



GRANTS.GOVSM

**WEB SERVICES
SECURITY**

Version 1.2

March 27, 2005

*This report is confidential and intended solely for the use and
information of the client to whom it is addressed.*

1.0 Introduction	4
2.0 Secure Socket Layer (SSL) and Digital Certificate	4
3.0 Agency System to Grants.gov Authentication	5
4.0 Client Certificate Requirements for Mutual Authentication	7
4.1 Client Certificate Requirements for Acceptance Testing	8
4.2 Client Certificates Requirements for Production	8
4.2.1 Configuring Bouncy Castle for JSSE.	9
4.3 Generating RSA KeyPairs and Certificates	9
4.3.1 Using Portecle to Generate a RSA Keypair	9
4.3.2 Configuring Mutual Authentication for the Microsoft .NET Platform	15
5.0 Obtaining a CA Signed Certificate	18
5.1 Using Portecle to Generate a Certificate Signing Request (CSR)	18
5.2 Importing the CA Reply	19
6.0 Grants.gov Hashing Standard	21
6.1 Preparing and Hashing Attachments	21
6.2 Hashing the Grant Forms XML	21
Appendix I – FAQs	22

Record of Changes

Version Number	Date of Change	Made By	Description of Change
1.0	03/20/2004	Mark Sommer	Initial Revision
1.1	02/28/2005	Mark Sommer	Updated KeyTool GUI information to Portecle
1.2	03/29/2005	Mark Sommer	Added section on hashing

1.0 Introduction

This document provides agencies with details on the Grants.gov security model and instructions on how to create digital certificates complying with the Grants.gov mutual authentication requirements.

2.0 Secure Socket Layer (SSL) and Digital Certificate

SSL is a commonly used protocol for managing the security of a message transmission on the Internet. SSL contains its own server-side authentication and an optional client authentication. SSL is a framework that specifies how two hosts connect, pass certificate between one another, negotiate which keys to use, and decide on an encryption mechanism.

SSL comes in two strengths, 40-bit and 128-bit, which refer to the length of the "session key" generated by every encrypted transaction. The longer the key, the more difficult it is to break the encryption code. Most browsers support 40-bit SSL sessions, and the latest browsers enable users to encrypt transactions in 128-bit sessions - trillions of times stronger than 40-bit sessions.

Encryption is performed in two ways: symmetric and asymmetric. In symmetric cryptography, the sender and the receiver share the same private key to encrypt the information that is sent between them. In asymmetric cryptography, the sender and the receiver have different private keys. The length of a private key can range from 512bit to 2048bit. The larger a key length, the more difficult it is to derive the private key from the public key, hence, making it more secure and less likely to be spoofed.

When HTTP is passed over the SSL infrastructure, HTTP can use all features of SSL. This combination of HTTP and SSL is commonly used for secure Internet systems and it is referred to as HTTPS.

SSL is normally used in tandem with a Digital Certificate. This means that it uses a third party, a Certification Authority (CA), to identify one end or both ends of the transactions. Digital certificates are electronic files used to uniquely identify people and resources over the Internet. They contain information about the ownership of the certificate as well as information about the issuing CA. SSL also provides a legal basis to perform transactions on the Internet and is an integral part of the process for establishing the validity of digital certificates.

Virtually all web servers and the leading browsers, including Internet Explorer and Netscape Communicator, are optimized and ready for SSL. To activate SSL sessions a digital certificate is required.

3.0 Agency System to Grants.gov Authentication

Grants.gov uses SSL with mutual authentication to establish a secure connection with agency systems. To do so, Agencies are required to obtain and install a digital certificate on their application server. This certificate will be used to authenticate the agency server when sending requests to Grants.gov over SSL.

Once the authentication step is completed and a secure connection is established, the Grants.gov system conducts an internal authorization process to ensure that the requesting agency server has been assigned the appropriate roles before performing the requested transaction.

The requirements for authenticating with Grants.gov are described below:

- Registering the agency system with Grants.gov as an agency user. To create a user profile for the system, the agency must provide the following information:
 - Name and Telephone Number of the agency's Point of Contact
 - Information included in the client Digital Certificate, to include its serial number, agency name, common name, and certification authority. This registered certificate must be installed on the agency server that will be making the https:// calls to the Grants.gov Web Services.
- Assigning roles to the agency system profile. Once the Grants.gov team has created a user profile for the agency's server certificate, an agency administrator must login to Grants.gov and assign roles to the certificate's profile. This step provides an additional measure of security by restricting the agency's Web Services account to the minimum required roles. Below are the steps for assigning those roles:
 - Log into the 'For Grantors' section of the site
 - Click on the 'Manage Agency Users' link on the left navigation bar
 - Select the check box next to the Client Certificate's profile and click the 'Reassign Roles' button

Manage Agency Users

Reassign Roles	Modify User	Delete User	Refresh
Organization Users			
<input checked="" type="checkbox"/>	ED Client Server		
<input type="checkbox"/>	ED Tester		
Check All Clear All			

- Select the 'View Applications', 'Agency Tracking Number Assigner', and 'Agency Grant Retriever' from the list of Remaining Roles.

Current Roles: No remaining Roles	Remaining Roles: View Applications Manage Agencies Agency Template Creator Agency Tracking Number Assigner Agency Grant Retriever Agency Grant Creator
<input type="button" value="Continue"/>	<input type="button" value="Cancel Reassign"/>

- Click the '<<' button to move the selected roles over to the 'Current Roles' box and then click the 'Continue' button to assign the roles.

After the certificate has been registered with Grants.gov, the agency is ready to establish a secure layer between an agency system and Grants.gov:

- The agency sends a request via an https:// call to a Grants.gov web server that includes a copy of its public certificate. The acceptance testing environment address is <https://atws.grants.gov:446> and the production site is <https://ws.grants.gov:446>. **Note:** the server that will be conducting the Web Service calls must have access to **port 446**.
- Next, Grants.gov validates the agency certificate serial number against the information in the agency profile. It also checks the valid period and the CA of the certificate. If validation fails, the SSL will not be established, otherwise, the following steps will occur.
 - Grants.gov web server presents the agency with its public key certificate.
 - If the agency accepts the certificate, it uses the public key to encrypt the data it is about to send.

