edu.Library

CERTivity® KeyStores Manager

June 14, 2013

Document Version 1.2.19
CERTivity® 1.2

edu.Library

Legal Notice

No part of this publication may be reproduced stored in a retrieval system, or transmitted, in any form or by any means, without the prior written permission of EduLib S.R.L..

EDULIB S.R.L. MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OR MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

CERTivity IS A REGISTERED TRADEMARK OF EDULIB S.R.L. OTHER PRODUCT NAMES AND SERVICE NAMES ARE THE TRADEMARKS OR REGISTERED TRADEMARKS OF THEIR RESPECTIVE OWNERS AND ARE USED FOR IDENTIFICATION ONLY.

Copyright

2013, EduLib S.R.L.

www.edulib.com

Calea Bucuresti, Bl. 27B, Sc. 1, Ap. 2
Craiova, DJ, 200675, Romania
Phone: +40 351 420970
Fax: +40 351 420971
E-mail: office@edulib.ro

Table of Contents

1 Overview 1

1.1 About 1

1.2 Features Summary 1

1.3 Documentation and Samples 5

2 CERTivity®'s Administrative Details 7

2.1 System requirements 7

2.2 Platforms and Java Virtual Machines 7

2.3 Install and Run 8

2.3.1 Installing CERTivity 8

2.3.2 License Key (File) Registration 9

2.3.3 Running CERTivity 9

2.3.4 Handling Multiple CERTivity versions 10

2.4 Java Virtual Machine settings 10

2.5 Purchase and Licensing Model 11

2.5.1 Payment details 12

2.5.2 What do I get after payment 12

3 CERTivity®'s Menus/Tool bar 14

3.1 File Menu 14

3.2 Edit Menu 15

3.3 KeyStore Menu 15

3.4 Signatures Menu 16

3.5 View Menu 16

3.6 Tools Menu 16

3.6.1 Main Options 17

3.6.2 Trust Path Options 17

3.6.3 Other Options 22

3.7 Window Menu 22

3.8 Help Menu 22
3.9 Contextual Menu 22
3.10 Toolbar 24
4 CERTivity®'s Certificates 26
4.1 Open Certificate 26
4.2 Get Revocation Status for a Certificate 28
4.3 View Associated CRL for a Certificate 29
4.4 Test Certificate on Custom Protocol 30
4.5 Certificate's Representations 32
4.5.1 PEM 32
4.5.2 ASN.1 32
4.6 Certificate's Public Key 33
4.7 Certificate Signing Request 33
4.7.1 Open Certificate Signing Request 34
4.7.2 Certificate Signing Request Details 34
4.8 Certificate Revocation Lists (CRL) 35
4.8.1 Open a Certificate Revocation List 36
4.8.2 CRL Details 37
5 CERTivity®'s KeyStore 43
5.1 KeyStores Capabilities 43
5.2 KeyStore Interface Organization 45
5.3 Create a New KeyStore 46
5.4 Open an Existing KeyStore 47
5.5 Open JREs CA KeyStores 48
5.6 KeyStore Persistence (Reloading opened KeyStores) 48
5.7 Open Microsoft Windows KeyStores 49
5.7.1 Open Windows Root KeyStore 50
5.7.2 Open Windows User KeyStore 50
5.8 Change KeyStore Password 50

5.9 View/Convert KeyStore Type 51
5.10 View Certificate Details 52
5.11 View Public Key Details 54
5.12 View Certificate Extensions Details 55
5.12.1 View Certificate Extensions ASN.1 Representation 56
5.13 View Certificate Chain Details 57
5.14 View Private Key Details 57
5.15 Generate Key Pair 58
5.15.1 Manage Certificate Extensions 60
5.16 Generate Secret Key 69
5.17 Import Trusted Certificate 71
5.17.1 Certificate Trust Established by User 72
5.18 Import Key Pair 73
5.19 SSL Certificates Retriever 74
5.20 Extend Validity 75
5.21 Regenerate Key Pair 76
5.22 Generate CSR File 77
5.23 Import CA Reply 77
5.24 Select CA Issuer 81
5.25 Sign Certificate by <aliasForIssuer> 81
5.26 Export Key Pair 82
5.27 Export Certificate Chain 82
5.28 Export Certificate 82
5.29 Export Public Key 83
5.30 Export Private Key 83
5.31 Rename a KeyStore Entry 85
5.32 Delete KeyStore Entry 85
5.33 Copy KeyStore Entry 85
5.34 Cut KeyStore Entry 86

