

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](#)
- [#Create a Certificate Signing Request \(CSR\) and import CA reply](#)
- [#Import trusted certificates](#)
- [#Add an HTTPS listener with client authentication](#)
- [#Install certificate on client](#)

This feature is not fully implemented in Geronimo v1.1, for this example you will have to use Geronimo v1.1.1

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.

Keystore Configuration

[view]

This tool walks you through the process of configuring keystores to use with SSL connectors (for the web container, etc.).

Keystores start out as locked against editing and also not available for usage by other components in the server. The **Editable** flag indicates whether the keystore has been unlocked for editing (by entering the keystore password), which lasts for the current login session. The **Available** flag indicates whether that password has been saved in order to make the keystore available to other components in the server.

| Keystore File | Contents | Editable | Available |
|-----------------------------|--------------------|----------|------------------|
| geronimo-default | Keystore locked | | 1 key ready |
| My_Keystore | 0 Keys and 0 Certs | | trust store only |

[New Keystore](#)

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**.

Keystore Configuration

[view]

On this screen you can configure the settings to generate a new private key. The next screen will let you review this information before generating the private key and accompanying certificate.

Alias for new key:

Password for new key:

Key Size:

Algorithm:

Valid for (# of days):

Certificate Identity

Server Hostname (CN):

Company/Organization (O):

Division/Business Unit (OU):

City/Locality (L):

State/Province (ST):

Country Code (2 char) (C):

[Cancel](#)

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

Keystore Configuration

[view]

This screen lists the contents of a keystore.

| | Alias | Type | Certificate Fingerprint |
|----------------------|--------------------------------|-------------|---|
| view | My_Private_Key | Private Key | 21:F7:DE:09:09:D2:02:E7:43:FD:2D:58:97:D6:DD:44 |

[Add Trust Certificate](#)
[Create Private Key](#)
[Return to keystore list](#)

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**.

Keystore Configuration

[view]

This tool walks you through the process of configuring keystores to use with SSL connectors (for the web container, etc.).

Keystores start out as locked against editing and also not available for usage by other components in the server. The **Editable** flag indicates whether the keystore has been unlocked for editing (by entering the keystore password), which lasts for the current login session. The **Available** flag indicates whether that password has been saved in order to make the keystore available to other components in the server.

| Keystore File | Contents | Editable | Available |
|-----------------------------|-------------------|----------|-------------|
| geronimo-default | Keystore locked | | 1 key ready |
| My-Keystore | 1 Key and 0 Certs | | |

[New Keystore](#)

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

Keystore Configuration

[view]

Enter keystore password:

Unlock Private Key: Password:

[Cancel](#)

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.

Keystore Configuration [view]

| keystore | alias | type |
|-------------|----------------|-------------|
| My_Keystore | My_Private_Key | Private Key |

[Generate CSR](#) [Import CA reply](#) [Delete Entry](#) [Back to keystore](#)

Certificate Info

Version: 1
Subject: CN=localhost,OU=Geronimo,O=Apache,L=My_City,ST=My_State,C=CC
Issuer: CN=localhost,OU=Geronimo,O=Apache,L=My_City,ST=My_State,C=CC
Serial Number: 1158864113843
Valid From: Thu Sep 21 14:41:53 EDT 2006
Valid To: Tue Jun 16 14:41:53 EDT 2009
Signature Alg: MD5withRSA
Public Key Alg: RSA

