Certificate Properties File Realm

{scrollbar}

This realm type allows you to configure Web applications to authenticate users against it. To get to that point, you will need to first configure Geronimo to use a custom SSL port listener and to get to that point you will need to configure SSL keys and keystore. The following sections describe step-by-step how to configure each of these modules.

Create keystore and certificate

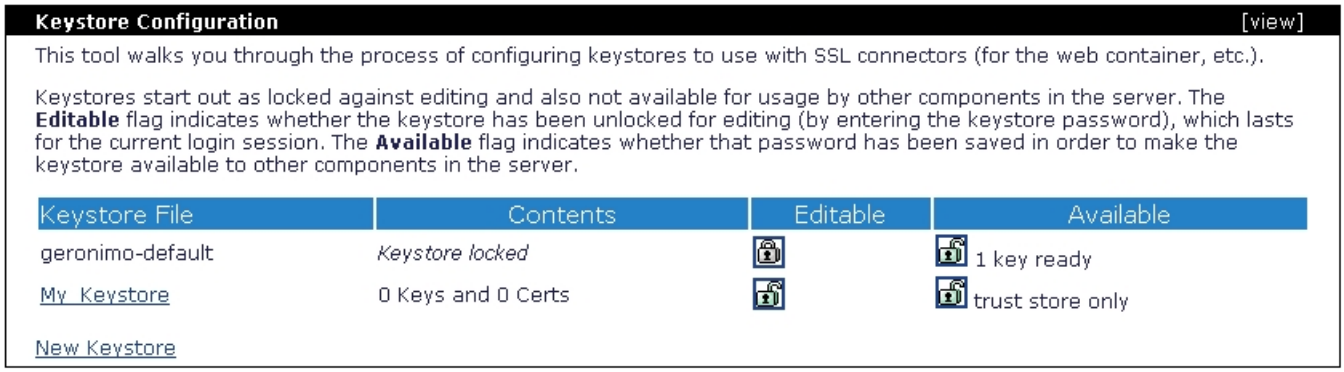
For this configuration we will create a new keystore, a new private key, a CSR and will import the CA reply

We already mentioned in the Administering Certificates section how to create a keystore and a private key, in this section we will complete the picture by generating a CSR and importing the CA's reply.

The keystores in Geronimo are stored in the **<geronimo_home>\var\security\keystores** directory, the default keystore already provided with the installation is **geronimo-default**. For this exercise we will create a new keystore.

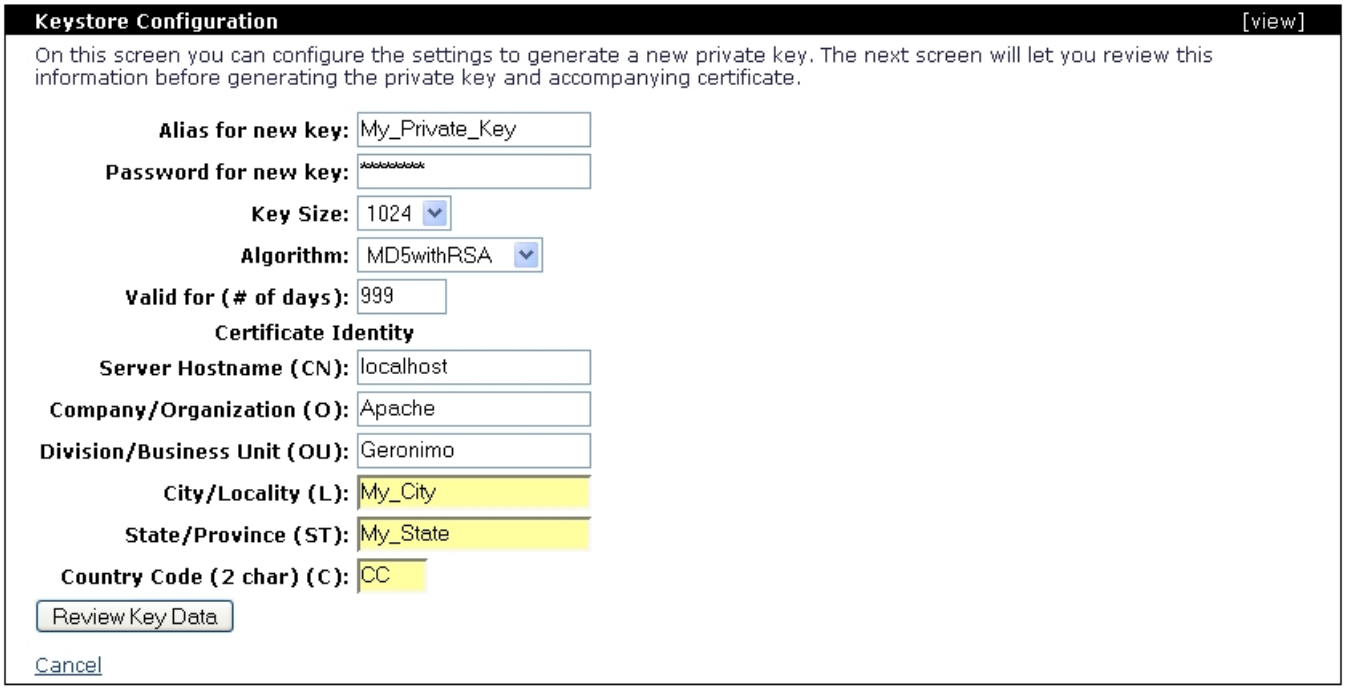
From the Geronimo Administration Console click on **Keystores** to access the **Keystore Configuration** portlet.

