Credential Service Administrator Guide 1.0

oats_cred_admin_guide

INAF-OATs Technical Report 234 - Installation, configuration, deployment and run guide for the Credential Delegation Service.

Sara Bertocco

Giuliano Taffoni

Credential Service Administrator Guide 1.0 oats_cred_admin_guide INAF-OATs Technical Report 234 - Installation, configuration, deployment and run guide for the Credential Delegation Service. Edition 1

Author	Sara Bertocco	*bertocco@oats.inaf.it*
Author	Giuliano Taffoni	*taffoni@oats.inaf.it*

This guide explains how to install, configure, deploy and run the Credential Delegation service of the open software library of Canadian Astronomy Data Center, freely available at *https://github.com/opencadc*

Introduction

The Credential Delegation is a client and server implementation of the IVOA Credential Delegation Protocol. The Credential Delegation Protocol allows a client program to delegate a user's credentials to a service such that that service may make requests of other services in the name of that user. The service provides a RESTful interface.

It requires an SQL Resource DataBase Management Service (RDBMS) as default persistence layer built-in and it provides a RESTful interface .

1.1. Requirements and recomendations

Prerequisites and recomendations to run the Credential Delegation Service are:

- Centos 7 Operative System is recommended,
- A Tomcat 7 installation with SSL/TLS support enabled is required,
- An SQL RDBMS installation. MySql or MariaDB are recommended.

Tomcat Web Server Installation

This chapter explains how to install Tomcat 7, the packages needed to support SSL/TLS and how to run it on privileged ports. the guide is referred to the centOS 7 distribution.

2.1. Tomcat Download and Install

Install epel repository:

-
  ```
  yum install epel-release
  ```

Install tomcat server:

-
  ```
  yum install tomcat -y
  ```

Install APR native support to add SSL/TLS support:

-
  ```
  yum install openssl-devel -y
  yum install tomcat-native -y
  ```

Install the tomcat-jsvc package. It allows tomcat process to bind to ports 80 and 443 running as non privileged user.

-
  ```
  yum install tomcat-jsvc -y
  ```

The server, as installed until now, does not serve any content. Some content can be placed into the server installing optional packages:

-
  ```
  yum install tomcat-admin-webapps.noarch
  yum install tomcat-docs-webapp.noarch
  yum install tomcat-javadoc.noarch
  yum install tomcat-systemv.noarch
  yum install tomcat-webapps.noarch
  ```

2.2. Run tomcat web server

Enable tomcat to start at boot time:

```
systemctl enable tomcat
```

Start tomcat:

```
systemctl start tomcat
```

2.3. Run tomcat web server as tomcat user on port 80 or 443 using JSVC

Using jsvc, the tomcat web server can be run as tomcat user on port 80 or 443:

Enable tomcat-jsvc to start at boot time:

```
systemctl enable tomcat-jsvc
```

Start tomcat-jsvc:

```
systemctl start tomcat-jsvc
```

Tomcat Web Server Configuration

This chapter explains how to configure tomcat web server to support

- SSL/TLS using APR (Apache Portable Runtime) native support
- proxies
- opencadc name-password login.

Reference guide for tomcat SSL/TLS configuration is:

https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html

3.1. Introduction to SSL/TLS

Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL), are cryptographic protocols which allow web browsers and web servers to communicate in a secure way. This means that the data being sent is encrypted by one side, transmitted and then decrypted by the other side before processing. This is a two-way process, meaning that both the server and the browser encrypt all traffic before sending out data.

SSL/TLS protocol requires Authentication. During the initial attempt to communicate with a web server over a SSL/TLS connection, the server will present to the web browser with a set of credentials, in the form of a "Certificate", proving the site it is who and what it claims to be. The "Client Authentication" can also be required, i.e. the server may also request a certificate from the web browser, as proof that it is who it claimis to be.

3.2. Server Certificate

In order to implement SSL/TLS, a web server must have an associated certificate for each external interface (IP address) that accepts secure connections. The certificate is like a "digital passport", it states that its owner is who it should be.

For the certificate to work in the visitors browsers without warnings, it needs to be cryptographically signed by a trusted third party. These third parties are called Certificate Authorities (CAs).

A signed certificate can be obtained asking to a Certification Authority following the instructions provided by the CA itself. A range of CAs is available and some CA offers certificates free of charge.

