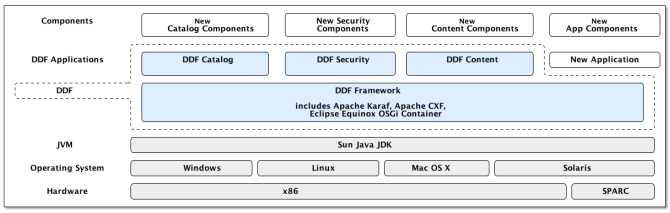
- Creating Bundles
  - http://blog.springsource.com/2008/02/18/creating-osgi-bundles/
- Bundle States
  - http://static.springsource.org/osgi/docs/1.2.1/reference/html/bnd-app-ctx.html

UI Development Recommendations

Recommendations for developing UI components.

"White Box" DDF Architecture

Architecture Diagram



As depicted in the architectural diagram above, 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 DDF out-of-the-box.

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 (https://tools.codice.org/#DDFArchitecture-AdditionalUpstreamDependencies).

DDF as a 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.

Nomenclature

Capability

A general term used to refer to an ability of the system

Application

One or more features that together form a cohesive collection of capabilities

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.

OSGi Core

DDF makes use of OSGi v4.2 to provide several capabilities:

- Has a Microkernel-based foundation, which is lightweight due to its origin in embedded systems.
- Enables integrators to easily customize components to run on their system.
- Software applications are deployed as OSGi components, or bundles. Bundles are modules that can be deployed into the OSGi container (Eclipse Equinox OSGi Framework by default).
- Bundles provide flexibility allowing integrators to choose the bundles that meet their mission needs.
- Bundles provide reusable modules that can be dropped in any container.
- Provides modularity, module-based security, and low-level services, such as Hypertext Transfer Protocol (HTTP), logging, events (basic publish/subscribe), and dependency injection.
- Implements a dynamic component model that allows application updates without downtime. Components can be added or updated in a running system.
- Standardized Application Configuration (ConfigurationAdmin and MetaType)

OSGi is not an acronym, but if more context is desired the name Open Specifications Group Initiative has been suggested.

More information on OSGi is available at http://www.osgi.org/.

Built on Apache Karaf

Apache Karaf is a FOSS product that includes an OSGi framework and adds extra functionality, including:

Web Administration Console

Useful for configuring bundles, installing/uninstalling features, and viewing services.

System Console

Provides command line administration of the OSGi container. All functionality in the Web Administration Console can also be performed via this command line console.

Logging

Provides centralized logging to a single log file (data/logs/ddf.log) utilizing log4j.

Provisioning

Of libraries or applications.

Security

Provides a security framework based on Java Authentication and Authorization Service (JAAS).

Deployer

Provides hot deployment of new bundles by dropping them into the <INSTALL_DIR>/deploy directory.

Blueprint

Provides an implementation of the OSGi Blueprint Container specification that defines a dependency injection framework for dealing with dynamic configuration of OSGi services.

- DDF uses the Apache Aries implementation of Blueprint. More information can be found at http://aries.apache.org/modules/blueprint.htm.

Spring DM

An alternative dependency injection framework. DDF is not dependent on specific dependency injection framework. Blueprint is recommended.

Additional Upstream Dependencies

DDF is a customized distribution of Apache Karaf, and therefore includes all the capabilities of Apache Karaf. DDF also includes additional FOSS components to provide a richer set of capabilities. Integrated components include their own dependencies, but at the platform level, DDF includes the following upstream dependencies:

Apache CXF

Apache CXF is an open source services framework. CXF helps build and develop services using front end programming APIs, such as JAX-WS and JAX-RS. More information can be found at http://cxf.apache.org.

Apache Commons

Provides a set of reusable Java components that extends functionality beyond that provided by the standard JDK (More info available at http://commons.apache.org)

OSGeo GeoTools

Provides spatial object model and fundamental geometric functions, which are used by DDF spatial criteria searches. More information can be found at http://geotools.org/.

Joda Time

Provides an enhanced, easier to use version of Java date and time classes. More information can be found at http://joda-time.sourceforge.net.

For a full list of dependencies, refer to the Software Version Description Document (SVDD).

Recommended Hardware

Because of its modular nature, DDF may require a few or many system resources, depending on which bundles and features are deployed. In general, DDF will take advantage of available memory and processors. A 64-bit JVM is required, and a typical installation is performed on a single machine with 16GB of memory and eight processor cores.

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.

Security Framework

The DDF Security Framework utilizes Apache Shiro as the underlying security framework. The classes mentioned in this section will 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.

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 (e.g., 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.

Implementations of Subject:

Classname	Description
ddf.security.impl.SubjectImpl	Extends org.apache.shiro.subject.support.DelegatingSubject

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.

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.

Authentication Tokens

`org.apache.shiro.authc.AuthenticationToken`

Authentication Tokens are used to verify authentication of a user when creating a subject. A common use-case is when a user is logging directly in to the DDF framework.

Classname	Description
ddf.security.service.impl.cas.CasAuthenticationToken	This Authentication Token is used for authenticating a user that has logged in with CAS. It takes in a proxy ticket which can be validated on the CAS server.

Realms

Authenticating Realms

`org.apache.shiro.realm.AuthenticatingRealm` Authenticating Realms are used to authenticate an incoming authentication token and create a Subject on successfully authentication.

Implementations of Authenticating Realms that come with DDF:

Classname	Description
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 a Subject.

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 AuthZ 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).

Other implementations of the Security API that come with DDF:

Classname	Description
org.codice.ddf.platform.filter.delegate.DelegateServletFilter	The DelegateServletFilter detects any servlet filters that have been exposed as OSGi services and places them in-order in front of any servlet or web application running on the container.
org.codice.ddf.security.filter.websso.WebSSOFilter	This filter serves as the main security filter that works in conjunction with a number of handlers to protect a variety of contexts, each using different authentication schemes and policies.
org.codice.ddf.security.handler.saml.SAMLAssertionHandler	This handler is executed by the WebSSOFilter for any contexts configured to use it. This handler should always come first when configured in the Web Context Policy Manager, as it provides a caching capability to web contexts that use it. The handler will first check for the existence of a cookie named "org.codice.websso.saml.token" to extract a Base64 + deflate SAML assertion from the request. If an assertion is found it will be converted to a SecurityToken. Failing that, the handler will check for a JSESSIONID cookie that might relate to a current SSO session with the container. If the JSESSIONID is valid, the SecurityToken will be retrieved from the cache in the LoginFilter.