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

Keystore Configuration [view]

keystore: My_Keystore
alias: My_Private_Key

PKCS10 Certification Request

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBqDCCARECAQAwajESMBAGA1UEAxMJbG9jYWxob3N0MREwDwYDVQQLEwhHZXJvbmltbz
EPMA0GA1UEChMGQXBhY2h1MRAwDgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0
ZTElMAkGA1UEBhMCQ0MwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIOSt7pc/0OYH+
tFCUai/VU7fnjdQssw15lZ/0k5PBbbvz0D7cK8Z0MNccNbf+TrjWeyEPpmMZcwmOw3xzJ0
lFBa03DMWbCnWUcEkvM7ok1O6O8v+53rVjZLluzS9VCma4jXj2ThsBWWihgRZ+r2j/Htb
3uNvzdgcjw276SGRv1AgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAMRhP1vEgNYBlvAfr9Cid
FVcs+iSQJe6d/Ugq34fl/ZIdwOvB72ZbyfaRpRUJJ9YGx6XC93aa811Q0pqSSr09ONRuMO
D8QUTbxxyjg10T9ox404w9P72Lj1mC8VExUfgd0FjQq0wpVSzbG+S/cbH1EPcu/djrAFhta
cimNSwy3zXs=
-----END CERTIFICATE REQUEST-----
```

[Back](#)

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
EPMA0GA1UEChMGQXBhY2h1MRAwDgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0
ZTElMAkGA1UEBhMCQ0MwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAIOSt7pc/0OYH+ tFCUai/VU7fnjdQssw15lZ
/0k5PBbbvz0D7cK8Z0MNccNbf+TrjWeyEPpmMZcwmOw3xzJ0 lFBa03DMWbCnWUcEkvM7ok1O6O8v+53rVjZLluzS9VCma4jXj2ThsBWWihgRZ+r2j/Htb
3uNvzdgcjw276SGRv1AgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAMRhP1vEgNYBlvAfr9Cid FVcs+iSQJe6d/Ugq34fl
/ZIdwOvB72ZbyfaRpRUJJ9YGx6XC93aa811Q0pqSSr09ONRuMO D8QUTbxxyjg10T9ox404w9P72Lj1mC8VExUfgd0FjQq0wpVSzbG+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-----
MIICajCCAdOgAwIBAgIEB1vNFTANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCUS4xEDAOBgNV
BBETB1pUENPREUxETAPBgNVBAGTCe15IFN0YXRIMRQwEgYDVQQHEwtNeSBMb2NhbGloeTEPMA0G
A1UEChMGTXkgT3JnMRQwEgYDVQQLEwtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3dulFJvb3Qg
Q0EwHhcNMjYwMTAxMDUwMDAwWhcNMjYwMTAxMDUwMDAwWjBqMRlwEAYDVQQDEwlsb2NhbGhvc3Qx
ETAPBgNVBAsTCEdlcm9uaW1vMQ8wDQYDVQQKEwZBcGJGaGUxEDAOBgNVBACMB015X0NpdHkxETAP
```

```
BgNVBAgMCE15X1N0YXRIMQswCQYDVQQGEwJDQzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA g5K3ulz
/Q5gf60UJRqL9VTt+eN1CyzDXmVn/STk8Ftu/PQPtwrxmgw1xw1t/5OuNZ7IQ+mYxlzCY 7DfHMnSUUFRtCmxZtw3BRwSS8zuiTU7o7y
/7netWNksi6LNL1UKZriNePZOGwFZakGBFn6vaP8e1 ve42/N2ByPDbvplZG+UCAwEAATANBgkqhkiG9w0BAQUFAAOBgQCtpRpkHrcM8
/sRejOgh6Mr2PV gi7cYXEToOtcOnM7vbbHQN1oqfJPOYGsOnEa81/jeYuQldKghCzZxOFp56tXOFWwJyH/NAaot6nu
r9kl9m97fxFo1WYQx17co7QwSsPuNIGDQcKfDMr9FCVD2+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**.

Keystore Configuration

[view]

keystore: My_Keystore
alias: My_Private_Key