We need a server certificate to implement the server side SSL/TLS configuration. For this reason, we will ask a certificate to a certification authority which release three files:

```
SERVER-CERT.pem
SERVER-KEY.pem
CA_CERT_CHAIN.pem
```

Sometimes the certificate file is provided in .p12 format. In this case the two files .crt(.pem) and .key can be obtained using openssl commands:

```
openssl pkcs12  -nocerts -in cert.p12 -out certkey.pem
openssl pkcs12 -clcerts  -nokeys -in cert.p12 -out certcrt.pem
```

The Certification Authority Certificate chain is a single file in case of a root CA. It can be a concatenation of more files in case of subordinate CAs. A subordinate CA is a CA that has been issued a certificate by another CA (root CA). There can be more levels of subordinate CAs. In case of a root CA with two layers of subordinate CAs and server certificate released by the second level subordinate CA, e.g.

```
    Root Certificate (root-cacert.pem)
    Sub CA 1 (subCA1-cacert.pem)
    Sub CA 2 (subCA2-cacert.pem)
    Server Certificate (my_server_cert.p12)
```

the CA certificate chain file is obtained chaining the files in the order below:

```
cat subCA2-cacert.pem subCA1-cacert.pem root-cacert.pem >> CA_CERT_CHAIN.pem
```

Be careful that the order matters. The right order is the inverse one:

```
 Intermediate 3, Intermediate 2, Intermediate 1, Root Certificate.
```

After combining the ASCII data into one file, the certificate chain validity for sslserver usagei can be verified issuing:

```
openssl verify -verbose -purpose sslserver -CAfile MY_CA_CERT_CHAIN.pem MY-SERVER-CERT.pem
```

3.3. Certificates bundle

In order to support the client authentication, tomcat must know the Certificates of Certification Authorities (CAs) whose clients he deal with. This is done configuring a certificate bundle. The best way is to configure tomcat to use the system certificate bundle adding to it all the needed CAs certificates. The default system certificate bundle is:

```
/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
```

The instructions to modify the system ca-bundle provided by the CentOS 7 distribution are below:

```
cat /etc/pki/ca-trust/extracted/pem/README
This directory /etc/pki/ca-trust/extracted/pem/ contains
CA certificate bundle files which are automatically created
based on the information found in the
/usr/share/pki/ca-trust-source/ and /etc/pki/ca-trust/source/
directories.

All files are in the BEGIN/END CERTIFICATE file format,
as described in the x509(1) manual page.

Distrust information cannot be represented in this file format,
and distrusted certificates are missing from these files.

If your application isn't able to load the PKCS#11 module p11-kit-trust.so,
then you can use these files in your application to load a list of global
root CA certificates.

Please never manually edit the files stored in this directory,
because your changes will be lost and the files automatically overwritten,
each time the update-ca-trust command gets executed.

Please refer to the update-ca-trust(8) manual page for additional information.
```

The steps to perform are slightly different (the target directory where to copy the certificate files is **`/etc/pki/ca-trust/source/anchors/`**. Using our example:

```
cp subCA2-cacert.pem /etc/pki/ca-trust/source/anchors/
cp subCA1-cacert.pem /etc/pki/ca-trust/source/anchors/
cp root-cacert.pem /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
```

A wide set of Certification Authorities can be supported. The CA certificates can be added using rpm distributions. Two useful reference links with instructions on how to install Certification Authorities are:

- *EGI IGTF Release*[1]
- *Open Science Grid: Installing Certificate Authorities Certificates and related RPMs* [2]

Here a quick reference to install EGI IGTF distribution and to populate the system ca-bundele with the newly installed certificates:

```
wget -O /etc/yum.repos.d/EGI-trustanchors.repo http://repository.egi.eu/sw/production/cas/1/
current/repo-files/EGI-trustanchors.repo
yum install ca-policy-egi-core
cd /etc/grid-security/certificates/
for i in `ls *.pem` do; ln -s /etc/grid-security/certificates/$i /etc/pki/ca-trust/source/
anchors/$i`; done
update-ca-trust extract
```

3.4. SSL tomcat configuration

Tomcat can use two different implementations of SSL:

- the JSSE implementation provided as part of the Java runtime (since 1.4)
- the APR implementation, which uses the OpenSSL engine by default.

This guide explains how to configure the APR implementation.

The APR connector can be configured in **`$TOMCAT_HOME/conf/server.xml`**, as below:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
           sslProtocol="TLS" />

<Connector port="80" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="443" />

<Connector port="443" protocol="org.apache.coyote.http11.Http11AprProtocol"
          maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
```

[1] https://wiki.egi.eu/wiki/EGI_IGTF_Release
[2] https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallCertAuth

```
            SSLCertificateFile="/opt/tomcat/conf/tls/MY-SERVER-CERT.pem"
            SSLCertificateKeyFile="/opt/tomcat/conf/tls/MY-SERVER-KEY.key"
            SSLCertificateChainFile="/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem"
            SSLCACertificateFile="/etc/pki/tls/certs/ca-bundle.crt"
            SSLVerifyClient="require" SSLVerifyDepth="10"
            SSLProtocol="ALL"
           />
```

Where

- **MY-SERVER-CERT.pem** and **MY-SERVER-KEY.key** are the server certificate
- **MY_CA_CERT_CHAIN.pem** is the CA certificate file chain as explained above.
- **ca-bundle.crt** is the system default ca-bundle enriched with the installed certificates, as explained above.

The defaul OpenSSL engine, can be used configuring it in **`${TOMCAT_HOME}/conf/server.xml`**:

```
<Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
```

the APR library must be available (see tomcat installation guide for instructions).

To check the SSL/TLS configuration and verify the set of supported CA, use:

```
openssl s_client -showcerts  -connect <YOUR_SERVER_HOST_FQDN>:443 -debug
```

3.5. Enable SSL debug

To enable the SSL debug output in catalina.out, set

```
JAVA_OPTS="$JAVA_OPTS -Djavax.net.debug=ssl -Djava.net.debug=record,keygen,handshake"
```

in **`${TOMCAT_HOME/bin/catalina.sh`**

3.6. Enable proxy usage in tomcat

To enable the proxy support in Tomcat, it must be defined an environment variable OPENSSL_ALLOW_PROXY_CERTS and set it equals to 1. This can be achieved adding

```
export OPENSSL_ALLOW_PROXY_CERTS=1
```

at the beginning of **`${TOMCAT_HOME}/bin/catalina.sh`** or in the systemd service script or in a file **`${TOMCAT_HOME}/bin/setenv.sh`**.

3.7. Enable login-password authentication cadc software specific

The login-password authentication in the cadc software is provided through a plugin for Tomcat 7. The authentication mechanism will call the access control web service (in module cadc-access-control-server) to see if the credentials are correct.

This plugin is obtained compiling the **`ac/cadc-tomcat`** module. It must be install it in the tomcat installation:

```
cp cadc-tomcat-<version>.jar ${TOMCAT_HOME}/lib
```

and configured adding to the file **`${TOMCAT_HOME}/conf/server.xml`** a **`Realm`** element as below:

```
     <Realm className="ca.nrc.cadc.tomcat.CadcBasicAuthenticator"
            loginURL="http://YOUR_AC_SERVER_HOST/ac/login" />
```

SQL database installation

The credential delegation service requires a SQL database as persistence layer. At CADC (Canadian Astronomy Data Center) Sybase or PostgreSQL, at INAF-OATs MySql or MariaDB are used. This means the high level of configurability of the service. This guide provides installation instructions for MySql and MariaDB. One of the two has to be choosed by the user. Main documentation references:

- *https://www.digitalocean.com/community/tutorials/how-to-install-mariadb-on-centos-7*
- *https://www.linode.com/docs/databases/mariadb/how-to-install-mariadb-on-centos-7*
- *https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-centos-7*
- *https://www.linode.com/docs/databases/mysql/how-to-install-mysql-on-centos-7*

4.1. MariaDB installation step by step

MariaDB is included in the CentOS 7 distribution. Steps to install it:

Update the system:

```
yum update
```

install MariaDB server:

```
yum install mariadb-server
```

start MariaDB server:

```
systemctl start mariadb
```

Check if the service correctly started:

```
systemctl status mariadb
```

enable MariaDB server start at boot time:

```
systemctl enable mariadb
```

To address several security concerns in a default MariaDB installation a script is provided: **`mysql_secure_installation`**. It is recommended to run it. It will ask:

- If you want change MariaDB root password
- If you want remove anonymous user accounts
- If you want disable root logins outside of localhost
- If you want remove test databases

It is recommended to answer yes to all these options.

4.2. Mysql installation step by step

Mysql is not included in the CentOS 7 distribution and it must be installed from the community repository. Steps to install it:

Update the system:

```
yum update
```

Download mysql rpm from the community repository:

```
wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
```

install the downloaded rpm:

```
yum install mysql-community-release-el7-5.noarch.rpm
```

start mysql server:

```
systemctl start mysqld
```

Check if the service correctly started:

```
systemctl status mysqld
```

enable MySql server start at boot time:

```
systemctl enable mysqld
```

To address several security concerns in a default MariaDB installation a script is provided: **`mysql_secure_installation`**. It is recommended to run it. It will ask:

- If you want change MariaDB root password
- If you want remove anonymous user accounts
- If you want disable root logins outside of localhost
- If you want remove test databases

It is recommended to answer yes to all these options.

The CADC Credential Delegation Protocol

The CADC (Canadian Astronomy Data Center) open source software is available on a github repository:

```
https://github.com/opencadc
```

5.1. Credential Delegation Protocol software description

The Credential Delegation Protocol (cdp) module is a client and server implementation of the IVOA Credential Delegation Protocol.

It contains two main modules:

1. **cadc-cdp** : Credential Delegation Protocol clients and common model objects and exceptions.

2. **cadc-cdp-server** : Credential Delegation Protocol server and common model objects.

5.2. Credential Delegation Protocol software repository

A copy of the Credential Delegation Protocol can be obtained by:

```
git clone https://github.com/opencadc/cdp
```

This is the main reference repository, but it is often updated and no tags are available, so, a tagged/versioned repo (forked by the main) can be reached at

http://www.ict.inaf.it/gitlab/OATS-CADC/gitlab/cdp

This guide is relative to the version tagged as v0.2.1

To work with it, issue:

```
git clone https://www.ict.inaf.it/gitlab/OATS-CADC/cdp.git
git checkout tags/v0.2.1
```

or to work on a new local branch:

```
git clone https://www.ict.inaf.it/gitlab/OATS-CADC/cdp.git
git checkout tags/v0.2.1 -b <branch-name>
```

The CADC libraries are highly configurable. In this case, the persistence must be customized extending the class

```
org/astrogrid/security/delegation/Delegations.java
```

The CADC repository provides an implementation working with in-memory certificates. To add a persistence layer, a

```
org/astrogrid/security/delegation/DelegationsImpl.java
```

has to be provided. An example of it can be reached at:

```
https://www.ict.inaf.it/gitlab/OATS-CADC/cdp/blob/master/cadc-cdp-server/
        src/main/java/org/astrogrid/security/delegation/DelegationsImpl.java
```

and using MySql or MariaDb and mantaining the database name equals to "oatscdp", it can be used as is.

Warning

Carefull must be taken: if the class org/astrogrid/security/delegation/DelegationsImpl.java does not exist in the code, the main operations on certificates ("delegate", "get" and "view") will work anyway by command line, but the created certificates will not be persisted and the cases in wich a persisted copy is needed will not work.

5.3. Credential Delegation Protocol software build

Each software module can be easily built using gradle (a build.gradle file is included in each module).

To build versus the newer library version published in maven central repository, the build.gradle file must contain cadc dependencies written as:

```
compile 'org.opencadc:cadc-log:1.+'
```

To build using a local build of cadc libraries, the file must contain cadc dependencies written as:

```
compile files('/<local-path>/core/cadc-log/build/libs/cadc-log.jar')
```

The right order to build opencadc libraries locally is:

-
  ```
  core/cadc-util
  ```

-
  ```
  core/cadc-log
  ```

-
  ```
  ac/cadc-tomcat
  ```

-
  ```
  reg/cadc-registry
  ```
-

```
reg/cadc-vosi
```

-
  ```
  uws/cadc-uws
  ```

-
  ```
  uws/cadc-uws-server
  ```

-
  ```
  cdp/cadc-cdp
  ```

-
  ```
  cdp/cadc-cdp-server
  ```

5.4. Credential Delegation Service deployment

The Credential Delegation Service deployment package can be obtained issuing:

```
git clone https://www.ict.inaf.it/gitlab/OATS-CADC/oats-cred-web.git
```

The repository contains some files to customize:

```
oats-cred-web/src/main/webapp/WEB-INF/web.xml.template
oats-cred-web/src/main/webapp/META-INF/context.xml.template
oats-cred-web/src/main/sql/create_cdp_db.sql.template
oats-cred-web/build.gradle.template
oats-cred-web/src/main/webapp/capabilities.xml
```

Each template file contains comments/instructions on how to customize it, except build.gradle.template which can be customized, as previously described in *Section 5.3, “Credential Delegation Protocol software build”*, vs. maven central hosted or local libraries.

The files context.xml.template and create_cdp_db.sql.template contain database connection parameters, so they must be customized accordingly and keeping into account the configuration encoded in the class DelegationsImpl.java (see *Section 5.2, “Credential Delegation Protocol software repository”*).

The file capabilities.xml defines standards and capabilities (e.g., IVOA defined interfaces) provided by the service and it must be customized accordingly to the local service features.

The package compilation produces the **`cred.war`** file. It can be deployed issuing:

```
systemctl stop tomcat
cp cred.war $TOMCAT_HOME/webapps
systemctl start tomcat
```

If some customization is done in the file contained in the deployment location, be careful that the expected behavior of Tomcat is to treat the .war file deletion as a request to redeploy or undeploy the application. So, if the war file is deleted until tomcat is running, the application is undeployed (removed) by tomcat.

If tomcat is stopped before deleting WAR file and started only after that, the application will not be undeployed.

Appendix A. Revision History

Revision 1.0-0 Fri March 03 2017 **Sara Bertocco**
sara.bertocco@oats.inaf.it

Initial creation by publican

Index