- Grants.gov using its own certificate private key decrypts the sent encrypted data and a secure connection is established. The subsequent request/response message exchanges occur over the secure socket.

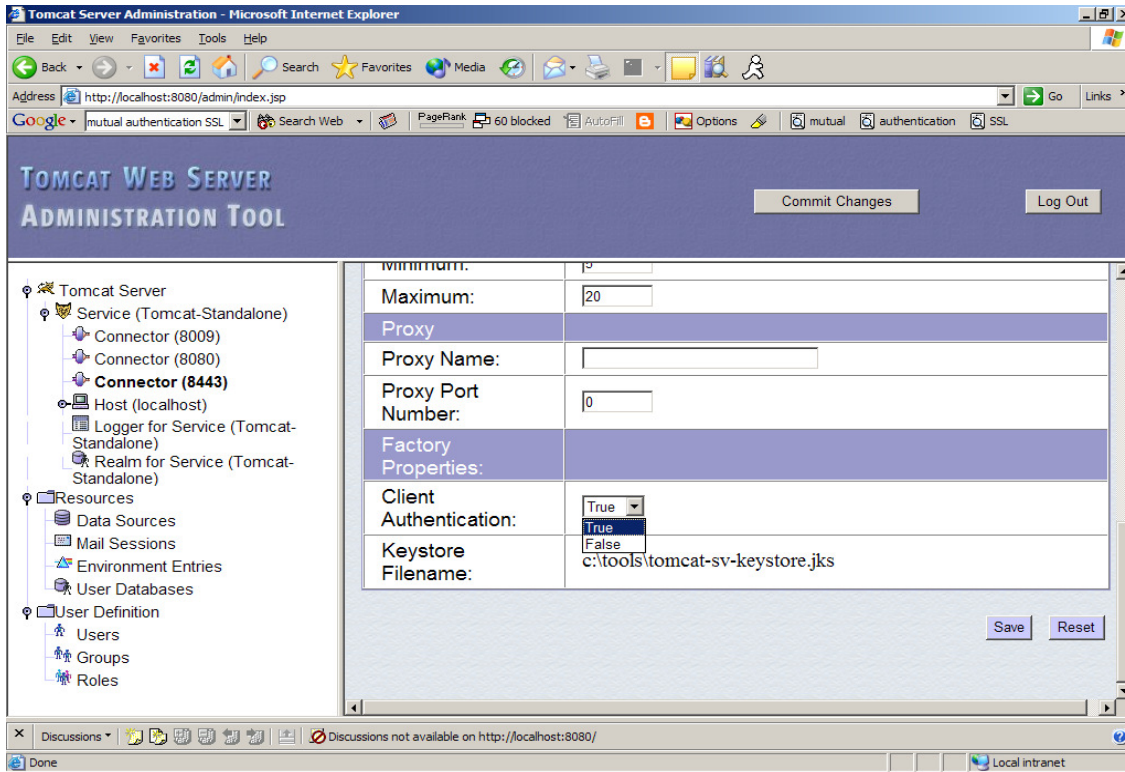
Once the authentication step is completed and a secure connection is established, the Grants.gov system conducts an internal authorization process to ensure that the requesting agency server has been assigned the appropriate roles before performing the requested transaction.

4.0 Client Certificate Requirements for Mutual Authentication

Each agency shall provide a digital certificate to interact with Grants.gov. In addition, Grants.gov requires mutual authentication to complete the SSL handshake. Mutual authentication will require the client application to authenticate itself by providing its server certificate. Grants.gov only supports certificates that were generated using the **RSA** algorithm and signed by a recognized CA.

Grants.gov recommends that developers test mutual authentication with the reference implementation. Testing with the reference implementation allows developers to minimize the time required to flush out potential SSL handshake issues within client applications. To test mutual authentication with the reference implementation it will be necessary to set the HTTPS connector's "Client Authentication" property to "True" in the Tomcat administration tool. Please refer to the reference implementation instructions for more information on the Tomcat setup.

Note: Tomcat and Sun's default JCE provider for JSSE does not support 1024 bit RSA. Ensure when creating an RSA keypair to use 2048 bit key sizes (or see [Configuring Bouncy Castle for JSSE](#)).



4.1 Client Certificate Requirements for Acceptance Testing

- The agency shall provide Grants.gov a copy of their public certificate.
- Certificates shall be created using 1024 or 2048 bit RSA.
- Grants.gov recommends testing with certificates signed by a CA (see [5.0 Obtaining a CA Signed Certificate](#)); but if the agency prefers to test with self-signed certificates, it must provide Grants.gov a copy of the public certificate in Base64/ PEM format.

4.2 Client Certificates Requirements for Production

- Certificates shall be created using 1024 or 2048 bit RSA.
- A well-know public CA must sign the client certificate for production usage. Listed below are several popular CAs:
 - Access Certificates For Electronic Services (ACES) - <http://www.trustdst.com/federal/aces.html>
 - Entrust - <http://www.entrust.com>
 - Thwate - <http://www.thawte.com>
 - VeriSign - <http://www.verisign.com>
- Self-signed certificates may not be used in Production.

4.2.1 Configuring Bouncy Castle for JSSE.

The Sun default JCE provider packaged with J2SE 1.4 currently does not support 1024 bit RSA certificates. Agencies using JSSE with 1024-bit RSA must, therefore, use an alternate JCE provider to enable mutual authentication with Grants.gov. One option that has been used with success by other agencies is Bouncy Castle (<http://www.bouncycastle.org/>). Bouncy Castle provides a free implementation of the JCE framework that is compatible with many versions of the J2SE. Agencies may use other providers capable of providing the same functionality.

Installation and configuration of Bouncy Castle is very simple. For detailed instructions see the following URL (<http://www.bouncycastle.org/specifications.html>).

4.3 Generating RSA KeyPairs and Certificates

There are many tools available for applicants to generate the required RSA keypair including Microsoft IIS, OpenSSL, Keytool, and many others.

Grants.gov provides reference instructions for Portecle to generate the applicant system's RSA keypair and client certificate.

