Security Service v.1.0.0

for the Planetary Data System

NASA Jet Propulsion Laboratory,
California Institute of Technology

10 November 2011

Table of Contents

1.1 Overview

About Security Service

The Security Service provides the authentication and authorization functions for the PDS 2010 system. The intent of this service is to control access to interfaces and services that require authentication and authorization (e.g., Monitor, Report, Registry interfaces, etc.). The functionality for this service is satisfied by the open source software package OpenDS , which is a directory service supporting the Lightweight Directory Access Protocol (LDAP).

Please send comments, change requests and bug reports to the PDS Operator at pds_operator@jpl.nasa.gov.

1.2 Release Notes

Release Notes

The purpose of this section is to provide a description of a Security Service release including any impact that the new or modified capabilities will have on the Discipline Nodes or the PDS user community. If viewing the web-based version of this document, a somewhat itemized list of changes for each release can be found on the Release Changes page.

Release 1.0.0

This release of the Security Service is a component of the integrated release 2.0.0 of the PDS 2010 System. This is an operational release of the system components to date. This release contains documentation updates.

Release 0.3.0

This release of the Security Service is a component of the integrated releases 1.1.0 and 1.2.0 of the PDS 2010 System. This release is intended as a prototype release in support of the assessment of the PDS4 standards and the system components to date. The new or modified capabilities for this release are as follows:

- Simplified the deployed solution by utilizing a single directory service to satisfy the requirements.

The liens for this release are as follows:

- Need to define a procedure for synching this list with the personnel entries in the Registry Service.
- Need to develop a solution for obtaining and passing cookies.

Release 0.2.0

This release of the Security Service is a component of the integrated release 1.0.0 of the PDS 2010 System. This release is intended as a prototype release in support of the assessment of the PDS4 standards. The new or modified capabilities for this release are as follows:

- Added support for controlling access to HTTP PUT and DELETE methods.
- Added support for controlling access based on group membership.

The liens for this release are as follows:

- Need to resolve an issue defining policy for multiple HTTP methods on the same URL.

- Need to clean up the user list and define a procedure for synching this list with the personnel entries in the Registry Service.

Release 0.1.0

This release of the Security Service is a component of the integrated release 0.1.0 of the PDS 2010 System. This release is intended as a prototype release in support of the demonstration at the Management Council Face-to-Face meeting in August 2010. This initial release of the service provides the capability to secure specific URLs for the Registry Service and force requests to those secure URLs to require authentication.

1.3 Installation

Installation

This section describes how to install the OpenDS software package. This package serve as the Security Service for the PDS 2010 system. The following topics can be found in this section:

- System Requirements
- Software Installation
- Configuration

System Requirements

The software that makes up this project consists of an open source package that is available for download and installation. The package and its release version is as follows:

- OpenDS 2.2.0

The above software package requires the following software to be installed in the target environment:

- Sun Java Standard Edition (J2SE) 1.6.X
- Apache Tomcat 6.0.X

Software Installation

1. Install Directory Server

We are choosing to install a directory server (OpenDS), within the application server so that it is accessible from other applications that require a standard LDAP interface.

- Download the package from http://www.opends.org/promoted-builds/2.2.0/ . There are a couple of options to choose from including the QuickSetup Installer or just downloading the ZIP package.
- Install the package so that it operates on the standard LDAP port 389. The ZIP package provides installation documentation. Additional documentation is available at https://docs.opends.org/2.2/page/InstallingTheDs .
- Startup the server.

Install OpenDS with QuickSetup

Launch QuickSetup Locally.

```
% $OPENDS_HOME/setup
```

Configure the LDAP server with SSL using a self-signed certificate

Assume that you are running following commands under *$OPENDS_HOME/config* directory.

1. Create a private key for the certificate

```
% /usr/java/jdk/bin/keytool -genkey -alias server-cert -keyalg rsa \
 -dname "CN=pdsops.jpl.nasa.gov,O=JPL,C=US" -keystore keystore -storetype JKS
```

2. Generate a self-signed certificate for the key with the command:

```
% /usr/java/jdk/bin/keytool -selfcert -alias server-cert -validity 1825 \
-keystore keystore -storetype JKS
```

   When you are prompted for the keystore password, enter the same password that you provided in Step 1.

3. Create a text file named *keystore.pin*.
   The file should contain the password that you chose to protect the contents of the keystore.