Click on **New Keystore**, specify a new keystore name and password and then click on **Create Keystore**. For this example we used `My_Keystore` and `password` respectively.



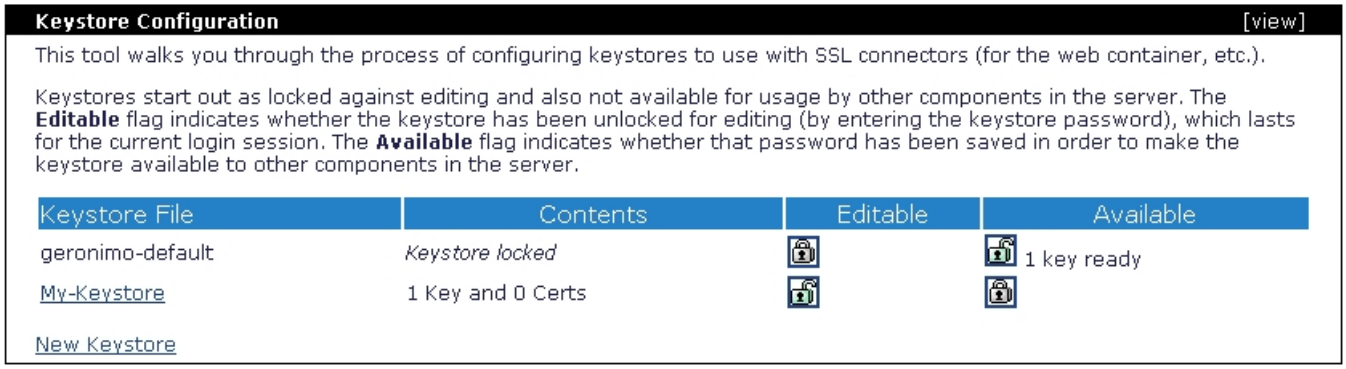
Click on the keystore file you just created, and create a private key by clicking on the appropriate link.

Fill in with the appropriate data and click on **Review Key Data**.

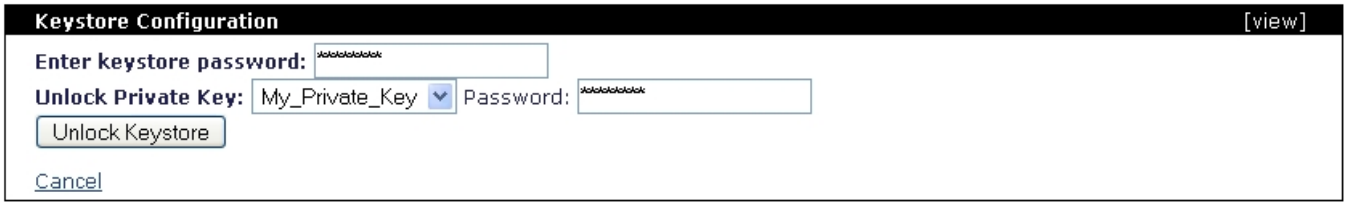


Once you verified the values are correct click on **Generate Key**.