Portecle (formerly know as KeyTool GUI) is an open source GUI for the java keytool. It can be downloaded at: <http://sourceforge.net/projects/portecle/>

The Java 2 Standard development kit is also required and may be downloaded at: <http://java.sun.com/>

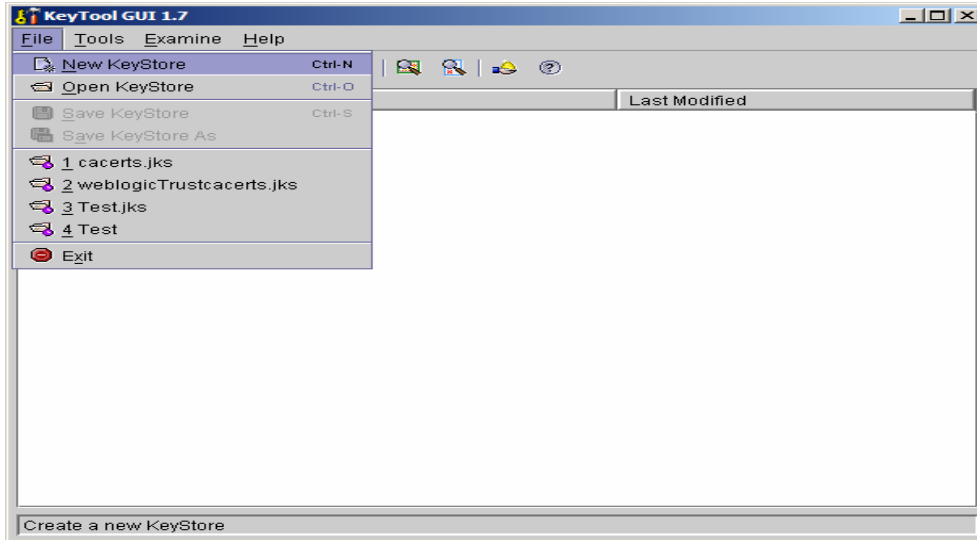
4.3.1 Using Portecle to Generate a RSA Keypair

Portecle is a GUI version of the command-line keytool provided with the JAVA SDK. Portecle can be downloaded at the following URL: <http://sourceforge.net/projects/portecle>

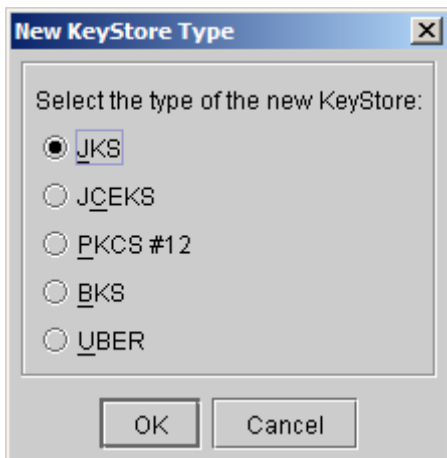
Follow the instructions below to generate a Keystore, RSA keypair, and a digital certificate. Upon completion the instructions below, applicant systems can expect to a have fully compliant digital certificate for Grants.gov Web Services.

Open Portecle and follow the steps below:

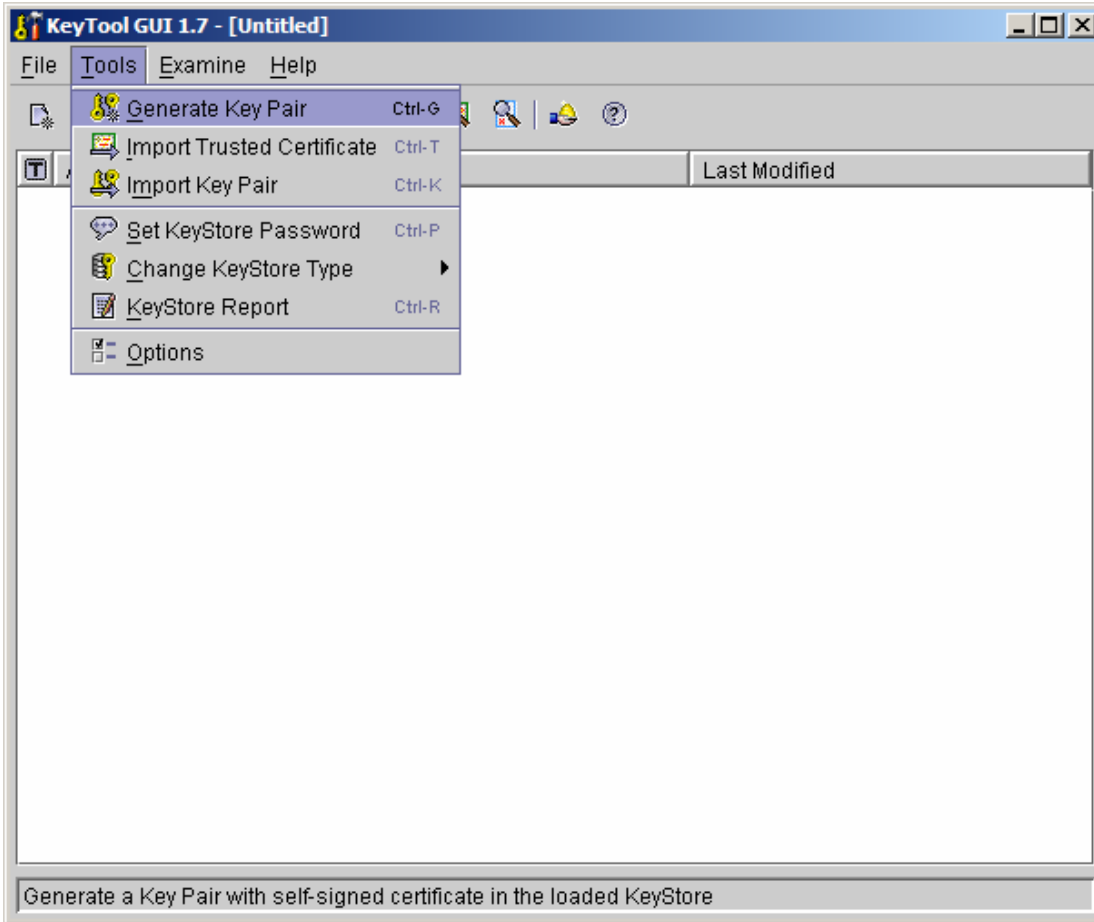
1. Create a new keystore. Select File→New KeyStore.