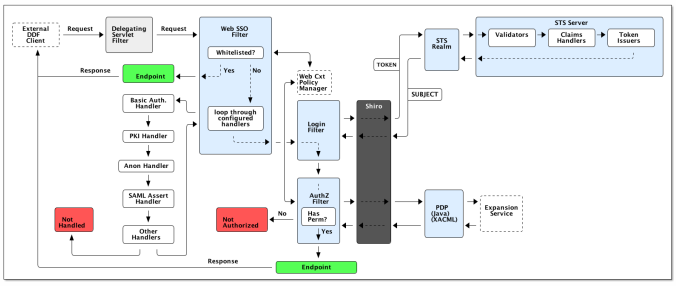
Classname	Description
org.codice.ddf.security.handler.basic.BasicAuthenticationHandler	Checks for basic authentication credentials in the http request header. If they exist, they are retrieved and passed to the LoginFilter for exchange.
org.codice.ddf.security.handler.pki.PKIHandler	Handler for PKI based authentication. X509 chain will be extracted from the HTTP request and converted to a BinarySecurityToken.
org.codice.ddf.security.handler.anonymous.AnonymousHandler	Handler that allows anonymous user access via a guest user account. The guest account credentials are configured via the org.codice.ddf.security.claims.anonymous.AnonymousClaimsHandler. The AnonymousHandler also checks for the existence of basic auth credentials or PKI credentials that might be able to override the use of the anonymous user.
org.codice.ddf.security.filter.login.LoginFilter	This filter runs immediately after the WebSSOFilter and exchanges any authentication information 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 user via Shiro.
org.apache.shiro.realm.AuthenticatingRealm	This is an abstract authenticating realm that exchanges an org.apache.shiro.authc.AuthenticationToken for a ddf.security.Subject in the form of an org.apache.shiro.authc.AuthenticationInfo
ddf.security.realm.sts.StsRealm	This realm is an implementation of org.apache.shiro.realm.AuthenticatingRealm and connects to an STS (configurable) to exchange the authentication token for a 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 focus on making the decision.
ddf.security.pdp.realm.XACMLRealm	This realm delegates the authorization decision to a XACML-based Policy Decision Point (PDP) backend. It creates a XACML 3.0 request and looks on the OSGi framework for any service implementing ddf.security.pdp.api.PolicyDecisionPoint.
ddf.security.pdp.realm.SimpleAuthZRealm	This realm performs the authorization decision without delegating to an external service. 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. Note that no PolicyExtension implementations are provided out of the box.

Classname	Description
org.codice.ddf.security.validator.*	A number of STS validators are provided for X.509 (BinarySecurityToken), UsernameToken, SAML Assertion, and DDF custom tokens. The DDF custom tokens are all BinarySecurityTokens that may have PKI or username/password information as well as an authentication realm (correlates to JAAS realms installed in the container). The authentication realm allows an administrator to restrict which services they wish to use to authenticate users. For example: installing the security-sts-ldaplogin feature will enable a JAAS realm with the name "ldap". This realm can then be specified on any context using the Web Context Policy Manager. That realm selection is then passed via the token sent to the STS to determine which validator to use.

WARNING

An update was made to the SAML Assertion Handler to pass SAML assertions via headers instead of cookies. Cookies are still accepted and processed to maintain legacy federation compatibility, but only headers are used when federating out. This means that it is still possible to federate and pass a machine's identity, but federation of a user's identity will ONLY work when federating from 2.7.x to 2.8.x+ or between 2.8.x+ and 2.8.x+.

Securing REST



The delegating servlet filter is topmost filter for all web contexts. It loads in all security filters. The first filter used is the Web SSO filter. It 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 goes directly to the desired endpoint. If the context is not on the whitelist, the filter will attempt to get a 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 will return 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. The Login Filter receives a token and return a subject. To retrieve the subject, the token is sent through Shiro to the STS Realm where the token will be exchanged for a SAML assertion through a SOAP call to an STS server. If the Subject is returned, the request moves to the Authorization Filter to check permissions on the user. If the user has the correct permissions to access that web context, the request is allowed to hit

the endpoint.

Encryption Service

The encryption service and encryption command, which are based on Jasypt, provide an easy way for developers to add encryption capabilities to DDF.

Encryption Command

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

Below is an example of the security:encrypt command used to encrypt the plain text "myPasswordToEncrypt". The output, `bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=`, is the encrypted value.

```
ddf@local>security:encrypt myPasswordToEncrypt

bR9mJpDVo8bTRwqGwIFxHJ5yFJzatKwjXjIo/8USWm8=
```

Filtering

Metacard filtering is performed in a Post Query plugin that occurs after a query has been performed.

How Filtering Works

Each metacard result will contain security attributes that are pulled from the metadata record after being processed by a PostQueryPlugin (Not provided! You must create your own plugin for your specific metadata!) that populates this attribute. 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 installed PDP (features:install security-pdp-java OR features:install security-pdp-xacml). The PDP that is being used returns a decision, and the metacard will either be 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 PostQueryPlugin that reads the metadata being returned and populates the metacard's security attribute. If the subject permissions are missing during filtering, all resources will be 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 will be converted to a KeyValuePermission object. These permission objects will be implied against the permission object generated from the metacard record. In this particular case, the metacard might be allowed if the policy is configured appropriately because all of the permissions line up correctly.

Filter Policies

The procedure for setting up a policy differs depending on which PDP implementation installed. The `security-pdp-java` implementation is the simplest PDP to use, so it will be covered here.

1. Open the https://localhost:8993/system/console/configuration.*
2. Click on the Authz Security Settings configuration.
3. Add any roles that are allowed to access protected services.

4. Add any SOAP actions that are not to be protected by the PDP.

5. Add any attribute mappings necessary to map between subject claims and metacard values.

    a. For example, the above example would require two Match All mappings of claim1=entry1 and claim2=entry2

    b. Match One mappings would contain claim3=entry3 and claim4=entry4.

NOTE	See the Security PDP AuthZ Realm (Java PDP) section of this documentation for a description of the configuration page.

With the security-pdp-java feature configured in this way, the above Metacard would be displayed to the user.