5.35 Paste KeyStore Entry 86

6 CERTivity®'s Signatures 87

6.1 Verify 87

6.1.1 Verify JAR Signatures 87

6.1.2 Verify XML Signatures 88

6.1.3 Verify PDF Signatures 89

6.2 Sign 91

6.2.1 Signing JAR Files 91

6.2.2 Signing XML Files 92

6.2.3 Signing PDF Files 94

6.2.4 Signing CSR Files 95

7 FAQ 97

7.1 How to Install the Unlimited JCE Jurisdiction Policy? 97

7.2 Which Are the Available KeyStores Types in CERTivity Application? 97

7.3 Sometimes the Entry Name (Alias) Changes its Case 97

7.4 Fonts too large 97

7.5 Where is the Help Window on MAC OS? 98

7.6 Having rendering issues? 98

7.7 Why do I get an "Access Denied" error when trying to save a KeyStore to a file located in Program Files? 99

8 License Agreement 100

8.1 Definition 100

8.2 Grant of License 100

8.3 Restricted Use for Evaluation 100

8.4 Support Services 101

8.5 Refund 101

8.6 Restrictions 101

8.7 High Risk Activities 102

8.8 Third Party Rights 102

8.9 Laws and Regulations 102

8.10 Limited Warranty 103

8.11 Limitation of Liability 103

8.12 General 104

8.13 Contact Information 104

8.14 Changes to our License Agreement 104

9 Sales and Support 105

A CERTivity®'s Features Matrix 106

1. Overview

1.1 About

CERTivity® is a powerful pure Java multi-platform visual KeyStores manager. This standalone GUI desktop application provides a natural experience for managing and using KeyStores, Certificates, Key Pairs (Private Key, Certificate Chains), Secret Keys in various formats. It covers and combines functions that otherwise are available through verbose command line tools or other operating system tools or browsers. It is not intended to be just a simple 1:1 visual equivalent of these tools - the features being combined and centralized in an intuitive and productive organization.

Thus developers and system administrator can gain valuable time and ensure the greatest productivity by letting CERTivity taking care of the low level details in a uniform manner on almost all the systems - Windows, Unix/Linux, Mac.

In the long term, CERTivity aims at being a centralized manager and platform for the digital security related assets.

1.2 Features Summary

CERTivity has the following main features and advantages:

- **GUI Representation** of the security related items in a Tabbed Document Interface allowing for visualizing in parallel the following types of models: KeyStores, individual Certificates and Test Certificate Scenarios. The GUI representation is taking advantage of the natural approach of using an IDE style interface.

  KeyStores entries are represented using a Tree Table structure, each entry and sub-entry being visualized in a Details Panel resembling the view of an e-mail client. As well, KeyStore entries take advantage of contextual menus or natural editing actions such as but not limited to delete, rename, expand, undo/redo. Many of these actions can also be used through Keyboard shortcuts.

  Navigation between KeyStore entries is enhanced by positioning based on the first character (case sensitive) of a KeyStore alias or by sorting the table columns.

  Many of the application's components expose Context Sensitive Help (default `F1`), the Table of Contents tree being synchronized with the current context.

  Status Bar is displaying useful information including if the KeyStore is case sensitive, or case aware.