PKCS7 Certificate Reply

-----BEGIN CERTIFICATE-----
MIICajCCAdOgAwIBAgIEB1vNFTANBgkqhkiG9w0BAQUFADCBIDELMAkGA1UEBhMCSU4xEDAOBgNV
BBETB1pJUENPREFUeTAPBgNVBAGTCE15IFNOYXR1MRQwEgYDVQQHEwtNeSBMb2NhbG10eTEPMAOG
A1UEChMGTk8Ftu/PQPtwrxmgw1xw1t/5OuNZ7IQ+mYxlzCY 7DfHMnSUUFRtCmxZtw3BRwSS8zuiTU7o7y
/7netWNksi6LNL1UKZriNePZOGwFZakGBFn6vaP8e1 ve42/N2ByPDbvplZG+UCAwEAATANBgkqhkiG9w0BAQUFAAOBgQCtpRpkHrcM8
/sRejOgh6Mr2PV gi7cYXEToOtcOnM7vbbHQN1oqfJPOYGsOnEa81/jeYuQldKghCzZxOFp56tXOFWwJyH/NAaot6nu
r9kl9m97fxFo1WYQx17co7QwSsPuNIGDQcKfDMr9FCVD2+BpW9bdpi/rW9VuuMENPm5JQ==
-----END CERTIFICATE-----

Save

Cancel

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**.

Keystore Configuration

[view]

| keystore | alias | type |
|-------------|----------------|-------------|
| My_Keystore | My_Private_Key | Private Key |

[Generate CSR](#) [Import CA reply](#) [Delete Entry](#) [Back to keystore](#)

Certificate Info

Version: 3
Subject: C=CC, ST=My_State, L=My_City, O=Apache, OU=Geronimo, CN=localhost
Issuer: CN=My Own Root CA, OU=My Org Unit, O=My Org, L=My Locality, ST=My State, OID.2.5.4.17=ZIPCODE, C=IN
Serial Number: 123456789
Valid From: Sun Jan 01 00:00:00 EST 2006
Valid To: Mon Jan 01 00:00:00 EST 2007
Signature Alg: SHA1withRSA
Public Key Alg: RSA

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-----
MIICajCCAe+gAwIBAgIBADANBgkqhkiG9w0BAQUFADCBIDELMAkGA1UEBhMCSU4xEDAOBgNVBBET
B1pJUENPREFUeTAPBgNVBAGTCE15IFNOYXR1MRQwEgYDVQQHEwtNeSBMb2NhbG10eTEPMAOGA1UE
```

```
ChMGTXkgT3JnMRQwEgYDVQQLewtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3QgQ0Ew
HhcNMDUxMjMxMTgzMDAwWWhcNMTkxMjMxMTgzMDAwWjCBiDELMAkGA1UEBhMCSU4xEDAOBgNVBBET
B1pJUENPRESUETAPBgNVBAGTCE15IFNOYXR1MRQwEgYDVQQHEwtNeSBMb2NhbG10eTEPMAOGA1UE
ChMGTXkgT3JnMRQwEgYDVQQLewtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3QgQ0Ew
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOGfnakKJlSw5daLSYA03dpAqk/HAz9LX0zFZili
8m6HiPaiWdkYVoA3WxLzrrzqRDFIdhZUHIRE+nBz/C/DyGkjm3qRT6EJt/h8IPizN2tgcbHEWtmo
GvaYoFMcQ8gjPeDPqSMB36ALH8zdlswZ/t0wR/TogCijVEUnQ+1ERmNHAgMBAAEwDQYJKoZIhvcN
AQEFBQADgYEAOb/KlJK2nQ5VTgC/wZ3G+3Ft4gRNv8iazjRs0N3CX/zAMGxV1ECkveLunBrCn/SN
Mqq9WMVvzJtDbcxU1cE75Ncng893QlfnboShpVsXeH6gTT5saCsleod+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.

Keystore Configuration

[view]

This screen lets you input a certificate to import into the keystore. Paste the content of the certificate file in the text area and specify an alias to store it under in the keystore. The next step will let you review the certificate before committing it to the keystore.

Trusted Certificate