The XACML PDP is explained in more detail in the XACML Policy Decision Point (PDP) section of this documentation. It the administrator's responsibility to write a XACML policy capable of returning the correct response message. The Java-based PDP should perform adequately in most situations. It is possible to install the security-pdp-java and security-pdp-xacml features at the same time. The system could be configured in this way in order to allow the Java PDP to handle most cases and only have XACML policies to handle more complex situations than what the Java PDP is designed for. Keep in mind that this would be a very complex configuration with both PDPs installed, and this should only be performed if you understand the complex details.

Filter a New Type of Metacard

To enable filtering on a new type of record, implement a PostQueryPlugin that is able to read the string metadata contained within the metacard record. The plugin must set the security attribute to a map of list of values extracted from the metacard. Note that in DDF, there is no default plugin that populates the security attribute on the metacard. A plugin must be created to populate these fields in order for filtering to work correctly.

Security Token Service

The Security Token Service (STS) is a service running in DDF that allows clients to request SAML v2.0 assertions. These assertions are then used to authenticate a client allowing them to issue other requests, such as ingests or queries to DDF services.

The STS is an extension of Apache CXF-STS. It is a SOAP web service that utilizes WS-Security policies. The generated SAML assertions contain attributes about a user and is used by the Policy Enforcement Point (PEP) in the secure endpoints. Specific configuration details on the bundles that come with DDF can be found on the Security STS application page. This page details all of the STS components that come out of the box with DDF, along with configuration options, installation help, and which services they import and export.

The STS server contains validators, claim handlers, and token issuers to process incoming requests.

When a request is received, the validators first ensure that it is valid. The validators verifies authentication against configured services, such as LDAP, DIAS, PKI. If the request is found to be invalid, the process ends and an error is returned. Next, the claims handlers determine how to handle the request, adding user attributes or properties as configured. The token issuer creates a SAML 2.0 assertion and associates it with the subject. The STS server sends an assertion back to the requestor, which is used in both SOAP and REST cases.

Using the Security Token Service (STS)

Once installed, the STS can be used to request SAML v2.0 assertions via a SOAP web service request. Out of the box, the STS supports authentication from existing SAML tokens, CAS proxy tickets, username/password, and x509 certificates. It also supports retrieving claims using LDAP.

Standalone Installation

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

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

STS Claims Handlers

Claims handlers are classes that convert the incoming user credentials into a set of attribute claims that will be populated in the SAML assertion. An example in action would be the LDAPClaimsHandler that 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.

Add a Custom Claims Handler

Description

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.

Motivation

A developer may want to add a custom claims handler to retrieve attributes from an external attribute store.

Steps

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, a developer may have to add the root CA of the external site to the DDF trustStore and add a valid certificate into the DDF keyStore. Doing so will allow the SSL to encrypt messages that will be accepted by the external service. For more information on certificates, refer to the Configuring a Java Keystore for Secure Communications page.

STS WS-Trust WSDL Document

NOTE This XML file is found inside of the STS bundle and is named ws-trust-1.4-service.wsdl.

```
<?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://localhost:8181/services/SecurityTokenService
"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Example Request and Responses for a SAML Assertion

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.

Most endpoints that have been used in DDF require the X.509 PublicKey SAML assertion.

BinarySecurityToken (CAS) SAML Security Token Request/Response

BinarySecurityToken (CAS) Sample Request/Response

Request

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:A
ddress>
                </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>
```

Response

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:Dige
stValue>
                            </ds:Reference>
                        </ds:SignedInfo>
                        <ds:SignatureValue>h+NvkgXGdQtca3/eKebhAKgG38tHp3i2n5uLLy8xXX
Ig02qyKgEP0FCowp2LiYlsQU9YjKfSwCUbH3WR6jhbAv9zj29CE+ePfEny7MeXvgNl3wId+vcHqti/DGGhhgtO2Mb
x/tyX1BhHQUwKRlcHajxHeecwmvV7D85NMdV48tI=</ds:SignatureValue>
                        <ds:KeyInfo>
                            <ds:X509Data>
                                <ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9
w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMH
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>MIIC5DCCAk2gAwIBAgIJAKj7R
OPHjo1yMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYDVQQGEwJVUzEQ
```

```
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/SecurityToke
nService</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:a
c:classes:unspecified</saml2:AuthnContextClassRef>
                        </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</w
sa: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>
```

UsernameToken Bearer SAML Security Token Request/Response

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.

Request

Explanation

- WS-Addressing header with Action, To, and Message ID
- Valid, non-expired timestamp
- Username Token containing a username and password that the STS will authenticate
- 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.

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>
```

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 will verify 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.

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:Dige
stValue>
                            </ds:Reference>
                        </ds:SignedInfo>
                        <ds:SignatureValue>X5Kzd54PrKIlGVV2XxzCmWFRzHRoybF7hU6zxbEhSL
MR0AWS9R7Me3epq91XqeOwvIDDbwmE/oJNC7vI0fIw/rqXkx4aZsY5a5nbAs7f+aXF9TGdk82x2eNhNGYpViq0YZJ
fsJ5WSyMtG8w5nRekmDMy9oTLsHG+Y/OhJDEwq58=</ds:SignatureValue>
                        <ds:KeyInfo>
                            <ds:X509Data>
                                <ds:X509Certificate>MIIDmjCCAwOgAwIBAgIBBDANBgkqhkiG9
w0BAQQFADB1MQswCQYDVQQGEwJVUzEQMA4GA1UECBMH
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:a
c:classes:unspecified</saml2:AuthnContextClassRef>
                            </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:Addre
ss>
                    </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>
```

X.509 PublicKey SAML Security Token Request/Response

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 will specify 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.

Request

Explanation 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 will be 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.

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>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcNAQEFBQAwXDELMAk
GA1UEBhMCVVMxGDAWBgNVBAoT
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>
```

Response

Explanation 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 will verify 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.