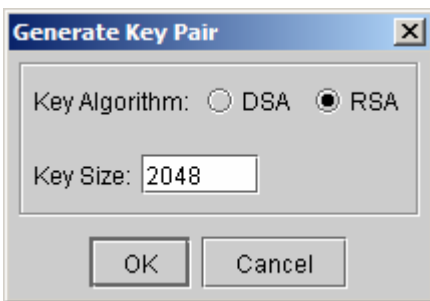
2. Select the JKS Format (for Java implementations) and click OK.



3. Select Tools→Generate KeyPair



4. Select 2048 bit RSA and click OK.



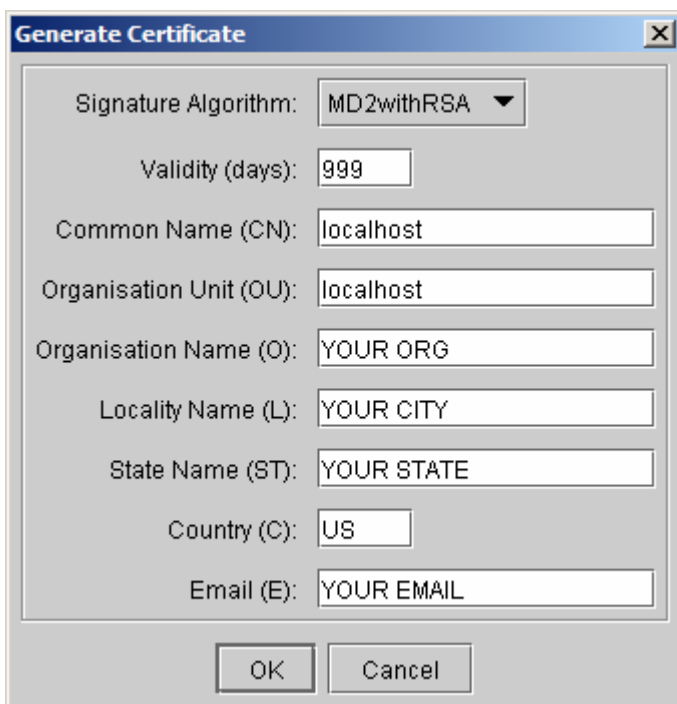
5. The RSA keypair will be generated.



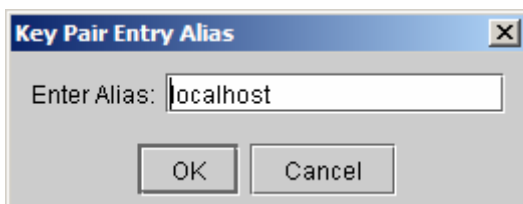
6. Select a Signature Algorithm and enter the information specific to the organization.

Notes: Microsoft .NET users should select MD5 with RSA.

The Common Name must be set to the server's DNS name.

