

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](#)

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

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	Type	Contents	Editable	Available
ca-keystore	JKS	Keystore locked		1 key ready
geronimo-default	JKS	Keystore locked		1 key ready
My_Keystore	JKS	0 Keys and 0 Certs		

[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

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: My_Private_Key

Password for new key: *****

Confirm Password: *****

Key Size: 1024

Algorithm: MD5withRSA

Valid for (# of days) : 365

Certificate Identity

Server Hostname (CN): localhost

Company/Organization (O): Apache

Division/Business Unit (OU): Geronimo

City/Locality (L): My_City

State/Province (ST): My_State

Country Code (2 char) (C): CC

Review Key Data

Cancel

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

Keystore Configuration

This screen lists the contents of a keystore.

	Alias	Type	Certificate Fingerprint
View	My_Private_Key	Private Key	EC:E8:B7:EE:8A:0B:0F:38:AF:D7:98:5F:C8:CD:EE:02

[Add Trust Certificate](#)
[Create Private Key](#)
[Change keystore password](#)
[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

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	Type	Contents	Editable	Available
ca-keystore	JKS	Keystore locked		 1 key ready
geronimo-default	JKS	Keystore locked		 1 key ready
My Keystore	JKS	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

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

Keystore	Alias	Type
My_Keystore	My_Private_Key	Private Key

[Generate CSR](#)
[Import CA reply](#)
[Change key password](#)
[Delete Entry](#)
[Back to keystore/a>](#)

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: 120362553328
Valid From: Thu Feb 21 14:25:33 CST 2008
Valid To: Fri Feb 20 14:25:33 CST 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

Keystore: My_Keystore

Alias: My_Private_Key

PKCS10 Certification Request

```

-----BEGIN CERTIFICATE REQUEST-----
MIIBqDCCARECAQAwajESMBAGA1UEAxMJbG9jYWxob3N0MREwDwYDVQQLEwhHZXJvbmltbz
EPMA0GA1UEChMGQXBhY2hlMRAwDgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0
ZTElMAkGA1UEBhMCQ0MwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMfqrJ/aMbVzm
EjDimnMQuVN/CaO7Yb89KP6ed3VQf+/Ea2i+p0dRskM8oNg+3kZeKuOplwq5KGEUnp+xbf
q7M6tLGrWqQ8qL3EzUFE2nizH5VzV093vKu5jgnR2RfbTc2Ap1cldCPofUVuMUbDLPsmE1
YQQR+OchtcNspZL5tdAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAZFuPz0gzKqMZNA0bYLm0
aPFLbR9a19NA0EbgJL2SYzoKnuKyp1G2JzMVQ6myaez0J8t+iWtuthz70kBiRzU2vqOWp
B4oqh+zbPwn4f871418PjJh3SkiDIYdMcL5U1rxwFNAAIEpfjft/uJLY/Bv7DZQG7UPsGz
+SPdn+DbdBo=
-----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
EPMA0GA1UEChMGQXBhY2hlMRAwDgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0
ZTElMAkGA1UEBhMCQ0MwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMfqrJ/aMbVzm
EjDimnMQuVN/CaO7Yb89KP6ed3VQf+/Ea2i+p0dRskM8oNg+3kZeKuOplwq5KGEUnp+xbf q7M6tLGrWqQ8qL3EzUFE2nizH5VzV093vKu5jgnR2RfbTc2Ap1cldCPofUVuMUbDLPsmE1
YQQR+OchtcNspZL5tdAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAZFuPz0gzKqMZNA0bYLm0
aPFLbR9a19NA0EbgJL2SYzoKnuKyp1G2JzMVQ6myaez0J8t+iWtuthz70kBiRzU2vqOWp B4oqh+zbPwn4f871418PjJh3SkiDIYdMcL5U1rxwFNAAIEpfjft/uJLY/Bv7DZQG7UPsGz
+SPdn+DbdBo= -----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-----
MIICQjCCAA2gAwIBAgIBAgjALBgqhkiG9w0BAQQwaDEWMBQGA1UEAxMNMR2Vyb25pbW8ncyBDQTER
MA8GA1UECzMIR2Vyb25pbW8xZDZANBgNVBAoTBkFwYWN0ZTENMA5GA1UEBxMEQ210eTEOMAwGA1UE
CBMFU3RhdGUxCzAJBgNVBAYTA1VTMB4XDTA4MDIyMDAwMFOxDTA5MDIyMTA2MDAwMFowajES
MBAGA1UEAxMJbG9jYWxob3N0MREwDwYDVQQLEwhHZXJvbmltbzEPMA0GA1UEChMGQXBhY2hlMRAw
DgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0ZTELMakGA1UEBhMCQ0MwgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMfqrJ/aMbVzmEjDimnMQuVN/CaO7Yb89KP6ed3VQf+/Ea2i+p0
dRskM8oNg+3kZeKuOplwq5KGEUnp+xbfq7M6tLGrWqQ8qL3EZUFE2nizH5VzV093vKu5jgnR2Rfb
Tc2AplcldCPofUVuMUbdLPSmE1YQQR+OcHtcNspZL5tdAgMBAAEwCwYJKoZIHvCNAQEEA4GBAB9s
1QuMD+dNe6H6XcizZSxNPOh1EocjGp05Z4VOpGFnB4gVRqJqyxuNqCBPvEo30IuHNJZOm6jFhGs
YWKGlzL1zw0yXWAVRnI7Cs8C7Ibeoo+I4yBA93w3XyGiB1Sb03yHOiCN06bf7BhCN6Z45NMhBGbP
pCpnP+uM9VI2gn9H -----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

Keystore: My_Keystore
Alias: My_Private_Key

PKCS7 Certificate Reply