Right after you created a new private key, this key is automatically locked. That means that you can only view it or delete it, to create a Certificate Signing Request (CSR) you will have to unlock the key. To do that click on **Return to keystore list**.



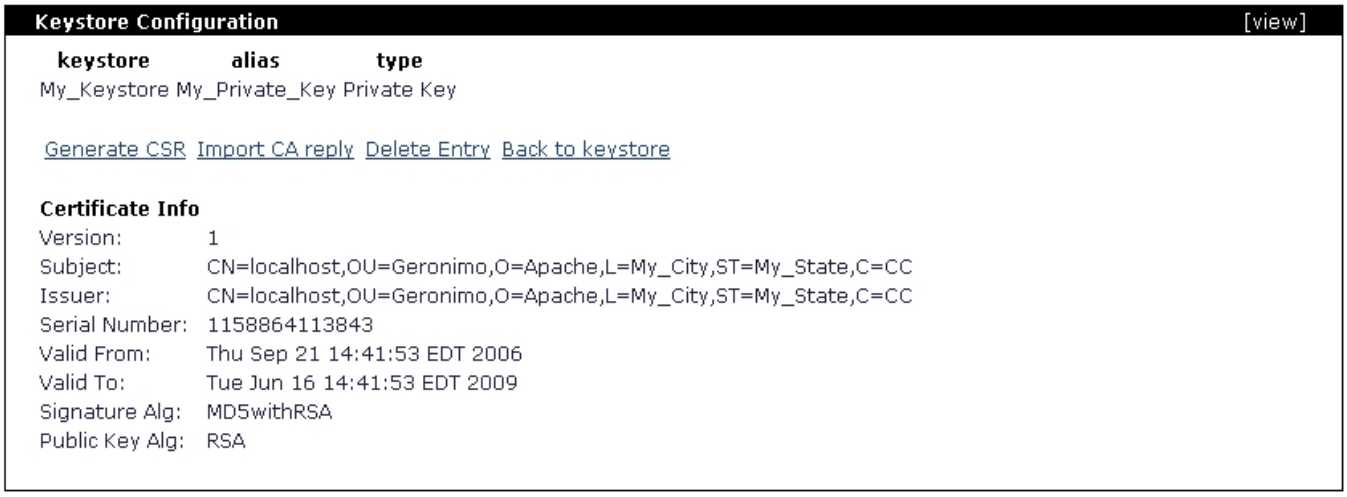
Click on the  to unlock the private key. You will be prompted with the password for the keystore and for the private key.



Click on **Unlock Keystore**.

Create a Certificate Signing Request (CSR) and import CA reply

Now that you have the private key unlocked you may now continue to create a CSR. From the **Keystore Configuration** portlet click on the keystore file you created to display the current content. In this example we only have one private key. Click on either **view** or the alias links for the current private key to display the details and additional actions.



Click on **Generate CSR**, the certificate request should be displayed as illustrated in the following figure.

This is a **PKCS10** certification request, you should copy this text and paste it into a flat txt file so it can be sent to a CA.

solidcsr.txt -----BEGIN CERTIFICATE REQUEST----- MIIBqDCCARECAQAwajESMBAGA1UEAxMJbG9jYWxob3N0MREwDwYDVQQLEwhHZXJvbmltbz
EPMA0GA1UEChMGQXBhY2hlMRAwDgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0
ZTELMAkGA1UEBhMCQ0MwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIOSt7pc/0OYH+ tFCUai/VU7fnjdQssw15lZ
/0k5PBbbvz0D7cK8ZoMNccNbf+TrjWeyEPpmMZcwmOw3xzJ0 lFBa03DMWbcNwUcEkvM7ok1O6O8v+53rVjZLIuizS9VCma4jXj2ThsBWWihgRZ+r2j/Htb
3uNvzdgcjw276SGRvlAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAMRhP1vEgNYBlvAfR9Cid FVcs+iSQJe6d/Ugq34fl
/ZIdwOvB72ZbyfaRpRUJJ9YGx6XC93aa811Q0pqSSr09ONRuM0 D8QUTbxyjgl0T9ox404w9P72Lj1mC8VExUfgd0FjQq0wpVSzbG+S/cbH1EPcu
/djrAFhta cimNSwy3zXs= -----END CERTIFICATE REQUEST-----