- **KeyStore management** - The application is able to work with a wide range of KeyStores types: (JKS, JCEKS, PKCS #12, BKS, UBER and Windows native ones) and supports the following KeyStore operations:

  1. Create (generate) a new KeyStore;
  2. Open an existent KeyStore;
  3. Opening the CA TrustStore(s) of the JRE(s) discovered on the current system;
  4. Save a KeyStore;
  5. Copy and Paste entries from one KeyStore to another;
  6. Copy to clipboard Certificates from Key Pair’s Certificate Chain;

7. Change a KeyStore's password;
8. Change a Key Pair's password;
9. Password manager to avoid entering Key passwords each time;
10. Emphasizing expired and about to expire Certificates or Key Pairs;
11. Emphasizing Certificates and Key Pairs for which the key size is smaller than a value set by the user;
12. Display trust status for Certificates in the KeyStore view;
13. Convert to other KeyStore format;
14. Delete KeyStore entry;
15. Change KeyStore entry alias;
16. Import Key Pairs from Key Pair files or from separate Private Key and one or more certificate files;
17. Import trusted Certificates;
18. Trust verification when importing certificates (with user confirmation when trust is not established);
19. Add Certificate Extensions;
20. Save Certificate Extensions as XML;
21. Generate self signed Key Pairs (Private Key with corresponding Certificate);
22. Generate new Key Pairs using the information from other already existing Key Pairs;
23. Set a custom minimum key size limit for new Key Pair generation;
24. Select the country code from a list of available countries resulting a valid ISO country code when generating a Key Pair;
25. Generate Secret Keys;
26. Retrieve certificates from servers (e-mail server, web server etc.) - This is based on the underlying SSL/TLS protocols;
27. Set SSL Connection Type (to be used when retrieving certificates);
28. View Private Key Details;
29. View Public Key Details;
30. View Certificate Chain Details;
31. Configurable KeyStore persistence on successive runs of the application.

- **Certificates operations**:
  1. Import Certificates / Certificate Chains into KeyStore either from files or from SSL connections;
  2. Open an existing Certificate as standalone (not part of a KeyStore);
  3. Display Certificate Details (having 11 Certificate Fingerprints types available);
  4. Display certificate trust status;

5. Display multiple certificates including certificate chains;
6. Obtain the revocation status from the signing CA through CRL;
7. View the CRL associated to a certificate;
8. Use/test a certificate against a SSL connection (including plain upgradable sockets) to an end-point and permitting raw TCP/IP level communication (similar to telnet/nc raw inspections); verbose handshaking information is also available;
9. View Public Key Details for the opened certificate;
10. View PEM representation;
11. View ASN.1 representation;
12. View Certificate Extensions;
13. View ASN.1 representation for a Certificate Extension;
14. Extend validity for a Key Pair entry.

- **Sign and verify**

  CERTivity aims to bridge the gap between keys management and digital signature functionality as well as offering a suitable introspection for developers interested in various investigations. CERTivity signs and verifies PDF, JAR and XML files with verbose details. CSR can be signed as well.

  1. Existent signature applications lack the in-depth key management and vice-versa, existent key management applications lacks the signing and verification process or the verbose details. CERTivity interconnects these functionalities;
  2. The embedded signature certificate can be directly imported into the active KeyStore;
  3. Signing is a contextual action while you browse the KeyStores so you will take advantage of all the existent key management features described above;
  4. PDF digital signature and verification:
     - Many existent PDF signature applications cover just the signature process, leaving the verification process to PDF readers or editors. CERTivity is offering a PDF signature verification process too, which can show you details that are not otherwise accessible, especially when you deal with the PDF specification;
     - Each signature details can be inspected;
     - All PDF standard SubFilter values are supported: adbe.pkcs7.sha1, adbe.pkcs7.detached and adbe.x509.rsa_sha1 as opposed to the general practice of supporting just adbe.pkcs7.detached;
     - The name of the PDF signature handler (Filter) is Adobe.PPKLite;
     - Multiple PDF signatures can be applied incrementally;
  5. XML digital signature and verification:
     - All XML signature types are supported: enveloped, enveloping and detached;
     - The XML signature is based on Java Specification Request JSR-105 which standardizes the XML Digital Signature APIs;

6. JAR digital signature and verification:
    - GUI alternative of the Java command line jarsigner tool, both for signing and verifying;
    - Sign and verify the Android Application Package (APK) signature.