4. Export the public key for the certificate.

```
% /usr/java/jdk/bin/keytool -export -alias server-cert -file server-cert.txt -rfc \
-keystore keystore -storetype JKS
```

5. Create a new trust store and import the server certificate into it.

```
% /usr/java/jdk/bin/keytool -import -alias server-cert -file server-cert.txt \
-keystore truststore -storetype JKS
```

   Type `yes` when you are prompted about whether you want to trust the certificate.

6. Use the `dsconfig` command to enable the key manager provider, trust manager provider, and

connection handler.

```
% $OPENDS_HOME/bin/dsconfig -h pdsops.jpl.nasa.gov -p 4444 -D "cn=Directory Manager"
-w password \
 -X -n set-key-manager-provider-prop --provider-name JKS --set enabled:true

% $OPENDS_HOME/bin/dsconfig -h pdsops.jpl.nasa.gov -p 4444 -D "cn=Directory Manager"
-w password \
-X -n set-trust-manager-provider-prop --provider-name "Blind Trust" --set
enabled:true

% $OPENDS_HOME/bin/dsconfig -h pdsops.jpl.nasa.gov -p 4444 -D "cn=Directory Manager"
-w password \
-X -n set-connection-handler-prop --handler-name "LDAPS Connection Handler" \
--set "trust-manager-provider:Blind Trust" --set key-manager-provider:JKS \
--set listen-port:636 --set enabled:true
```

Port `636` is the standard LDAPS port, but you might not be able to use it if it is already taken or you are a regular user. If you need to use other port than `636`, then change the `listen-port` property in the last command to the number being used.

7. Test the LDAPS connection with the `ldapsearch` command.

```
% $OPENDS_HOME/bin/ldapsearch --port 636 --useSSL --baseDN "" \
--searchScope base "(objectClass=*)"
```

Type `yes` when you are prompted about whether you want to trust the server's certificate.

2. Install Application Server

Although other application servers are supported (e.g., GlassFish), Apache Tomcat is the preferred application server.

- Download the appropriate binary package for your platform from http://tomcat.apache.org/download-60.cgi .
- Install the package so that it operates on port 80. Documentation can be found in the binary distributions and at http://tomcat.apache.org/tomcat-6.0-doc/ .

  Modify the *$CATALINA_HOME/bin/catalina.sh* file as follows:

```
CATALINA_OPTS="-Xms256m -Xmx1024m"
JAVA_OPTS="-Xmx1024m -XX:MaxPermSize=256m"
```

- Startup the server.

Configuration

This section details the Directory Sever, the Tomcat Server, and the Tomcat Application configuration.

Directory Server Configuration

With the software configuration complete, it is time to add groups and users to the directory server. Execute the commands as follows:

```
% $OPEN_DS/bin/ldapmodify -p 389 -h pdsops.jpl.nasa.gov -D "cn=Directory Manager" \
-w <password> -c -a -f pdspers_schema.ldif

% $OPEN_DS/bin/ldapmodify -p 389 -h pdsops.jpl.nasa.gov -D "cn=Directory Manager" \
-w <password> -c -a -f pds_groups.ldif

% $OPEN_DS/bin/ldapmodify -p 389 -h pdsops.jpl.nasa.gov -D "cn=Directory Manager" \
-w <password> -c -a -f pdsops_pers.ldif
```

Tomcat Server Configuration

Type following command to generate a self-signed server certificate:

```
% $JAVA_HOME/bin/keytool -genkey -alias virtualhostname -keyalg RSA \
-keystore /usr/local/tomcat7/.keystore
```

The password you enter in the first password prompt will be the password for the keystore where your server certificate is stored. For the operational system, you may need to purchase a Certificate from a well-known *Certificate Authority(CA)* such as VeriSign or Thawte.

After generating the server certificate, edit the Tomcat's server configuration file (`$CATALINA_HOME/conf/server.xml`) to have Tomcat server listening on the port 8080. The `redirectPort` option is the port that will be used when redirecting from *http* to *https*.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

To make Tomcat listen on the port 8443, with an SSL transport, the following needs to be configured in the `server.xml` file.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
          maxThreads="150" scheme="https" secure="true"
          clientAuth="false" sslProtocol="TLS"
          keystoreFile="/usr/local/tomcat7/.keystore"
          keystorePass="password"/>
```

Add an OpenDS realm to the Tomcat Server to authenticate the users with the Directory Server.