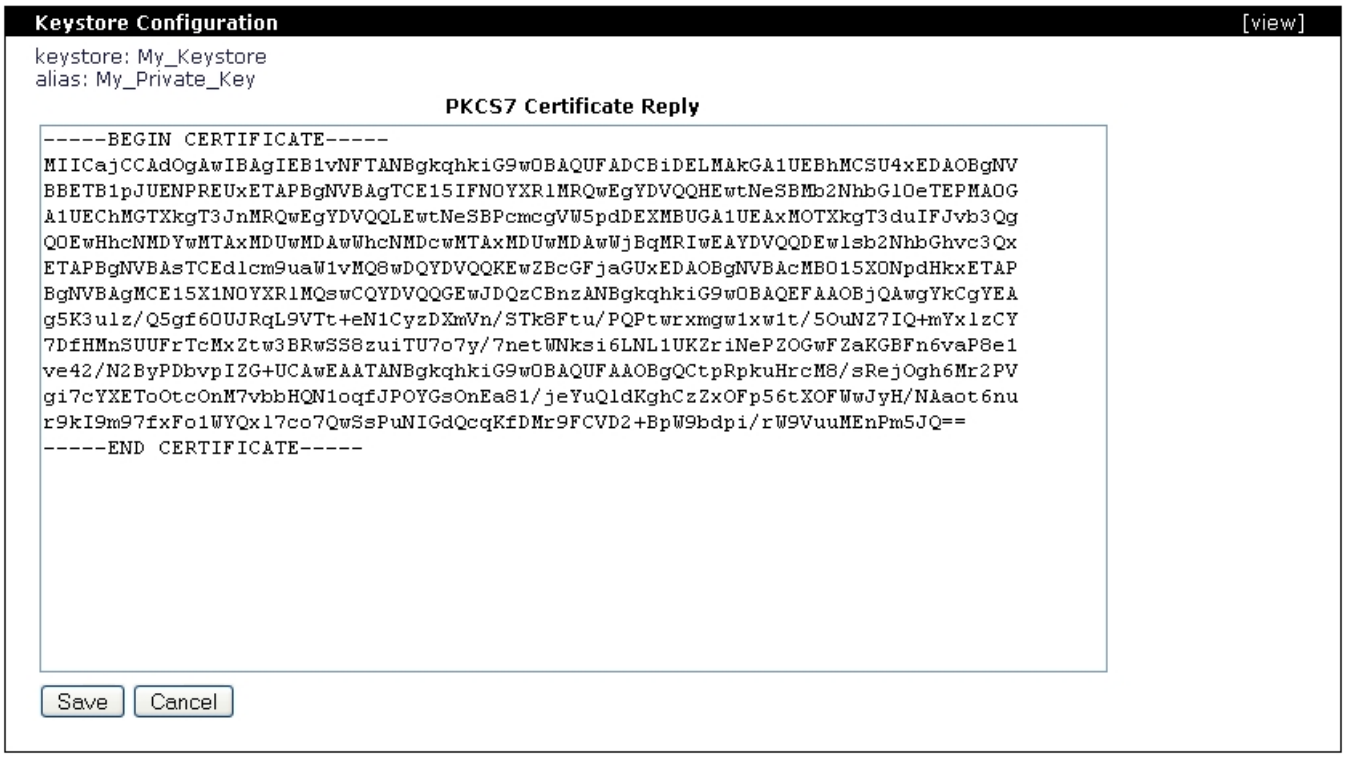
You can now click **Back** to return to the private key details portlet.

For this example we used a custom, home made CA so we could sign our own certificates for this test without altering the standard procedure. Assuming that you sent you CSR to a CA, the CA should respond back with another similar file containing the CA signed certificate.