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>MIIFHTCCBAWgAwIBAgICJe8wDQYJKoZIhvcNAQEFBQ
AwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT
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>MIIFGDCCBACgAwIBAgICJe0wDQYJKoZIhvcN
AQEFBQAwXDELMAkGA1UEBhMCVVMxGDAWBgNVBAoT
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:classe
s:unspecified</saml2:AuthnContextClassRef>
                     </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>
```

Authz filter

The Authz filter determines authentication of the subject properties by calling the policy manager. It then sends the subject through Shiro to a SimpleAuthZ Realm

NOTE	The SimpleAuthz realm can be replaced by a XACML implementation of the same functionality.

The SimpleAuthZ realm will call the expansion service to translate any properties associated with the subject and returns a boolean value if authorization is met. Upon receiving the correct value, the AuthZ filter allows access to the endpoint.

XACML Policy Decision Point (PDP)

After unzipping the DDF distribution, place the desired XACML policy in the <distribution root>/etc/pdp/policies directory. This is the directory in which the PDP will look for XACML policies every 60 seconds. A sample XACML policy is located at the end of this page. Information on specific bundle configurations and names can be found on the Security PDP application page.

Creating a Policy

This document assumes familiarity with the XACML schema and does not go into detail on the XACML language. There are some DDF-specific items that need to be considered, when creating a policy, to be compatible with the XACMLRealm. 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. Generally targets are geared toward the actions that are set in the request.

Actions

For DDF, these actions are populated by various components in the security API. The actions and their population location are identified in the following table.

Operation	Action-id Value	Component Setting the action	Description
Filtering	filter	security-pdp-xacmlrealm	When performing any filtering, the XACMLRealm will set the action-id to "filter".
Service	<SOAPAction>	security-pep-interceptor	If the PEP Interceptor is added to any SOAP-based web services for service authorization, the action-id will be the SOAPAction of the incoming request. This allows the XACML policy to have specific rules for individual services within the system.

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

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

Attributes

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

Subject

The attributes for the subject are obtained from the SAML claims and populated within the XACMLRealm 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.

Resource

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

Example Requests and Responses

The following items are a sample request, response, and the corresponding policy. For the XACML PDP, the request is made by the XACML realm (security-pdp-xacmlrealm), passed to the XACML processing engine (security-pdp-xacmlprocessor), which reads the policy and outputs a response.

Policy

This is the sample policy that was used for the following sample request and responses. The policy was made to handle the following actions: filter, query, and LocalSiteName. The filter action is used to compare subject's SUBJECT_ACCESS attributes to metacard's RESOURCE_ACCESS attributes. The query and LocalSiteName actions differ, as they are used to perform service authorization. For a query, the user must be associated with the country code ATA (Antarctica), and a LocalSiteName action can be performed by anyone.

Policy

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="xpath-target-
single-req" RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-
algorithm:permit-overrides" Version="1.0">
    <PolicyDefaults>
        <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
    </PolicyDefaults>
    <Target>
        <AnyOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
>filter</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </Match>
            </AllOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
>query</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </Match>
            </AllOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
>LocalSiteName</AttributeValue>
                    <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                </Match>
            </AllOf>
        </AnyOf>
    </Target>
    <Rule Effect="Permit" RuleId="permit-filter">
        <Target>
            <AnyOf>
                <AllOf>
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="
```

```
http://www.w3.org/2001/XMLSchema#string">filter</AttributeValue>
                        <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                    </Match>
                </AllOf>
            </AnyOf>
        </Target>
        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-subset">
                <AttributeDesignator AttributeId="RESOURCE_ACCESS" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
                <AttributeDesignator AttributeId="SUBJECT_ACCESS" Category=
"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
            </Apply>
        </Condition>
    </Rule>
    <Rule Effect="Permit" RuleId="permit-action">
        <Target>
            <AnyOf>
                <AllOf>
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="
http://www.w3.org/2001/XMLSchema#string">ATA</AttributeValue>
                        <AttributeDesignator AttributeId=
"http://www.opm.gov/feddata/CountryOfCitizenship" Category=
"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
                    </Match>
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="
http://www.w3.org/2001/XMLSchema#string">query</AttributeValue>
                        <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
                    </Match>
                </AllOf>
                <AllOf>
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="
http://www.w3.org/2001/XMLSchema#string">LocalSiteName</AttributeValue>
                        <AttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:1.0:action:action-id" Category=
"urn:oasis:names:tc:xacml:3.0:attribute-category:action" DataType=
```

```
 "http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
                    </Match>
                </AllOf>
            </AnyOf>
        </Target>
    </Rule>
    <Rule Effect="Deny" RuleId="deny-read"/>
</Policy>
```

Service Authorization

Allowed Query

Request

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="
false" CombinedDecision="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
query</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
users</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
admin</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
B</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" IncludeInResult=
"false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
testuser1</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" IncludeInResult="false
">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/feddata/CountryOfCitizenship"
```

```
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
ATA</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Response

```
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

Denied Query

Request

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="
false" CombinedDecision="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
query</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test User
USA</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
users</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
admin</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
B</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" IncludeInResult=
"false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
testuser1</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" IncludeInResult="false
">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/feddata/CountryOfCitizenship"
```

```
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
USA</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Response

```
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

Metacard Authorization

Subject Permitted

All of the resource's RESOURCE_ACCESS attributes were matched with the Subject's SUBJECT_ACCESS attributes.

Request

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="
false" CombinedDecision="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
filter</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
users</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
admin</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" IncludeInResult=
"false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
testuser1</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" IncludeInResult="false
">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
B</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/feddata/CountryOfCitizenship"
```

```
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
ATA</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
        <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Response

```
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

Subject Denied

The resource had an additional RESOURCE_ACCESS attribute 'C' that the subject did not have.

Request

```
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="
false" CombinedDecision="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
filter</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
users</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
admin</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" IncludeInResult="false
">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Test
User</AttributeValue>
        </Attribute>
        <Attribute AttributeId=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" IncludeInResult=
"false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
testuser1</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="SUBJECT_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
B</AttributeValue>
        </Attribute>
        <Attribute AttributeId="http://www.opm.gov/feddata/CountryOfCitizenship"
```