- **Export options**:
    1. Retrieve and Export Certificates from multiple sources into multiple formats;
    2. Export Key Pairs, Certificate Chains, Private Keys, Public Keys;
    3. Some of the formats supported besides the KeyStores themselves are:
        - X.509 Certificate Files;
        - X.509 Certificate Files (PEM encrypted);
        - PKCS #7 Certificate Files;
        - PKCS #7 Certificate Files (PEM encrypted);
        - PKI Path Certificate Files;
        - PKCS #12 Key Pairs;
        - PKCS #8 Key Pairs;
        - OpenSSL Public Key;
        - OpenSSL Public Key (PEM encrypted);
        - PKCS #8 Private Key Files;
        - PKCS #8 Private Key Files (PEM encoded);
        - OpenSSL Private Key Files (PEM encoded);
        - PKCS #10 for CSR;
        - SPKAC for CSR;
        - ASN and PEM for visualizing most of the items.
- **TrustStores Management**:
    1. Set/Remove TrustStores at runtime without restarting the application;
    2. Configure Trust Path validation options at runtime without restarting the application;
- **Certificate Authority functions**:
    1. Certificate Signing made easier using "Select as CA Issuer" and "Sign Certificate by <aliasForIssuer>" actions;
    2. Generate Certificate Signing Request (CSR) files;
    3. Sign Certificate Signing Request (CSR) files;
    4. Import CA Reply;
    5. Trust verification when Importing CA Reply (with user confirmation when trust is not established);
    6. Check PKI file type;

7. Open Certificate Revocation Lists (CRL) from files or URLs;
8. Open Certificate Signing Request (CSR) files;
9. Certificate chain management: append and remove signer certificate (with Copy/Paste/Delete/Undo/Redo functionality included);
10. By generating CSR files, signing CSR and importing CA reply the application can act as a testing purposes CA.

- Multi-platform

  Being a Java application it runs anywhere an Oracle (Sun) JRE or Apple JRE can run. Depending on the platforms, CERTivity comes bundled with JRE or standalone. Please check the download page for the suitable setup package.

Most of the operations are executed on separate threads, so that for example while generating a key or signing a PDF one can perform other tasks as well.

The existence of some of these features are controlled by the category of your license - either Standard or Professional. For the features matrix of CERTivity see the Appendix A, *CERTivity® 's Features Matrix*.

1.3 Documentation and Samples

CERTivity provides documentation in more format types:

1. documentation file, `CERTivity.pdf`, contained in the `doc` subdirectory from the distribution kit and also available on our website at http://www.edulib.com/products/keystores-manager/download/#manual;
2. application help which can easily be accessed using the application's Menu Help;
3. many of the application's components expose Context Sensitive Help (default `F1`), the Table of Contents tree being synchronized with the current context;
4. the HTML CERTivity manual is available on our website for reading on line (using http://www.edulib.com/keystores-manager/resources/doc/html/CERTivity/CERTivity.html).

Samples files are contained in the `samples` subdirectory of the `doc` directory from the distribution kit. You have further details (including passwords) in the `Readme.txt` file which will be opened when you first run the application. The `Readme.txt` file can be opened again using **`Menu File > Open > Open Sample Readme.txt`**.