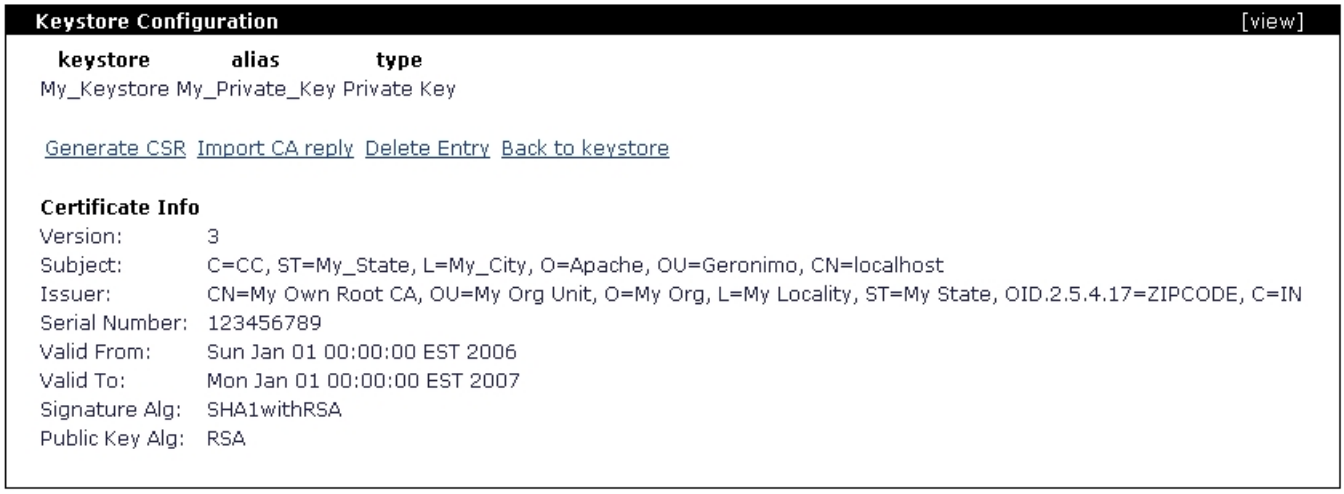
solidcsr_ca_reply.txt -----BEGIN CERTIFICATE-----
MIICajCCAdOgAwIBAgIEB1vNFTANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCSU4xEDAOBgNV
BBETB1pJUENPREUxETAPBgNVBAgTCE15IFN0YXRlMRQwEgYDVQQHEwtNeSBMb2NhbGl0eTEPMA0G
A1UEChMGTXkgT3JnMRQwEgYDVQQLEwtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3Qg
Q0EwHhcNMDYwMTAxMDUwMDAwWhcNMDcwMTAxMDUwMDAwWjBqMRIwEAYDVQQDEwlsb2NhbGhvc3Qx
ETAPBgNVBAsTCEdlcm9uaW1vMQ8wDQYDVQQKEwZBcGFjaGUxEDAOBgNVBAcMB015X0NpdHkxETAP

BgNVBAgMCE15X1N0YXRlMQswCQYDVQQGEwJDQzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA g5K3ulz
/Q5gf60UJRqL9VTt+eN1CyzDXmVn/STk8Ftu/PQPtwrxmgw1xw1t/5OuNZ7IQ+mYxlzCY 7DfHMnSUUFrTcMxZtw3BRwSS8zuiTU7o7y
/7netWNksi6LNL1UKZriNePZOGwFZaKGBFn6vaP8e1 ve42/N2ByPDbvpIZG+UCAwEAATANBgkqhkiG9w0BAQUFAAOBgQCtpRpkuHrcM8
/sRejOgh6Mr2PV gi7cYXEToOtcOnM7vbbHQN1oqfJPOYGsOnEa81/jeYuQldKghCzZxOFp56tXOFWwJyH/NAaot6nu
r9kI9m97fxFo1WYQxl7co7QwSsPuNIGdQcqKfDMr9FCVD2+BpW9bdpi/rW9VuuMEnPm5JQ== -----END CERTIFICATE-----

From the private key details portlet click on **Import CA reply**. Remove any pre-filled text in the certificate reply window and paste the text from the CA reply file and click on **Save**.



After saving the CA reply you should now notice that the certificate now shows a different **Issuer**. Click on **Back to keystore** and then on **Return to keystore list**.