```
IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
ATA</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
        <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
A</AttributeValue>
        </Attribute>
        <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
B</AttributeValue>
        </Attribute>
        <Attribute AttributeId="RESOURCE_ACCESS" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
C</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Response

```
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Deny</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

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 attributes 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.

Each instance of the expansion service consists of a collection of rule sets. Each rule set consists of a key value and its associated set of rules. Callers of the expansion service provide a key and an original value to be expanded. The expansion service then looks up the set of rules for the specified key. The expansion service then cumulatively applies each of the rules in the set starting with the original value, with the resulting set of values being returned to the caller.

Key (Attribute)	Rules (original new)	
key1	value1	replacement1
	value2	replacement2
	value3	replacement3
key2	value1	replacement1
	value2	replacement2

The examples below use the following collection of rule sets:

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

Note that the rules listed for each key are processed in order, so they may build upon each other, i.e., a new value from the new replacement string may be expanded by a subsequent rule.

Instances and Configuration

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

Property Name	Value	Description
mapping	security.user.attribute.mapping	This instance is configured with rules that expand the user's attribute values for security checking.
mapping	security.metacard.attribute.mapping	This instance is configured with rules that expand the metacard's security attributes before comparing with the user's attributes.

Each instance of the expansion service can be configured using a configuration file. The configuration file can have three different types of lines: * comments - any line prefixed with the # character is ignored as a comment (for readability, blank lines are also ignored) * attribute separator - a line starting with separator= defines the attribute separator string. * rule - all other lines are assumed to be rules defined in a string format `<key>:<original value>:<new value>`

The following configuration file defines the rules shown above in the example table (using the space as a separator):

```
# This defines the separator that will be used when the expansion string contains
multiple
# values - each will be separated by this string. The expanded string will be split at
the
# separator string and each resulting attributed added to the attribute set (duplicates
are
# suppressed). No value indicates the defualt value of ' ' (space).
separator=

# The following rules define the attribute expansion to be performed. The rules are of
the
# form:
#       <attribute name>:<original value>:<expanded value>
# The rules are ordered, so replacements from the first rules may be found in the
original
# values of subsequent rules.
Location:Goodyear:Goodyear AZ
Location:AZ:AZ USA
Location:CA:CA USA
Title:VP-Sales:VP-Sales VP Sales
Title:VP-Engineering:VP-Engineering VP Engineering
```

Expansion Commands

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

Expansion Commands

security:expand	security:expansions

Command Descriptions

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

Expansion Command Examples and Explanation

`security:expansions`

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

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

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

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

`security:expand`

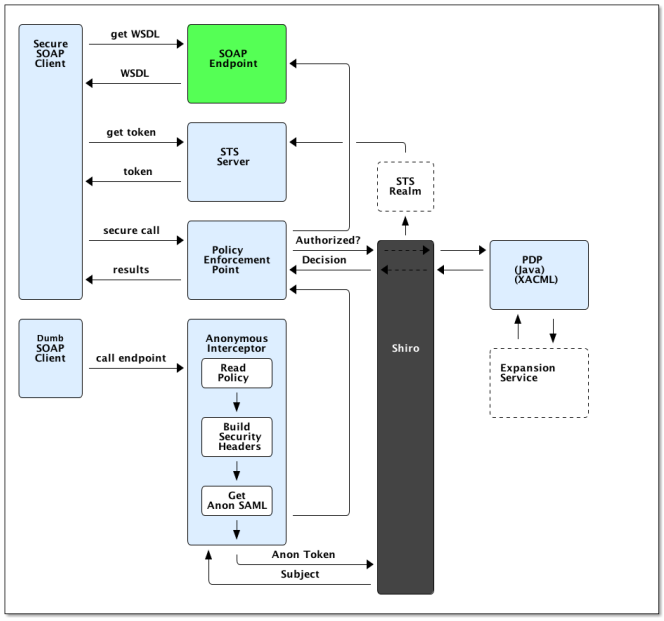
The security:expand command runs the expansion service on the provided data. It takes an attribute and an original value, expands the original value using the current expansion service and rule set and dumps the results. For the rule set shown above, the expand command produces the following results:

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

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

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

Securing SOAP



SOAP Secure Client

When calling to an endpoint from a SOAP secure client, it first requests the WSDL from the endpoint and the SOAP endpoint returns the WSDL. The client then calls to STS for authentication token to proceed. If the client receives the token, it makes a secure call to the endpoint and receives results.

Dumb SOAP Client

If calling endpoint from a non-secure client, at the point the of the initial call, the Anonymous Interceptor catches the request and prepares it to be accepted by the endpoint. First, the interceptor reads the configured policy, builds a security header, and gets an anonymous SAML assertion. Using this, it makes a `getSubject` call which is sent through Shiro to the STS realm. Upon success, the STS realm returns the subject and the call is made to the endpoint.

Developing DDF Applications

The DDF applications are comprised of components, packaged as Karaf features, which are collections of OSGi bundles. These features can be installed/uninstalled using the Web Console or command line console. DDF applications also consist of one or more OSGi bundles and, possibly, supplemental external files. These applications are packaged as Karaf KAR files for easy download and installation. These applications can be stored on a file system or a Maven repository.

A KAR file is a Karaf-specific archive format (Karaf ARchive). It is a jar file that contains a feature descriptor file and one or more OSGi bundle jar files. The feature descriptor file identifies the application's name, the set of bundles that need to be installed, and any dependencies on other features that may need to be installed.

Creating a KAR File

The recommended method for creating a KAR file is to use the `features-maven-plugin`, which has a `create-kar` goal (available as of Karaf v2.2.5, which DDF 2.X is based upon). This goal reads all of the features specified in the features descriptor file. For each feature in this file, it resolves the bundles defined in the feature. All bundles are then packaged into the KAR archive. An example of using the create-kar goal is shown below:

create-kar goal

```
<plugin>
<groupId>org.apache.karaf.tooling</groupId>
<artifactId>features-maven-plugin</artifactId>
<version>2.2.5</version>
<executions>
<execution>
<id>create-kar</id>
<goals>
<goal>create-kar</goal>
</goals>
<configuration>
<descriptors>
<!-- Add any other <descriptor> that the features file may reference here -->
</descriptors>
<!--
Workaround to prevent the target/classes/features.xml file from being included in the
kar file since features.xml already included in kar's repository directory tree.
Otherwise, features.xml would appear twice in the kar file, hence installing the
same feature twice.
Refer to Karaf forum posting at http://karaf.922171.n3.nabble.com/Duplicate-feature-
repository-entry-using-archive-kar-to-build-deployable-applications-td3650850.html
-->
<resourcesDir>/Users/jlcsmith/source/2.8.x/ddf/distribution/docs/target/doesNotExist</res
ourcesDir>

<!--
Location of the features.xml file. If it references properties that need to be filtered,
e.g., 2.8.2, it will need to be
filtered by the maven-resources-plugin.
-->
<featuresFile>/Users/jlcsmith/source/2.8.x/ddf/distribution/docs/target/classes/features.
xml</featuresFile>