The `samples` subfolder is further divided in 10 subfolders:

1. `keystore` - can be used to observe the KeyStore features;
2. `certificates` - can be used to observe the certificates features;
3. `crl` - can be used to observe the open CRL feature;
4. `csr` - can be used to observe the CSR features - generate and sign;
5. `jar` - can be used to observe the sign/verify JAR files features;
6. `pdf` - can be used to observe the sign/verify PDF files features;
7. `xml` - can be used to observe the sign/verify XML files features;
8. `keypair` - contains examples of exported Key Pairs useful to observe the export/import features for key pairs;

9. `privatekey` - contains examples of a Private Key exported in myriad formats. Together with the suitable certificate from the `certificate` directory a Key Pair can be imported in a KeyStore;
10. `publickey` - contains examples of a public key exported in various formats.

2. CERTivity®'s Administrative Details

2.1 System requirements

CERTivity being a GUI desktop application it requires a Graphical Interface to install and run.

The minimum and recommended hardware configurations are depicted in the table below:

Table 2.1. System Requirements

Minimum Configuration	Recommended Configuration
Processor: 800MHz Intel Pentium III or equivalent	**Processor**: 2.6 GHz Intel Pentium IV or equivalent
Memory: 512 MB	**Memory**: 2 GB
Disk space: 200 MB of free disk space	**Disk space**: 400 MB of free disk space

2.2 Platforms and Java Virtual Machines

CERTivity is a Java based application, available for download and install through:

- Bundled JRE Java Native Installers, the recommended option;
- Normal Java Native Installers (standalone, no JRE bundled);
- Plain archives (Zip/TAR.GZ) - manual process.

The recommended CERTivity distribution for Microsoft Windows and Linux platforms (x86/AMD64) is the Installer with the embedded Java Runtime Environment (JRE) as everything is out of the box. If the target operating system is not in the list of the supported embedded JRE distribution, or there are other reasons for which you don't want an embedded JRE, you will need to have an already installed JRE from Sun Microsystems (Oracle) or from Apple (if the target system is a Mac OS). Note that the embedded JRE is exclusively used for running the CERTivity application and is not affecting the existent applications.

Being a Java based application, CERTivity could run anywhere where an Oracle or Apple JRE is available - this covers a wide range of systems such as Microsoft Windows, Linux, Unix and Mac. The exact JRE for the CERTivity application is configured in the `jdkhome` property from the configuration file `${certivity_home}/etc/certivity.conf` (on Windows), respectively `${certivity_home}/etc/certivity.conf.sh` (on Linux, Unix and Mac). The Installer is transparently setting this path according to the installer type:

- If you are using the recommended bundled JRE Installer, then the embedded JRE is used to run the installer itself and is also configured in the `jdkhome` property.
- The standalone (non bundled JRE) Installer will need a JRE to start with and to set for CERTivity. If you are using the standalone Installer make sure your system has a Sun Microsystems (Oracle) / Apple JRE available for the user you intend to use to install CERTivity. If not, then please install JRE 1.6/1.7 according to the Java vendors instructions. We do not recommend using JRE 1.8.

  The standalone Installer will automatically look for a suitable JRE on your platform for the installation itself and for configuring CERTivity to start with. The minimum required version is 1.6. The first JRE found in this search order is mainly used:

  - Environment variables `${JAVA_HOME}`, `${JDK_HOME}`;
  - Windows registry (if the OS is Windows) ;

- Standard locations.

  If the JVM found is not the standard recommended one a warning message is displayed but the installation will continue.

- If you are using a plain archive distribution, then you will need to install JRE 1.6 or JRE 1.7 according to Sun Microsystems (Oracle) / Apple instructions and to configure the `jdkhome` property from `${certivity_home}/etc/certivity.conf(.sh)` by uncommenting it and pointing it to the JRE path, for example `jdkhome="C:\Program Files\Java\jre6"`. If you are not configuring `jdkhome`, CERTivity will try to use the default location of JDK/JRE on the platform, but in case the target system has more Java distributions installed, it is safer to expressly point the `jdkhome` property to the exact location.