Import trusted certificates

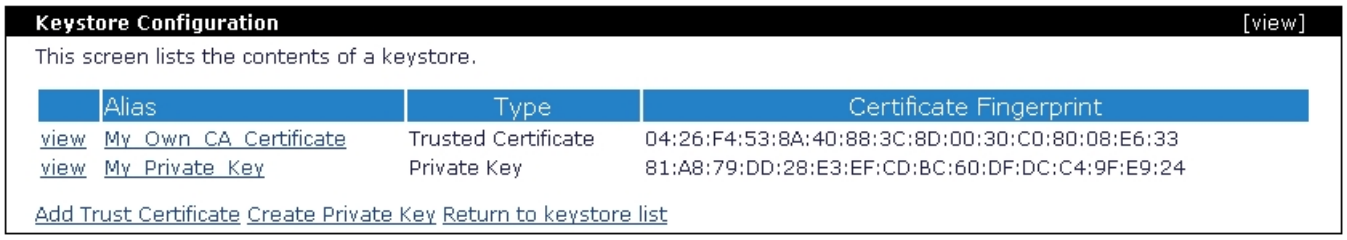
In order to enable client authentication you will need to import the CA who signed your CSR as a trusted certificate, this process has to be only once. The CA should provide along with the signed CSR a separate certificate for the CA itself. For this example we are using our own CA so we generated the following CA certificate.

solidMy_Own_CA_Certificate.txt -----BEGIN CERTIFICATE-----
MIIChjCCAe+gAwIBAgIBADANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCSU4xEDAOBgNVBBET
B1pJUENPREUxETAPBgNVBAgTCE15IFN0YXRlMRQwEgYDVQQHEwtNeSBMb2NhbGl0eTEPMA0GA1UE

ChMGTXkgT3JnMRQwEgYDVQQLEwtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3QgQ0Ew
HhcNMDUxMjMxMTgzMDAwWhcNMTkxMjMxMTgzMDAwWjCBiDELMAkGA1UEBhMCSU4xEDAOBgNVBBET
B1pJUENPREUxETAPBgNVBAgTCE15IFN0YXRlMRQwEgYDVQQHEwtNeSBMb2NhbGl0eTEPMA0GA1UE
ChMGTXkgT3JnMRQwEgYDVQQLEwtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3QgQ0Ew
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOGfnakKJlSw5daLSYAo3dpAqk/HAz9LX0zFZiIi 8m6HiPaiWdkYVoA3WxLzrrzqRDFlDhZUHIRE+nBz
/C/DyGkJm3qRT6EJt/h8lPizN2tgcbHEWtmo GvaYoFMcQ8gjPeDPqSMB36ALH8zdlswZ/t0wR/TogCijVEUnQ+1ERmNHAgMBAAEwDQYJKoZIhvcN
AQEFBQADgYEAOb/KlJK2nQ5VTgC/wZ3G+3Ft4gRNv8iazjRs0N3CX/zAMGxV1ECkveLunBrCn/SN
Mqq9WMVvzJtDbcxU1cE75Ncng893QlfnboShpVsXeH6gTT5saCsIeoD+VoqBaktuS6QuSN1JUz+i NSr1SNzbyoZiPq/UgexQMxFqowA3PDI= -----END
CERTIFICATE-----

While in the Keystore Configuration portlet click on the keystore file you created and then click on **Add Trust Certificate**. Delete any pre-filled content from **Trusted Certificate** window and paste the content from the CA certificate and add an alias to this certificate.

Click on **Review Certificate** and then click on **Import Certificate**. You should now see the trusted certificate you just imported.



Add an HTTPS listener with client authentication

Apache Geronimo comes with a predefined HTTPS listener on port 8443 but this listener is not configured for client authentication. In this example we will add a new HTTPS listener and configure it to request client authentication using the certificates we created and imported in the previous steps.

Note that in this example we are using the Tomcat distribution of Geronimo, although the process is the same some names and links may vary slightly if you are using the Jetty distribution.

From the Geronimo Administration Console click on **Web Server** to access the Network Listener portlet.

Network Listeners help [view]

Name	Protocol	Port	State	Actions	Type
TomcatWebSSLConnector	HTTPS	8443	running	stop edit delete	Tomcat Connector
TomcatWebConnector	HTTP	8080	running	stop edit delete	Tomcat Connector
TomcatAJPConnector	AJP	8009	running	stop edit delete	Tomcat Connector

Add new HTTP listener for Tomcat
Add new HTTPS listener for Tomcat
Add new AJP listener for Tomcat

From the Network Listener portlet click on **Add new HTTPS listener for Tomcat**

Network Listeners help [view]

Add new HTTPS listener for Tomcat

Unique Name: SSL_Client_Authentication
A name that is different than the name for any other web connectors in the server (no spaces in the name please)

Host: 0.0.0.0
The host name or IP to bind to. The normal values are `0.0.0.0` (all interfaces) or `localhost` (local connections only)

Port: 443
The network port to bind to.