<!-- Name of the kar file (.kar extension added by default). If not specified, defaults
to docs-2.8.2 -->
<finalName>ddf-ifis-2.8.2</finalName>
</configuration>
</execution>
    </executions>
</plugin>
```

Examples of how KAR files are created for DDF components can be found in the DDF source code under the ddf/distribution/ddf-kars directory.

The .kar file generated should be deployed to the application author's maven repository. The URL to

the application's KAR file in this Maven repository should be the installation URL that is used.

Including Data Files in a KAR File

The developer may need to include data or configuration file(s) in a KAR file. An example of this is a properties file for the JDBC connection properties of a catalog provider.

It is recommended that: * Any data/configuration files be placed under the src/main/resources directory of the maven project. Sub-directories under src/main/resources can be used, e.g., etc/security. * The Maven project's pom file should be updated to attach each data/configuration file as an artifact (using the build-helper-maven-plugin). * Add each data/configuration file to the KAR file using the <configfile> tag in the KAR's features.xml file.

Installing a KAR File

When the user downloads an application by clicking on the **Installation** link, the application's KAR file is downloaded. This KAR file should be placed in the `<DDF_INSTALL_DIR>/deploy` directory of the running DDF instance. DDF then detects that a file with a .kar file extension has been placed in this monitored directory, unzips the KAR file into the `<DDF_INSTALL_DIR>/system` directory, and installs the bundle(s) listed in the KAR file's feature descriptor file. The user can then go to the Web Console's Features tab and verify the new feature(s) is installed.

OGC Filter

OGC Filter

An OGC Filter is a Open Geospatial Consortium (OGC) standard that describes a query expression in terms of XML and Key-Value Pairs (KVP).

DDF originally had a custom query representation that some found difficult to understand and implement. In switching to a well-known standard like the OGC Filter, developers benefit from various third party products and third party documentation, as well as any previous experience with the standard. 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.

OGC filter in the DDF Catalog

The DDF Catalog Framework uses the implementation provided by Geotools, which provides a Java representation of the standard.

Geotools originally provided standard Java classes for the OGC Filter Encoding 1.0, under the package name `org.opengis.filter`, which is where `org.opengis.filter.Filter` is located. Java developers should

use the Java objects exclusively to complete query tasks, rather than parsing or viewing the XML representation.

Utilities

Each DDF Application is located in its own code repository. In addition to the applications, there are other utilities that are available in other code repositories on Github. These utilities are deployed into Nexus for easier accessibility.

DDF-libs

DDF-libs is a repository for library modules in DDF. Typically the modules in this repository are re-usable across different components of DDF.

DDF Load Balancer

Provides utility that allows incoming traffic to be distributed over multiple instances of DDF, via HTTP or HTTPS

Contained within DDF is a Load Balancer utility that allows incoming traffic to be distributed over multiple instances of DDF. The DDF Load Balancer supports two protocols: HTTP and HTTPS. The Load Balancer can be configured to run one protocol or both at the same time. The DDF Load Balancer has been configured to utilize a "Round Robin" algorithm to distribute transactions. The load balancer is also equipped with a failover mechanism. When the load balancer attempts to access a server that is non-functional, it will receive an exception and move on to the next server on the list to complete the transaction. The action will try to be replayed on every server once before failing back to the client.

Set up the DDF Load Balancer

The main method for installing the DDF Load Balancer is to install the application (.kar) file into the hot deploy folder of a full DDF distribution.

Prerequisites

Before the DDF Load Balancer can be installed:

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

Install

Complete the following procedure to install the DDF Load Balancer.

1. Download the application (.kar) file from the artifacts repo (http://artifacts.codice.org/).

2. Copy the KAR file into the <INSTALL_DIRECTORY>/deploy folder of a currently running DDF distribution.

3. Uninstall the included jetty feature. **Note: This is due to a bug in the current version of jetty being delivered with DDF, this step will be removed once that version is updated**.

```
features:uninstall jetty
```

Verify

1. Verify all of the Load Balancer's appropriate features have been successfully installed.

   DDF Load Balancer installed features

   ```
   ddf@local>features:list | grep -i loadbalancer-app
   [installed  ] [1.0.0               ] codice-load-balancer      loadbalancer-app-1.0.0
   Load Balancer
   [installed  ] [7.6.12.v20130726    ] loadbalancer-jetty        loadbalancer-app-1.0.0
   Provide Jetty engine support
   ```

2. Verify the DDF Load Balancer bundles are Active.

DDF Load Balancer active bundles

```
[ 223] [Active     ] [            ] [       ] [   50] camel-http (2.12.1)
[ 224] [Active     ] [            ] [       ] [   50] camel-jetty (2.12.1)
[ 258] [Active     ] [            ] [       ] [   80] Jetty :: Utilities
(7.6.12.v20130726)
[ 259] [Active     ] [            ] [       ] [   80] Jetty :: IO Utility
(7.6.12.v20130726)
[ 260] [Active     ] [            ] [       ] [   80] Jetty :: Http Utility
(7.6.12.v20130726)
[ 261] [Active     ] [            ] [       ] [   80] Jetty :: Asynchronous HTTP Client
(7.6.12.v20130726)
[ 262] [Active     ] [            ] [       ] [   80] Jetty :: Continuation
(7.6.12.v20130726)
[ 263] [Active     ] [            ] [       ] [   80] Jetty :: JMX Management
(7.6.12.v20130726)
[ 264] [Active     ] [            ] [       ] [   80] Jetty :: Server Core
(7.6.12.v20130726)
[ 265] [Active     ] [            ] [       ] [   80] Jetty :: Security
(7.6.12.v20130726)
[ 266] [Active     ] [            ] [       ] [   80] Jetty :: Servlet Handling
(7.6.12.v20130726)
[ 267] [Active     ] [            ] [       ] [   80] Jetty :: Utility Servlets and
Filters (7.6.12.v20130726)
[ 268] [Active     ] [            ] [       ] [   80] Jetty :: XML utilities
(7.6.12.v20130726)
[ 269] [Active     ] [            ] [       ] [   80] Jetty :: Webapp Application
Support (7.6.12.v20130726)
[ 270] [Active     ] [            ] [       ] [   80] Jetty :: JNDI Naming
(7.6.12.v20130726)
[ 271] [Active     ] [            ] [       ] [   80] Jetty :: Plus (7.6.12.v20130726)
[ 272] [Active     ] [            ] [       ] [   80] Jetty :: Websocket
(7.6.12.v20130726)
[ 273] [Active     ] [Created     ] [       ] [   80] Codice :: Loadbalancer :: Camel
(1.0.0)
```

Uninstall

WARNING	It is very important to save the KAR file for the application prior to an uninstall so that the uninstall can be reverted if necessary.

Complete the following procedure to uninstall the DDF Load Balancer.

1. Delete the KAR file (loadbalancer-app-X.Y.kar) from the <INSTALL_DIRECTORY>/deploy directory.
2. Re-install the jetty feature.