```
-----BEGIN CERTIFICATE-----
MIICQjCCAA2gAwIBAgIBAgjALBgqhkiG9w0BAQQwaDEWMBQGA1UEAxMNMR2Vyb25pbW8ncyBDQTER
MA8GA1UECzMIR2Vyb25pbW8xZDZANBgNVBAoTBkFwYWN0ZTENMA5GA1UEBxMEQ210eTEOMAwGA1UE
CBMFU3RhdGUxCzAJBgNVBAYTA1VTMB4XDTA4MDIyMDAwMFOxDTA5MDIyMTA2MDAwMFowajES
MBAGA1UEAxMJbG9jYWxob3N0MREwDwYDVQQLEwhHZXJvbmltbzEPMA0GA1UEChMGQXBhY2hlMRAw
DgYDVQQHDAdNeV9DaXR5MREwDwYDVQQIDAhNeV9TdGF0ZTELMakGA1UEBhMCQ0MwgZ8wDQYJKoZI
hvcNAQEBBQADgY0AMIGJAoGBAMfqrJ/aMbVzmEjDimnMQuVN/CaO7Yb89KP6ed3VQf+/Ea2i+p0
dRskM8oNg+3kZeKuOplwq5KGEUnp+xbfq7M6tLGrWqQ8qL3EZUFE2nizH5VzV093vKu5jgnR2Rfb
Tc2AplcldCPofUVuMUbdLPSmE1YQQR+OcHtcNspZL5tdAgMBAAEwCwYJKoZIHvCNAQEEA4GBAB9s
1QuMD+dNe6H6XcizZSxNPOh1EocjGp05Z4VOpGFnB4gVRqJqyxuNqCBPvEo30IuHNJZOm6jFhGs
YWKGlzL1zw0yXWAVRnI7Cs8C7Ibeoo+I4yBA93w3XyGiB1Sb03yHOiCN06bf7BhCN6Z45NMhBGbP
pCpnP+uM9VI2gn9H
-----END CERTIFICATE-----|
```

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

Keystore

Alias

Type

My_Keystore My_Private_Key Private Key

[Generate CSR](#)
[Import CA reply](#)
[Change key password](#)
[Delete Entry](#)
[Back to keystore/a>](#)

Certificate Info

Version:

3

Subject:

C=CC, ST=My_State, L=My_City, O=Apache, OU=Geronimo, CN=localhost

Issuer:

C=US, ST=State, L=City, O=Apache, OU=Geronimo, CN=Geronimo's CA

Serial Number:

2

Valid From:

Wed Feb 20 00:00:00 CST 2008

Valid To:

Sat Feb 21 00:00:00 CST 2009

Signature Alg:

MD5withRSA

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-----
MIICJTCCAZCgAwIBAgICK2cwCwYJKoZIhvcNAQEEMF0xDTALBgNVBAMTBFRlc3QxZDZANBgNVBAST
BkFwYWNoZTERMA8GA1UEChMIR2Vyb25pbW8xZCZAJBgNVBACTAkxMMQswCQYDVQQLIEwJTVDDELMAkG
A1UEBhMCTEswHhcNMDCwMjAyMTgwMDAwWhcNMDCwMjAyMTgwMDAwWjBaMQ0wCwYDVQQDEWRUZXR0
MQ8wDQYDVQQLEwZBcGFjaGUxETAPBgNVBAoTCEDlcm9uaW1vMQswCQYDVQQHEwJMTDELMAkGA1UE
CBMCU1QxZCZAJBgNVBAYTAkxLMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCUI1e1eTKLoh0
15vfYqqvhk6lviva7BWQxZ6mOV9Ye2mii37Btmxajnnngz0jKfiwHKqWRQBp6CUzbd9gfZrz2go9g
TwsUBWQwSf6iVypKX1q0Y4WhtTwLcEx78Lx5XN1YCqk34pn4by26SjiHdugs7/CiOillcpCt9QVa
Q9BH7wIDAQABMA5GCSqGSIb3DQEBAQOBgQANmoT
/dLvJa7JGstvZJLrsWtMwWQNVJ1ZQmbrDGq9u oFnkAH1mGHIDbaz2avy/wotHJUlysGBIDP0btk5GVskl45EG/feWHLgCVmqwf3NkdRdLI+CznBBJ
KCC5tINbcl6GqXsbO8hhjlrOGweNyV1653WEvZiQVuMYaHTnGNx+RA== -----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-----
MIICJTCCAZCgAwIBAgICK2cwCwYJKoZIhvcNAQEEFQxDTALBgNVBAMTBFRlc3QxDzANBgNVBAsT
BkFwYWN0ZTERMA8GA1UEChMIR2VyY25pbW8xCzAJBgNVBACTAkxMMQswCQYDVQIEwJTVDLMakG
A1UEBhMCTESwHhcNMDcwMjAyMTgwMDAwWWhcNMDgwMjAyMTgwMDAwWjBaMQowCwYDVQQDEwRUZXN0
MQ8wDQYDVQLEwZBcGFjaGUxETAPBgNVBAoTCedlcm9uaWlvMQswCQYDVQQHEwJMTDELMAkGA1UE
CBMCU1QxCzAJBgNVBAYTAkxLMiGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCUCZ1le1eTKLoh0
15vfYqqvkh6Iviva7BNQxZ6mOV9Ye2mii37Btmxajnnzg0jKfiwHKqWRQBp6CUzbd9gfZrz2go9g
TwsUBWQwSf6iVypKXlq0Y4WhtTwLcEx78Lx5XN1YCqk34pn4by26SjiHdugs7/C1oiIlcpCt9QVa
Q9BH7wIDAQABMA5GCSqGSIb3DQEBAQBAOBgQANmoT/dLvJa7jGstVZJLrsWtMwWQNVJ1ZQmbrDGq9u
oFnKAHlmGHIDbaz2avy/wotHJUIysGBLDP0btk5GVsk145EG/feWHLgCvMqwf3NkdRdLl+CznBBJ
KCC5tINbcI6GqXsb08hhjIroGweNyV1653WEvZiQVumYaHTnGNx+RA==
-----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_Trust_CA	Trusted Certificate	E7:04:A8:42:24:58:7C:E5:CC:FD:71:0C:44:A6:01:00
view	My_Private_Key	Private Key	45:2C:C9:23:2A:07:83:23:45:68:08:7D:53:C2:B8:C2

[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



Name	Protocol	Port	State	Actions	Type
TomcatWebConnector	HTTP	8080	running	Stop Restart Edit Delete	Tomcat Connector HTTP BIO