Max Threads: 50
The maximum number of threads this connector should use to handle incoming requests

SSL Settings

Keystore File: ır/security/keystores/My_Keystore
The file that holds the keystore (relative to the Geronimo install dir)

Keystore Password: ••••••••
Set the password used to access the keystore file. This is also the password used to access the server private key within the keystore (so the two passwords must be set to be the same on the keystore).

Keystore Type: JKS
Set the keystore type. There is normally no reason not to use the default (`JKS`).

Truststore File:
The file that holds the truststore (relative to the Geronimo install dir)

Truststore Password:
Set the password used to verify the truststore file.

Truststore Type: JKS
Set the truststore type. There is normally no reason not to use the default (`JKS`).

HTTPS Algorithm: Sun
Set the HTTPS algorithm. This should normally be set to match the JVM vendor.

HTTPS Protocol: TLS
Set the HTTPS protocol. This should normally be set to `TLS`, though some (IBM) JVMs don't work properly with popular browsers unless it is changed to `SSL`.

Client Auth Required: ☑
If set, then clients connecting through this connector must supply a valid client certificate. The validity is checked using the CA certificates stored in the first of these to be found:

1. The trust store configured above
2. A keystore file specified by the `javax.net.ssl.trustStore` system property
3. *java-home*`/lib/security/jssecacerts`
4. *java-home*`/lib/security/cacerts`

Save Reset Cancel

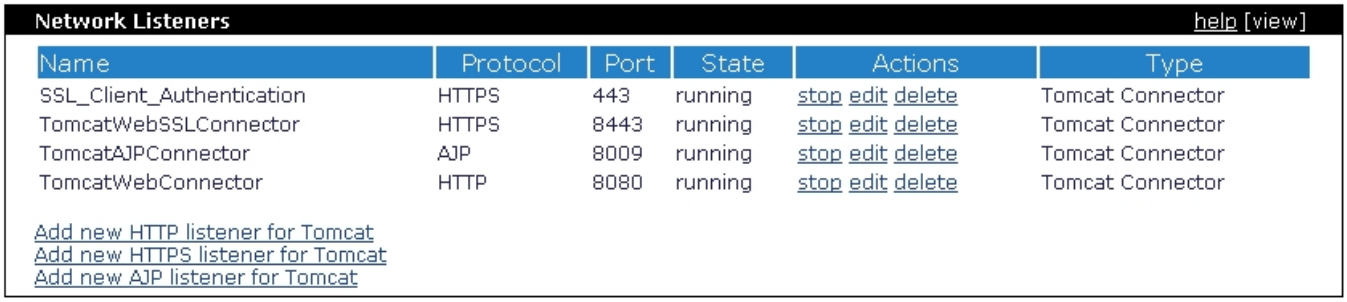
List connectors

Fill in the fields with the appropriate data and click **Save**. For this example we only specified the keystore and not a trustore. When specifying the keystore file path you should add something similar to **`var/security/keystores/<your_keystore>`**, this path is relative to Geronimo's installation home directory.

Select the **Client Auth Required** check box, this tells the HTTPS listener to only establish an encrypted connection with a client that provides a valid client certificate. The client certificates are verified against the CA certificates stored in any of these locations (in order):

1. The trust store configured above
2. A keystore file specified by the javax.net.ssl.trustStore system property
3. java-home/lib/security/jssecacerts
4. java-home/lib/security/cacerts

Once you saved this HTTPS network listener configuration it will get started automatically as you can see in the status displayed. If you try to access this port with your browser it should fail because at this poing you have not configured your client with a valid certificate.



Network Listeners help [view]

Name	Protocol	Port	State	Actions	Type
SSL_Client_Authentication	HTTPS	443	running	stop edit delete	Tomcat Connector
TomcatWebSSLConnector	HTTPS	8443	running	stop edit delete	Tomcat Connector
TomcatAJPConnector	AJP	8009	running	stop edit delete	Tomcat Connector
TomcatWebConnector	HTTP	8080	running	stop edit delete	Tomcat Connector

Add new HTTP listener for Tomcat
Add new HTTPS listener for Tomcat
Add new AJP listener for Tomcat

Install certificate on client