```
features:install jetty
```

3. Restart DDF to ensure that all of the Jetty bundles are refreshed properly.

Configure the DDF Load Balancer

The DDF Load Balancer can be configured to allow multiple DDF nodes to be balanced. It can also be configured with a port on which to accept connections. Configurations differ slightly between the HTTP- and HTTPS-based load balancers. All configurations are dynamic in that configuration settings are immediately applied, and it is not necessary to restart DDF.

Configure the HTTP Load Balancer

Complete the following procedure to access the load balancer configuration.

1. Click the Configuration tab in the DDF management console.
2. Scroll down to the configuration entry that is labeled Platform **HTTP Load Balancer**. The configuration for the load balancer contains two fields: **Load Balancer Port** and **IP Address and Port**.
3. In the **Load Balancer Port** field, enter the port number to be accessed when reaching other systems.
4. In the IP Address and Port field, enter a comma-delimited list of IP addresses and ports for each DDF node to be balanced. The format for IP address and port is <IP_ADDRESS>:<PORT> (e.g., 192.168.1.123:8181,192.168.1.22:8181).
5. Select the **Save** button when all configuration have been added.

At this point, the load balancer is reset and ready to accept requests. These configurations can be updated at any time without starting the host DDF instance.

Configure the HTTPS Load Balancer

It is possible to run the HTTPS load balancer by itself or run it in parallel with the HTTP load balancer. The HTTPS load balancer utilizes the centralized SSL configurations within DDF, along with the load balancer configurations. Complete the following procedure to configure the HTTPS load balancer.

1. Find the SSL configurations and verify that the values are correct.
2. Click the Configuration tab in the DDF management console.
3. Scroll down to the configuration entry that is labeled **Pax Web Runtime**.
4. Select **Pax Web Runtime**.

5. Ensure that the displayed settings match what is configured in the DDF nodes to be balanced.
6. Save the updated settings or close the window, as applicable.
7. In the configuration table, scroll down to the configuration entry labeled **Platform HTTPS Load Balancer**.
8. The configuration for the load balancer contains three fields: **Load Balancer Host**, **Load Balancer Port**, and **IP Address and Port**.
9. In the **Load Balancer Host** field, enter the host name or IP address to be used for the host load balancer machine.
10. In the **Load Balancer Port** field, enter the port number to be accessed on the load balancer to reach the other systems.
11. In the **IP Address and Port** field, enter a comma delimited list of IP address and port for each DDF node that will be balanced. The format for IP address and port is <IP_ADDRESS>:<PORT> (e.g. 192.168.1.123:8993,192.168.1.22:8993)
12. Select the **Save** button when all configuration have been added.

Since SSL requests will be coming from a client into the load balancer, it is essential that the nodes being balanced have the same security policy and settings. The client has no idea which DDF server it will be connecting with behind the load balancer. The client is responsible for connecting securely with the load balancer, and the load balancer is responsible for connecting securely and consistently with all DDF nodes.

WARNING	The DDF Load Balancer cannot run on the same port as the DDF Web Console or other web services. If you would like the load balancer to run on this port, change the web console port to a different port number. This configuration parameter can be found in the **Pax Web Runtime configuration**.

DDF STOMP

The DDF STOMP application allows query subscription messages to be sent to the DDF server via STOMP protocol.

Subscription query messages are defined in JSON format following a defined schema. These messages allow for the management of subscriptions using create, time to live (TTL), update, and delete functions. Catalog queries within the message are defined in CQL format. Results are sent to a STOMP-based topic, which can be subscribed to via a STOMP-based client. Content results will be delivered over time as the subscription query matches incoming data published to the DDF catalog.

STOMP

STOMP is a streaming text-based messaging protocol that supports the delivery of messages as well as

publish and subscribe. STOMP mimics HTTP and can utilize TCP-IP, thus making it compatible with many different programming languages. STOMP is very simple to implement and can easily be tested. For more information, refer to http://stomp.github.io/.

Common Query Language (CQL)

Common Query Language or CQL is a query language the the OGC has chosen for expressing data filtering. The power of CQL is its strong integration into GeoTools, which helps to represent complex queries as text strings. For more information, refer to http://docs.geotools.org/latest/userguide/library/cql/index.html

Publish and Subscribe Query Subscription Message

Subscriptions are stateful and will survive when the server is restarted. Subscription messages are specified in a JSON format. The schema section specifies the values that construct this message, along with the defaults. The sections below describe the meaning of the message values.

Subscription Identifier (`subscriptionId`)

The subscription identifier is a unique string that is provided by the query subscription requester. The preferred value should be a generated UUID. Example: “`subscriptionId” : "faf4e8493h389fh4398f3h0040`"

Action (`action`)

The action value tells the system what action should take place. The following actions are available: `CREATE:` Creates a new subscription `UPDATE:` Updates an existing subscription (requires subscriptionId) `DELETE:` Deletes an existing subscription (requires subscritionId) Example: `"action" : "CREATE"`

Subscription Time to Live (`subscriptionTtlType` and `subscriptionTtl`)

A time to live value can be set on a subscription by providing values for the TTL type and TTL. The subscriptionTtlType value describes the type of TTL that will be used. The following values are available for `subscriptionTtlType`:

- `MILLISECONDS`
- `SECONDS`
- `MINUTES`
- `HOURS`
- `MONTHS`
- `YEARS`

The subscriptionTtl value is an integer value that specifies the quantity of the subscriptionTtlType. For example, a value of 60 for subscriptionTtl along with a value of 'HOURS' for subscriptionTtlType tells the system that the subscription should last for 60 hours. If a value of 0, -9, or no value is set for

subscriptionTtl, the time to live will be infinite. Example: “subscriptionTtlType” : “HOURS”, “subscriptionTtl” : 90

Query String (`queryString`)

The query string allows the user to specify a query that can filter targeted results. Query string values are specified in CQL format. See the CQL section for more information on using this format. Example: “queryString” : “anyText LIKE 'Red Truck'”

Sources (sources)

Source targets can be specified in the message. Sources will be passed on to all queries to retrieve the correct results from all of the correct sources. The sources value is an array. Example: `"sources" : [ "source1", "source2", "source3" ]`

Creation Date and Last Modified Date (`creationDate` and `lastModifiedDate`)

The creation date and last modified date are values that are written into the subscription by the system. The creation date specifies the date and time that the subscription was created in the system. The last modified date specifies the date and time of the last change to the subscription. The user can specify these values in the subscription message, but it will be ignored.

DDF STOMP Setup

DDF STOMP can be found at https://github.com/codice/ddf-stomp. When building DDF STOMP a KAR application file is provided. This file can be added to the DDF server by placing the file in the deploy folder. Once the application has been deployed it is best to restart DDF.

Configuration

DDF STOMP has a default configuration that it utilizes. By default, the STOMP server within DDF STOMP runs on port 61613. To change this port number, modifications must be made to activemq.xml and the DDF configuration. The activemq.xml file can be found in the etc/ directory of DDF. Open the file and navigate to the bottom of the page. An entry for transportConnector reads stomp://0.0.0.0:61613. The port number at the end can be changed to the desired choice. Once chosen, save the file and exit. The second half of configuring the port number and other options can be found in the DDF configuration web console. Upon selecting the Configurations tab, look for the configuration named Publish Subscribe Subscription Query Service. This configuration has the following options:

Destination Topic Name

The topic destination where query subscription messages are sent.

Subscription Topic Name

The prefix of the topic where subscription results are sent.

STOMP Host

The host name of the STOMP server.

STOMP Port

The port number of the STOMP server.

Default Max Results

The maximum number of results to return from a given query.

Default Request Timeout

The maximum time to wait before a request is timed out.

Transformer ID

ID that specifies the format in which results are produced (default: geojson). See Extending Catalog Transformers for more information. Once these configurations have been made, it is best to restart the DDF server.

Send a Subscription Message

Subscription messages can be sent to the system using a STOMP-based client. STOMP is a standardized publish subscribe messaging protocol that utilizes the HTTP protocol for sending data. Connections to the system are asynchronous. For more information, refer to the section on STOMP. To connect the STOMP client to the system, the following information is typically required: STOMP server host, STOMP port, user name, password, and topic name. The username, password, port, and topic name can be found in the publish subscribe query subscriptions configuration. Refer to the Configuration section for more information.

The Gozirra STOMP client (http://www.germane-software.com/software/Java/Gozirra/) and the Fuse Source STOMP client (https://github.com/fusesource/stompjms) have both been used successfully to test functionality.

Subscribe or Results

A STOMP-based client can be used to retrieve results from a query subscription. STOMP is a standardized publish subscribe messaging protocol that utilizes the HTTP protocol for receiving data. Connections to the system are asynchronous. For more information, refer to the section on STOMP. To connect the STOMP client to the system, the following information is typically required: STOMP server host, STOMP port, user name, password, and topic name. The username, password, port, and topic name can be found in the publish subscribe query subscriptions configuration. Refer to the Configuration section for more information. The topic name for subscription results, which is found in the configuration, is a partial name. The end of the topic name will include the subscription id. For example, if the partial topic name was “/topic/result/”, and the subscription ID for the subscription was “faf4e8493h389fh4398f3h0040”, the actual topic name that would be subscribed to is: “/topic/result/faf4e8493h389fh4398f3h0040”. This is the full topic name that would be used with your STOMP client to retrieve results.

The Gozirra STOMP client (http://www.germane-software.com/software/Java/Gozirra/) and the Fuse Source STOMP client (https://github.com/fusesource/stompjms) have both been used sucessfully to test

functionality.

Returned Messages

Return messages will be delivered back to a subscribing STOMP client. All delivered messages are returned in GeoJSON format. When a subscription is first submitted, an initial query is executed on existing data in the catalog. All results based on this query are immediately returned to the subscribing STOMP client via the defined topic. As content is added to the catalog (when these items match the query in the subscription), the content items are immediately returned to the subscribing STOMP client via the defined topic. Examples Create a new subscription that will last for 90 hours and searches for any text that is matching red car:

```
{
  "subscriptionId" : "faf4e8493h389fh4398f3h0060",
  "action" : "CREATE",
  "subscriptionTtlType" : "HOURS",
  "subscriptionTtl" : 90,
  "queryString" : "anyText LIKE 'red car'"
}
```

Update a previous subscription to last for three months instead of 90 hours:

```
{
  "subscriptionId" : "faf4e8493h389fh4398f3h0060",
  "action" : "UPDATE",
  "subscriptionTtlType" : "MONTHS",
  "subscriptionTtl" : 3,
  "queryString" : "anyText LIKE 'red car'"
}
```

Delete the previous subscription:

```
{
  "subscriptionId" : "faf4e8493h389fh4398f3h0060",
  "action" : "DELETE"
}
```

Architecture

The API is set up to utilize STOMP as the protocol for receiving subscription management messages. External third party applications will utilize STOMP to send commands for subscription management and delivery.

Subscription Message Schema

The messages sent over STOMP will be in JSON format. The messages used for subscription management utilize the following schema:

```
{
"type":"object",
"$schema": "http://json-schema.org/draft-03/schema",
"id": "http://jsonschema.net",
"required":false,
"properties":{
"queryString": {
"type":"string",
"id": "http://jsonschema.net/searchPhrase",
"required":true
},
"sources": {
"type":"array",
"id": "http://jsonschema.net/sources",
"required":false,
"items":
{
"type":"string",
"id": "http://jsonschema.net/sources/0",
"required":false
}
},

"subscriptionId": {
"type":"string",
"id": "http://jsonschema.net/subscriptionId",
"required":true
},
"action": {
"type":"string",
"id": "http://jsonschema.net/subscriptionId",
"required":true
},
"subscriptionTtl": {
"type":"number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
"subscriptionTtlType": {
"type":"number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
"creationDate": {
"type":"number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
```

```
},
"lastModifiedDate": {
"type":"number",
"id": "http://jsonschema.net/subscriptionId",
"required":false
},
 }
}
```