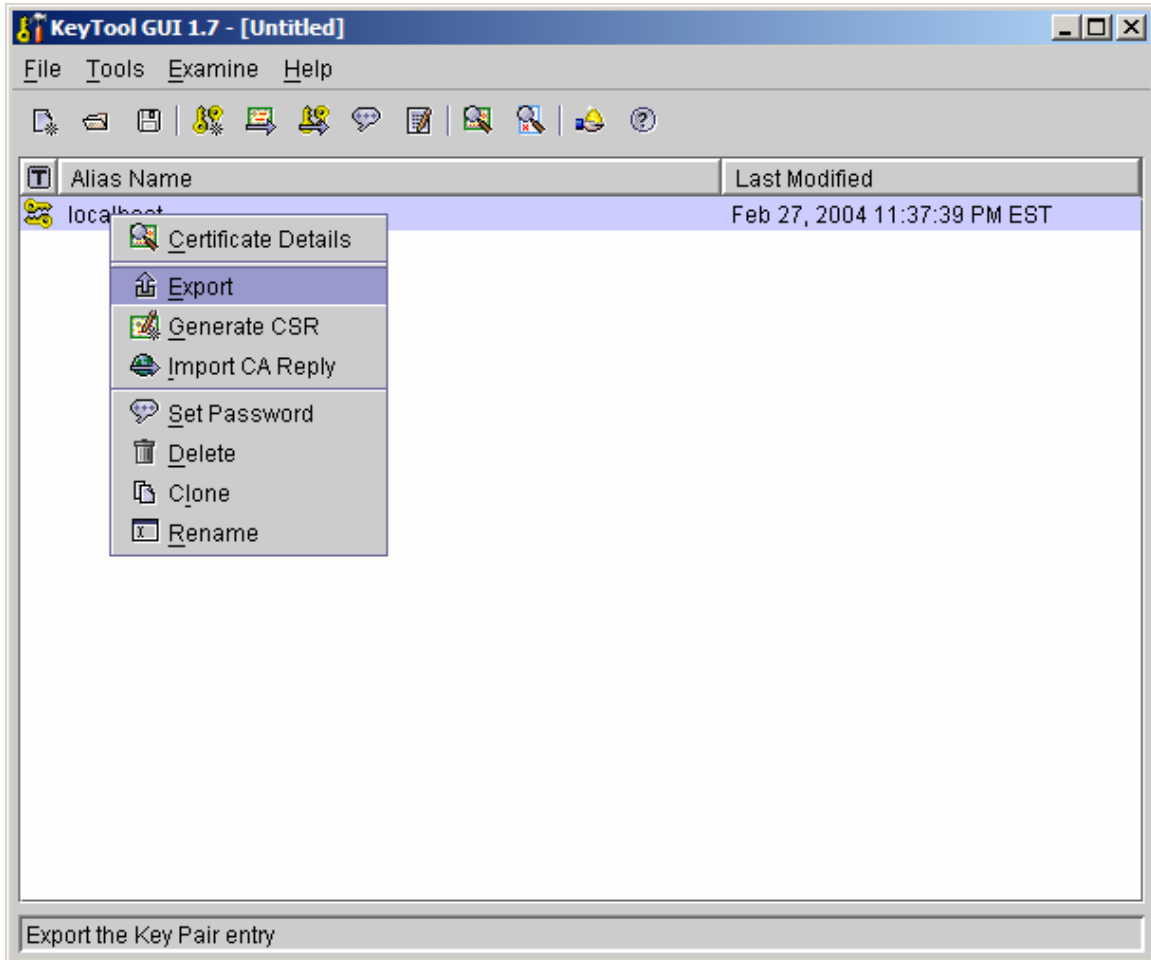


7. Enter the alias.

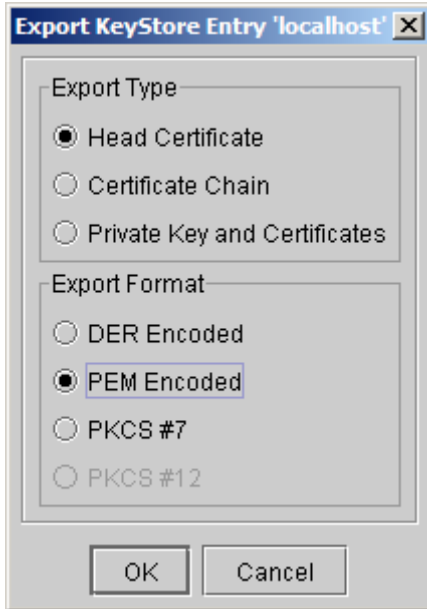


8. Create a password and click OK.

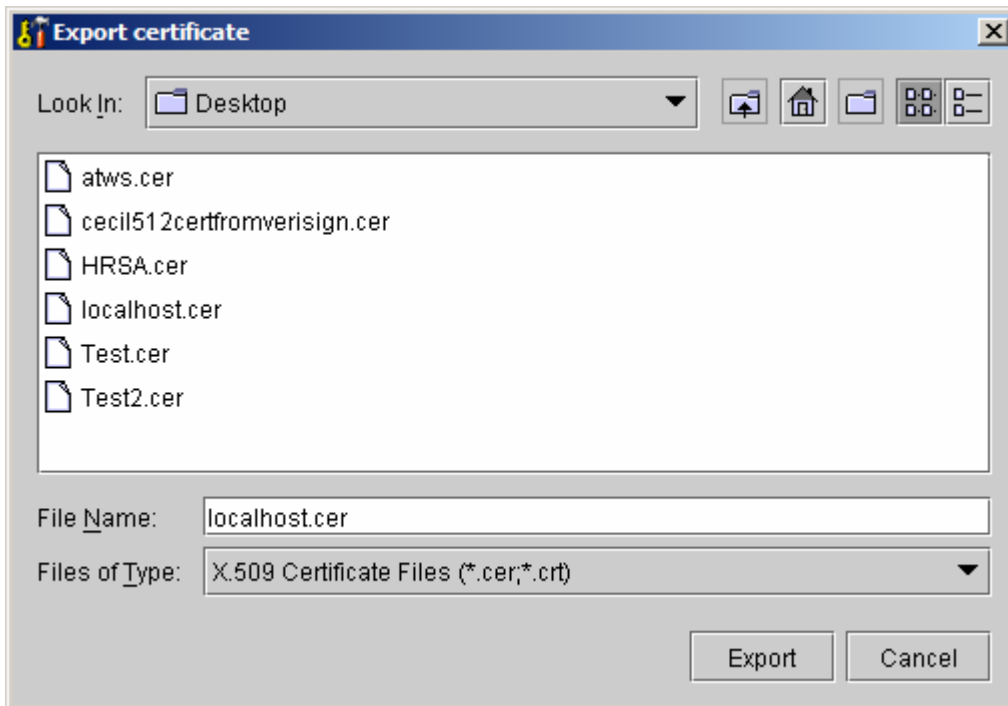
9. Export a client certificate for Grants.gov. Right click the keypair generated in step 8 and select Export.



10. Export the Head Certificate and Select the PEM Encoded Export Format.



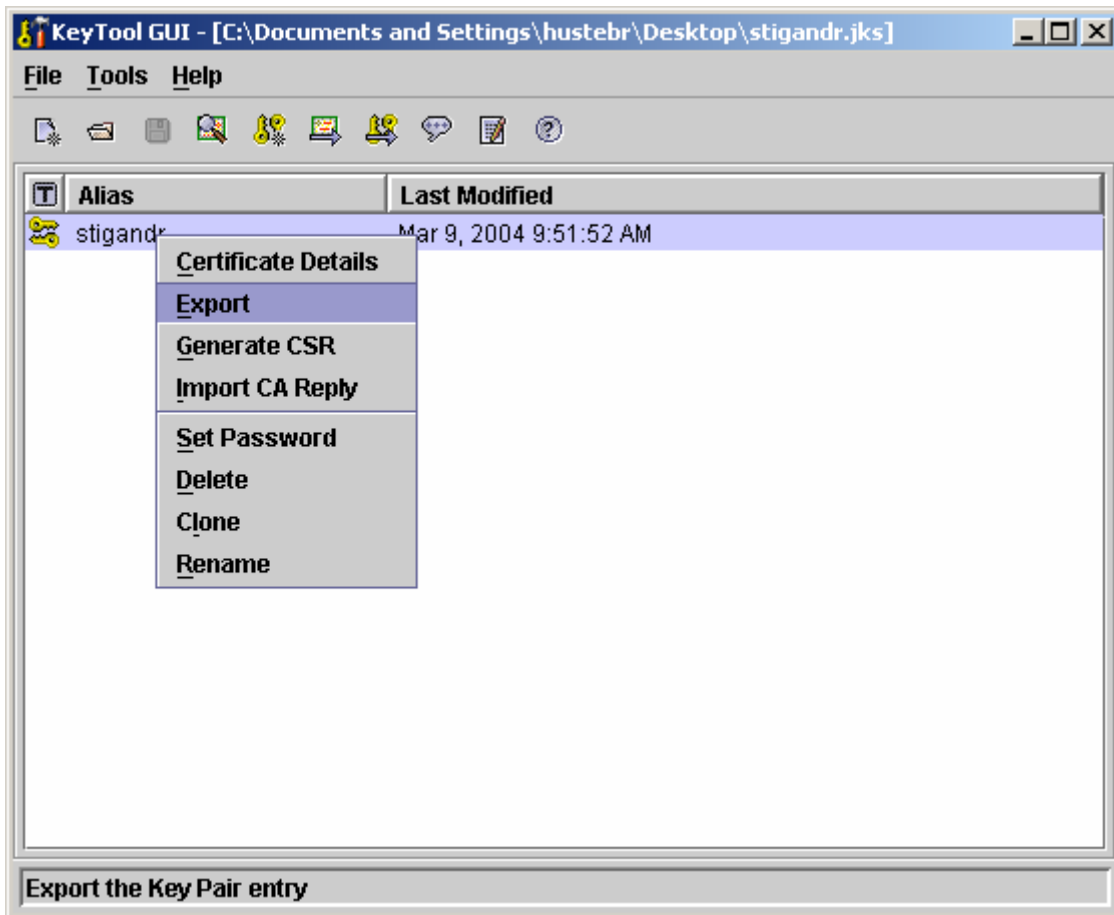
11. Save and name the client certificate.