TomcatWebSSLConnector	HTTPS	8443	running	Stop Restart Edit Delete	Tomcat Connector HTTPS BIO
TomcatAJPConnector	AJP	8009	running	Stop Restart Edit Delete	Tomcat Connector AJP

Add new:

- [Tomcat BIO HTTP Connector](#)
- [Tomcat BIO HTTPS Connector](#)
- [Tomcat NIO HTTP Connector](#)
- [Tomcat NIO HTTPS Connector](#)
- [Tomcat AJP Connector](#)

Notes:

- To enable APR connectors creation, you need to install APR native library firstly.

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

Network Listeners



Add a newTomcat NIO HTTPS Connector

(* denotes a required attribute)

Attribute	Type	Value	Description
*Unique Name	String	<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)
*address	String	<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).
*keystoreFile	String	<input type="text" value="/var/security/kestores/My_keystore"/>	The file that holds the keystore (relative to the Geronimo install dir)
*port	Integer	<input type="text" value="8443"/>	The TCP port number on which this Connector will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address.
acceptCount	Integer	<input type="text" value="10"/>	The maximum queue length for incoming connection requests when all possible request processing threads are in use. Any requests received when the queue is full will be refused.
acceptorThreadCount	Integer	<input type="text" value="1"/>	The number of threads to be used to accept connections. Increase this value on a multi-CPU machine, although you would never really need more than 2. Also, with a lot of non-keep-alive connections, you might want to increase this value as well.
acceptorThreadPriority	Integer	<input type="text" value="5"/>	The priority of the acceptor threads. The threads used to accept new connections.
algorithm	String	<input type="text" value="IbmX509"/>	The certificate encoding algorithm to be used.
allowTrace	Boolean	<input type="checkbox"/>	A boolean value which can be used to enable or disable the TRACE HTTP method.
bufferSize	Integer	<input type="text" value="2048"/>	The size (in bytes) of the buffer to be provided for input streams created by this connector.
ciphers	String	<input type="text"/>	A comma separated list of the encryption ciphers that may be used. If not specified, then any available cipher may be used.
clientAuth	Boolean	<input checked="" type="checkbox"/>	Set to true if you want the SSL stack to require a valid certificate chain from the client before accepting a connection. Set to want if you want the SSL stack to request a client Certificate, but not fail if one isn't presented. A false value (which is the default) will not require a certificate chain unless the client requests a resource protected by a security constraint that uses CLIENT-CERT authentication.
command_line_options	Boolean	<input checked="" type="checkbox"/>	The following command line options are available for the NIO connector: -Dorg.apache.tomcat.util.net.NioSelectorShared=true/false. Set this value to false if you wish to use a selector for each thread. the property. If you do set it to false, you can control the size of the pool of selectors by using the selectorPool.maxSelectors attribute

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

Select the **ClientAuth** 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 point you have not configured your client with a valid certificate.