```
-----BEGIN CERTIFICATE-----
MIICChjCCAe+gAwIBAgIBADANBgkqhkiG9wOBAQUFADCBIDELMAkGA1UEBhMCSU4xEDAOBgNVBBET
B1pJUENPRESUETAPBgNVBAGTCE15IFNOYXR1MRQwEgYDVQQHEwtNeSBMb2NhbG10eTEPMAOGA1UE
ChMGTXkgT3JnMRQwEgYDVQQLewtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3QgQ0Ew
HhcNMDUxMjMxMTgzMDAwWWhcNMTkxMjMxMTgzMDAwWjCBiDELMAkGA1UEBhMCSU4xEDAOBgNVBBET
B1pJUENPRESUETAPBgNVBAGTCE15IFNOYXR1MRQwEgYDVQQHEwtNeSBMb2NhbG10eTEPMAOGA1UE
ChMGTXkgT3JnMRQwEgYDVQQLewtNeSBPcmcgVW5pdDEXMBUGA1UEAxMOTXkgT3duIFJvb3QgQ0Ew
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOGfnakKJlSw5daLSYA03dpAqk/HAz9LX0zFZili
8m6HiPaiWdkYVoA3WxLzrrzqRDFIdhZUHIRE+nBz/C/DyGkjm3qRT6EJt/h8IPizN2tgcbHEWtmo
GvaYoFMcQ8gjPeDPqSMB36ALH8zdlswZ/t0wR/TogCijVEUnQ+1ERmNHAgMBAAEwDQYJKoZIhvcN
AQEFBQADgYEAOb/KlJK2nQ5VTgC/wZ3G+3Ft4gRNv8iazjRs0N3CX/zAMGxV1ECkveLunBrCn/SN
Mqq9WMVvzJtDbcxU1cE75Ncng893QlfnboShpVsXeH6gTT5saCsleod+VoqBaktus6QuSN1JUz+i
NSr1SNzbyoZiPq/UgexQMxFqowA3PDI=
-----END CERTIFICATE-----
```

Alias for certificate:

[Cancel](#)

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

Keystore Configuration

[view]

This screen lists the contents of a keystore.

| | Alias | Type | Certificate Fingerprint |
|----------------------|---------------------------------------|---------------------|---|
| view | My Own CA Certificate | Trusted Certificate | 04:26:F4:53:8A:40:88:3C:8D:00:30:C0:80:08:E6:33 |
| view | My Private Key | Private Key | 81:A8:79:DD:28:E3:EF:CD:BC:60:DF:DC:C4:9F:E9:24 |

[Add Trust Certificate](#)
[Create Private Key](#)
[Return to keystore list](#)

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: | <input type="text" value="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: | <input type="text" value="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: | <input type="text" value="443"/> | |
| | The network port to bind to. | |
| Max Threads: | <input type="text" value="50"/> | |
| | The maximum number of threads this connector should use to handle incoming requests | |
| SSL Settings | | |
| Keystore File: | <input type="text" value="sr/security/keystores/My_Keystore"/> | |
| | The file that holds the keystore (relative to the Geronimo install dir) | |
| Keystore Password: | <input type="password" value="xxxxxxxx"/> | |
| | 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: | <input type="text" value="JKS"/> | |
| | Set the keystore type. There is normally no reason not to use the default (JKS). | |
| Truststore File: | <input type="text"/> | |
| | The file that holds the truststore (relative to the Geronimo install dir) | |
| Truststore Password: | <input type="password"/> | |
| | Set the password used to verify the truststore file. | |
| Truststore Type: | <input type="text" value="JKS"/> | |
| | Set the truststore type. There is normally no reason not to use the default (JKS). | |
| HTTPS Algorithm: | <input type="text" value="Sun"/> | |
| | Set the HTTPS algorithm. This should normally be set to match the JVM vendor. | |
| HTTPS Protocol: | <input type="text" value="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: | <input checked="" type="checkbox"/> | |
| | 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: | |
| | <ol style="list-style-type: none"> 1. The trust store configured above 2. A keystore file specified by the javax.net.ssl.trustStore system property 3. <i>java-home/lib/security/jssecacerts</i> 4. <i>java-home/lib/security/cacerts</i> | |
| | <input type="button" value="Save"/> | <input type="button" value="Reset"/> <input type="button" value="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