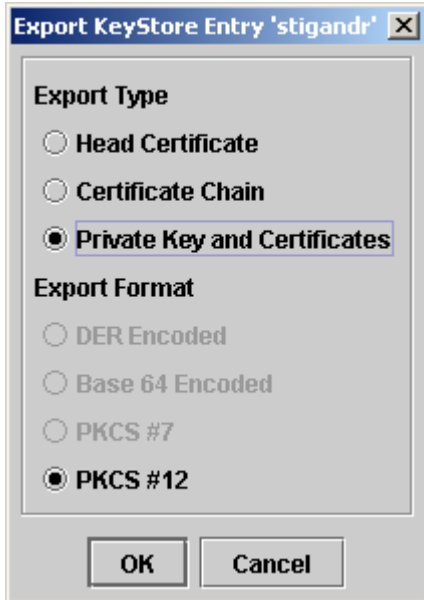
At this point, the client application will need access to the keystore generated from above. The public certificate created above should be sent to Grants.gov WebServices team. The WebServices team will add the certificate to the Grants.gov truststore, register its serial number and associate it with the agency's profile.

4.3.2 Configuring Mutual Authentication for the Microsoft .NET Platform

1. Follow the [instructions for creating a Key Pair](#) from above and be sure to select *MD5 with SHA* as the signature algorithm.
2. Once the certificate has been created, right click on the public private key pair and select export.



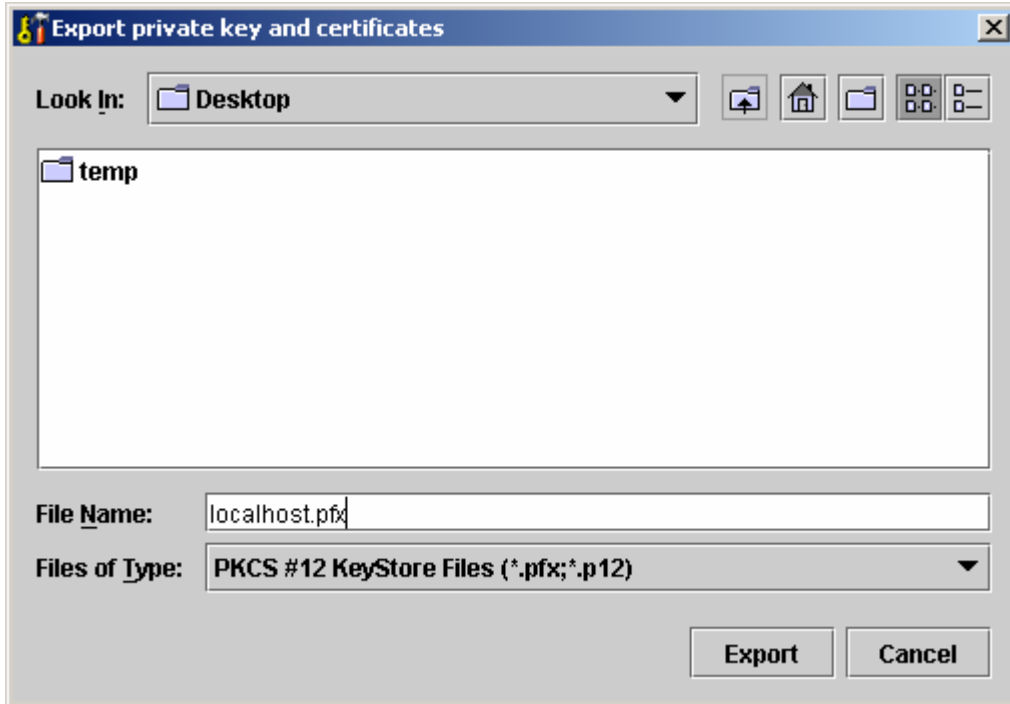
3. Export the Private Key and Certificates using PKCS #12 format



4. Enter a new password for the PKCS #12 keystore



5. Save the newly created PKCS #12 keystore with a .pfx extension to the local drive.



6. Double click on the .pfx file that was just created, this will open up and pop up which will walk you through the installation process for the certificate.
7. Once the certificate has been installed, click on START and then RUN and then type MMC
8. When the MMC opens, click on the CONSOLE menu and then click on the Add/remove Snap In.
9. Click Add and then Choose Certificates.
10. Click Add and then Choose Computer Account and then local computer
11. Once you see the certificates under the MMC, open up the certificates and here you will see the various certificates that are installed.
12. Right click on the certificate that has to be exported and go to all tasks and then export.
13. Select no when prompted if the private keys should be exported

