Certificates

A certificate performs authentication and encryption for websites that use the HTTPS protocol. A certificate contains information about an entity and contains a public key. The public key is related to a private key which is NOT part of the certificate, but is used by one entity when it communicates with another entity.

MapR stores the private key and certificate in a key store file called `ssl_keystore`. A certificate is also digitally signed so that it cannot be altered. The signer is known as the signing certificate.

In order for an HTTPS connection to be established, the following criteria must be met:

- the *server* must have a key file that contains a certificate and private key
- the *client* must provide a trust file that contains a signer who signed the certificate used by the server
- the server certificate must be valid and not expired
- the client must determine that the SubjectDN in the certificate is acceptable

The process of enabling MapR security generates the common `ssl_keystore` and `ssl_truststore` files on the first CLDB server that are used by all clients and servers.

Warning:

You can also generate your own `ssl_keystore` and `ssl_truststore` files. Pass the `-nocerts` flag to the `configure.sh` script during to suppress automatic generation of keystore and truststore files, then specify the location of your own storefiles in the `mapr.login.conf` file. Contact MapR Support for assistance in generating JKS-format storefiles from your vendor-supplied certificates.

- The `ssl_keystore` contains a single self-signed certificate with a wildcard SubjectDN (for example, if the hostname of the CLDB is `a.b.com` the SubjectDN would be `CN=*.b.com`).
- The `ssl_truststore` contains the signer for the certificate in the `ssl_keystore`.

Adding Your Certificate to the Keystore

The REST API calls in a MapR cluster communicate over the HTTPS protocol on port 8443. These calls are secured with SSL certificates that identify a node to the cluster. The `./adminuiapp/webapp/WEB-INF/ssl_keystore` file stores SSL certificates.

The default keystore contents are generated during installation. To add an existing certificate and its corresponding key from a vendor to the keystore, use the `keytool` command.

Warning: The following steps assume your vendor has delivered the certificate in Java KeyStore (JKS) format, or another format supported by the `keytool` utility.

On a Linux of MacOS system, use a syntax similar to the following example:

```
keytool -import -trustcacerts -file <certificate>.crt -alias <hostname> -keystore $JAVA_HOME/jre/lib/security/cacerts
```

On a Windows system, use a syntax similar to the following example:

```
keytool -import -trustcacerts -file <certificate>.crt -alias <hostname> -keystore %JAVA_HOME%\jre\lib\security\cacerts
```

The default password for the keystore is `changeit`.

Verify that the certificate is now in the keystore with the following command:

```
keytool -list -v -keystore $JAVA_HOME/jre/lib/security/cacerts
```