```
<Realm className="org.apache.catalina.realm.JNDIRealm" debug="99"
        connectionName="cn=Directory Manager"
        connectionPassword="password"
        connectionURL="ldap://pdsdev.jpl.nasa.gov:389"
        userPattern="uid={0},ou=people,dc=pdsdev,dc=jpl,dc=nasa,dc=gov"
        roleBase="ou=groups,dc=pdsdev,dc=jpl,dc=nasa,dc=gov"
        roleName="cn"
        roleSearch="(uniqueMember={0})"/>
```

To enable Single Sign On feature of the Tomcat server, make sure following element is not commented out.

```
<Valve class="org.apache.catalina.authenticator.SingleSignOn"/>
```

Tomcat Application Configuration

Add the security-constraint, role, and login-config elements to your application's *web.xml* file as shown below.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>registry-service</web-resource-name>
    <url-pattern>/*</url-pattern>
    <http-method>DELETE</http-method>
    <http-method>POST</http-method>
    <http-method>PUT</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>PDS_Affiliate</role-name>
  </auth-constraint>
</security-constraint>
```

```
<login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>OpenDS</realm-name>
</login-config>
```

Add the following in the your application's web.xml (`$CATALINA_HOME/webapps/yourapplication/WEB-INF/web.xml`) in the `<security-constraint>` tag:

```
<user-data-constraint>
   <transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
```

This forces a switch from *http* to *https*, using the secure protocol. With this configuration, you can create a Tomcat application that will automatically be secured if accessing it at:

`http://localhost:8080/registry-service/extrinsics`

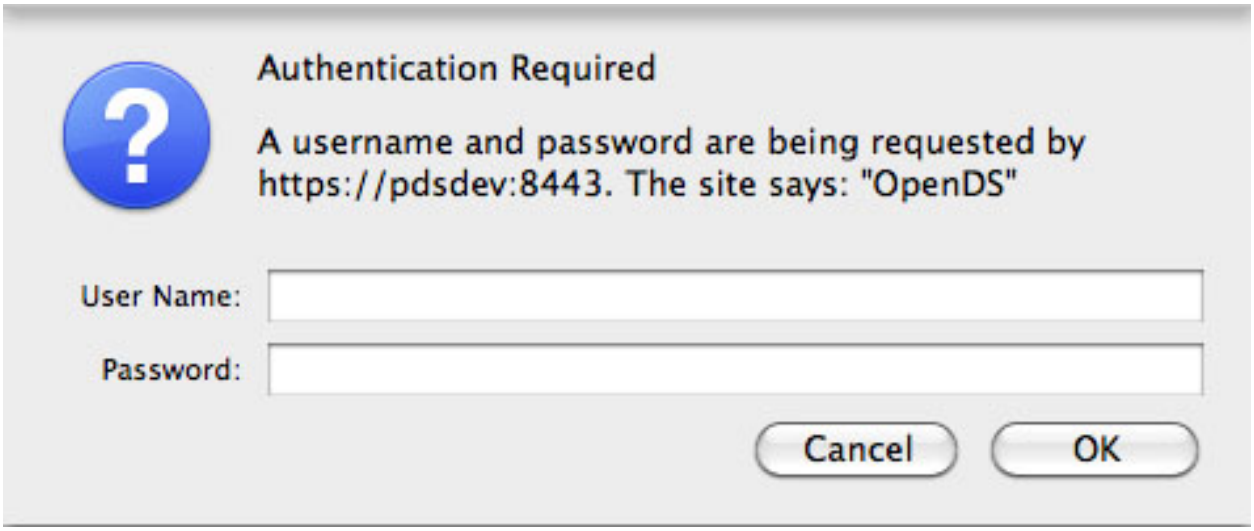
You will be automatically redirected to:

`https://localhost:8443/registry-service/extrinsics`

1.4 Operation

Operation

When Graphical User Interface (GUI) applications have been configured for access control, the following login screen is displayed for users to provide their identification credentials:



Once the user enters their identification credentials (user name and password) correctly, they can access the requested application. Also, you can utilize the *curl* command to access a controlled application. An example of the command is as follows:

```
% curl -X POST -H "Content-type:application/xml" -d @./new_product_build0.xml \
-v https://localhost:8443/registry-service/registry/extrinsics -u username:password \
-k -c tomcat_cookie.txt
```

The cookie file *tomcat_cookie.txt* can then be passed to the next *curl* command:

```
% curl -X POST -H "Content-type:application/xml" -d @./new_product_build1.xml \
-v https://localhost:8443/registry-service/registry/extrinsics/1234 -k -b
tomcat_cookie.txt
```

The example above also applies to a *DELETE* request.