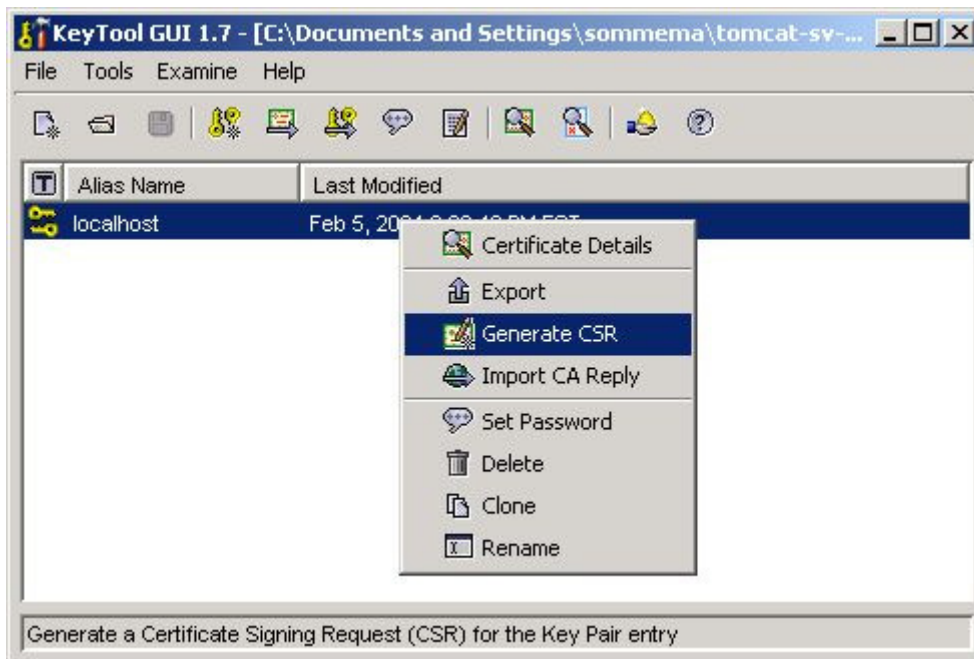
14. Next select BASE64 encoded, as opposed to DER encoded, as the export format of the public certificate.
15. Save this to a file with a .CER extension and forward to the Grants.gov Web Services team.

5.0 Obtaining a CA Signed Certificate

To receive a certificate digitally signed by a CA, you must send a Certificate Signing Request (CSR) to the CA. A CSR is a text file containing information about the requester's server's client public certificate. The CA will use the CSR to generate and send you your signed digital certificate.

5.1 Using Portecle to Generate a Certificate Signing Request (CSR)

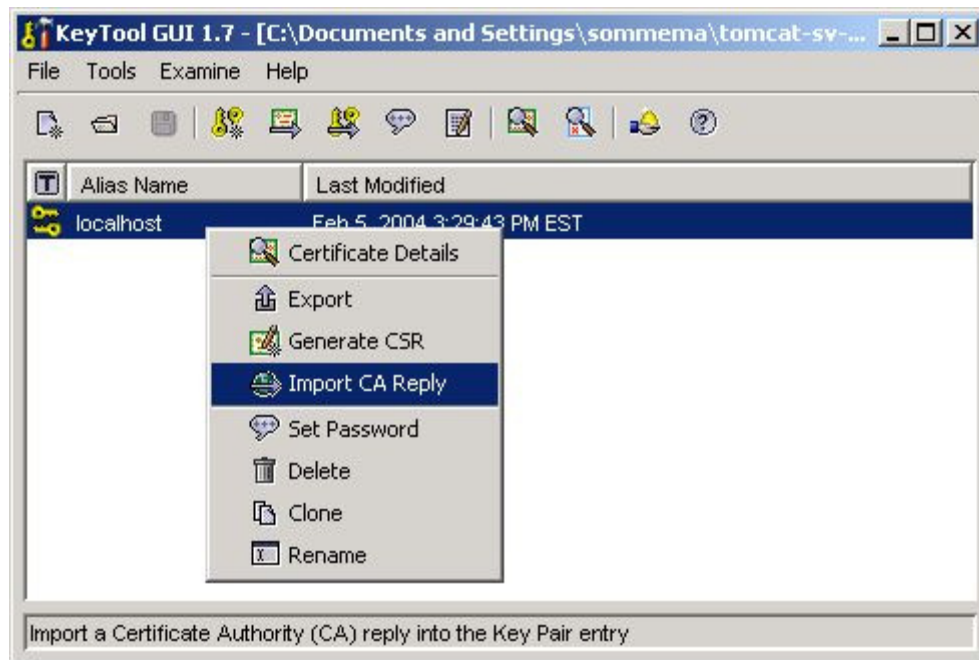
1. Start Portecle and open up the keystore created in the [Using Portecle to Generate a RSA Keypair](#) section.
2. Right click the mouse on the key pair and select Generate CSR



3. Save the generated CSR file with a .csr extension and submit it to your selected CA.

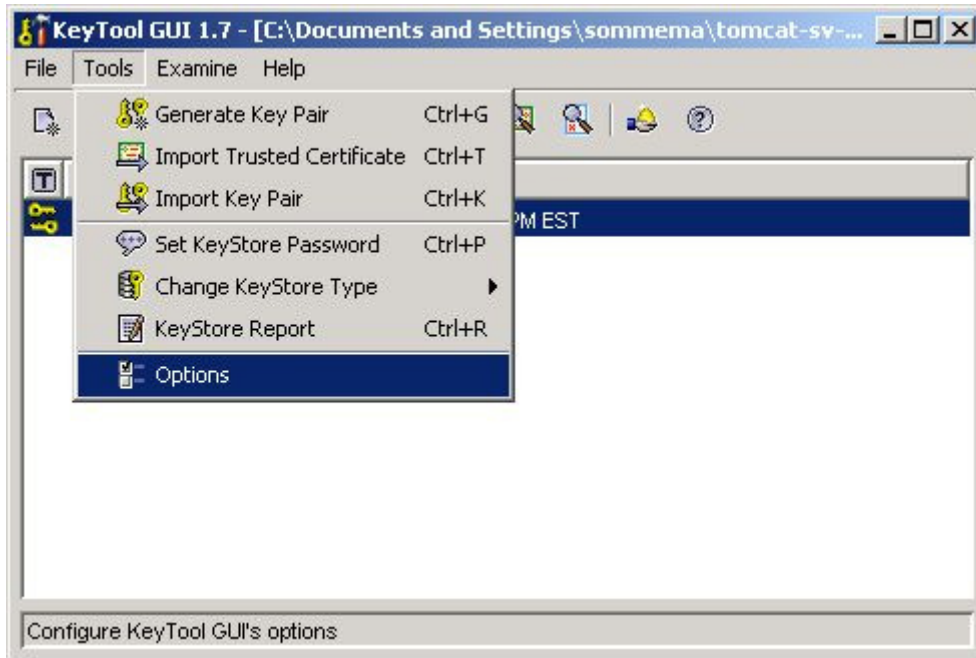
5.2 Importing the CA Reply

1. Once you have received your signed digital certificate from the CA, you must import the certificate into your keystore. To do this, use Portecle to open your original keystore that was used to generate the CSR.
2. Right click on the key pair and select Import CA Reply.

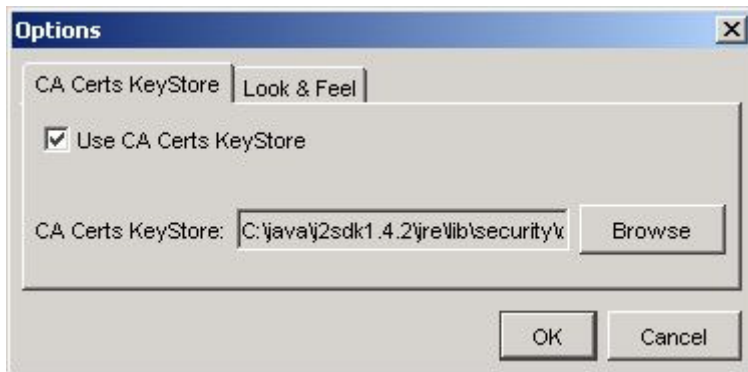


3. Select the signed certificate you received from your CA, click OK, and save the changes to the keystore. Your public/private key pair has now been digitally signed by the CA.

Note: You may need to set the location of the Java cacerts file that contains the root certificates of most signing authorities. Portecle uses these certificates to verify your signed certificate. This setting can be found under Tools > Options.



Typically the cacerts file can be found in <Java_Home>\jre\lib\security\



6.0 Grants.gov Hashing Standard

Grants.gov uses the Secure Hash Algorithm (SHA-1) (<http://www.itl.nist.gov/fipspubs/fip180-1.htm>) for computing hash values. The resulting hash value shall be encoded using the Base64 data encoding specification (<http://www.ietf.org/rfc/rfc3548.txt>). The resulting value will be populated in the global schema 'HashValue' element. For each submissions, Grants.gov will hash the <grant:Forms> xml node and each SOAP attachment. The next few sections describe the process in greater detail.

6.1 Preparing and Hashing Attachments

The following sub-section is intended to provide clarification on the hashing process for applicant Attachments. The Attachments schema is required any time an attachment is submitted with an electronic grant application package. The schema can be accessed by the following URL <http://apply.grants.gov/system/schemas/Attachments-V1.0.xsd>. The schema contains a field named 'FileLocation'. This element represents the Content-Id (CID) for the attachment. The href attribute of the 'FileLocation' element will be populated with the appropriate CID. Moreover, the CID in the schema will match the CID contained within the SOAP attachment header of the SOAP response when a GetApplication request is made. The attachment will be hashed using the [Grants.gov hashing standard](#) and the value placed in the 'HashValue' xml element in the attachment xml. For more information on Content-Id and SOAP attachments, please refer to the following links below:

RFC 2111 - <http://www.ietf.org/rfc/rfc2111.txt>

IBM Article - <http://www-106.ibm.com/developerworks/xml/library/x-tippass.html>

6.2 Hashing the Grant Forms XML

Once the Grant Application XML document is prepared, the hash value of the XML Forms will be computed using the [Grants.gov hashing standards](#). The hash value should be computed over the “grant:Forms” sub-element and not the entire application XML document. The <grant:Forms> element and it’s sub-elements shall be canonicalized before hashing to guarantee equivalence of hash values for “logically” equivalent Grant application XML documents. The canonicalization standard that will be used is the “Exclusive XML Canonicalization” W3C specification (<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>) because the hash is taken over a subset of nodes. This specification, when applied, will produce XML that has the exact same lexical structure for all XML Node inputs that are “logically” equivalent. The specification will not include namespace specific attributes that are in ancestor Nodes of canonicalized sub-elements. Nor will it include the namespace declarations of ancestor Nodes that are not used by the sub-elements to be canonicalized. This shields the

canonicalized sub-elements from being affected by namespace declarations in ancestor Nodes that are not to be canonicalized.

There are three important points to take note of in the exclusive canonicalization specification: 1) white spaces between XML elements will not be normalized; 2) different namespace prefixes that are bound to the same namespace are not resolved; 3) comments in the XML are excluded. In the case of the first point, agency users must recognize that white spaces between XML elements are “retained” by canonicalization and thus will produce different hash values if white space values are changed. The hash value will also differ if namespace prefixes do not stay consistent between the times when the agency retrieves the Grant Application XML from Grants.gov and when it validates the hash value.

The resulting hash value computed using the [Grants.gov hashing standard](#) shall be populated in the ‘HashValue’ element within the Header Schema. Lastly, the entire XML document, including the Form Schemas and Header Schema, will be canonicalized, hashed, and populated in the ‘HashValue’ element of the Footer Schema.

Appendix I – FAQs

Q1. What types of digital certificates does Grants.gov support?

A1. Grants.gov only supports certificates that use the RSA algorithm. It does not support DSA certificates.

Q2. Where are the SSL URLs for the Grants.gov Web Services?

A2. The Acceptance Testing site where Agencies may conduct testing is located at <https://atws.grants.gov:446/>. The production site is located at <https://ws.grants.gov:446/>. Note that both sites require you to access them via port 446.

Q3. My agency is not ready to invest in a CA signed certificate; can we begin testing with a self-signed certificate?

A3. Yes, you may use a self-signed certificate in the Grants.gov Acceptance Testing environment. In Fact, testing should not be delayed pending receipt of a CA signed certificate. However, agencies **must** purchase and test a CA signed certificate before moving to production.

Q4. Typically, what should be the length of my private key?

A4. 1024 bit is normally the recommend length

Q5. What is the difference between a keystore and a truststore?

A5. A keystore is the digital file that contains both your private and public key. A truststore is a set of public CA root certificates that are used to validate the authenticity of a CA signed server certificate.

Q6. Why does Grants.gov require a CA signed certificate?

A6. There are two reasons for doing this:

- 1) Grants.gov is only required to maintain the root CA certificates, rather than each Agencies' certificate.
- 2) A CA signed certificate is more secure and difficult to spoof than a self-signed certificate.