The exact Java version the CERTivity application started with is displayed in the `Help > About` menu, for example:

```
Java: 1.6.0_25; Java HotSpot(TM) Server VM 20.0-b11
```

As there are many flavours of hardware, operating systems and versions available it is practically impossible to test each one of these. We have successfully run CERTivity on Micrososft Windows XP, Microsoft Windows 7, Microsoft Windows Server 2003, Microsoft Windows Server 2008, Linux Debian, Linux Ubuntu, RedHat, Solaris Intel and Mac OS X. Both 32 and 64 bit OS distributions as well as JREs are supported.

Even if there is a wide platform distribution there are just few known issues:

- Default font size being ignored by the GTK Look and Feel on Gnome Desktop Environment on Linux;
- Contextual Help Window is not brought up to front on Mac OS X;
- On Windows 64 bit with JRE 1.6 64 bit distribution the Windows KeyStores are not functional, though they work with JRE 1.6 32 bits or JRE 1.7 64 bit distribution.
- In Windows 7, like for many other applications, an "Access Denied" error may be encountered when trying to save KeyStores or when exporting Certificates, Key Pairs, or Private / Public Keys to files that are located in the Windows special folders (e.g. Program Files, Program Files (x86)), due to the User Account Control (UAC) function. This can be fixed by setting the UAC to a lower level or by turning it off.

2.3 Install and Run

2.3.1 Installing CERTivity

On Linux/Unix platforms after downloading the Installer you must make it executable, by opening a shell, going into the downloads directory (via `cd`) and running the command `chmod +x`, for example:

```
chmod +x CERTivity-unix-1.0.sh
```

Then you start the installer either from a file manager or from the command line by running the above executable script in the same directory, for example:

```
./CERTivity-unix-1.0.sh
```

The Installer will guide you through the necessary steps for having CERTivity installed. The default installation directory and other links are containing the version of the product in

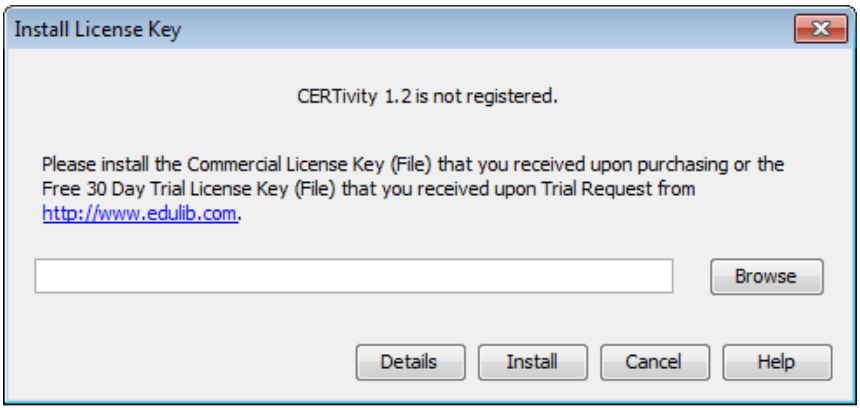
them. We strongly recommend to keep this naming convention. Default Installation directory examples are given below:

- On Windows: `C:\Program Files\EduLib\CERTivity 1.0`;
- On Linux: `/opt/EduLib/CERTivity 1.0` or `/home/user/EduLib/CERTivity 1.0` depending on the user's rights for the `/opt` directory;
- On Mac: `/Applications/EduLib/CERTivity 1.0/`.

As stated in the previous section, if the Java version that we recommend is not detected, the Installer will present a warning dialog, continuing the installation. After the installation finishes, we advise you to install the standard JRE (recommended 1.7) and configure CERTivity to use that standard JRE according to the above section.

2.3.2 License Key (File) Registration

After the installation, when you first run CERTivity, you will be asked to activate it either for your trial or for your purchased license. Make sure you correctly point to your license file, then press Install.



A Commercial License for a CERTivity version is valid for all the minor versions. For example, if you purchased CERTivity 1.0, installed and registered it you will also be able to install and run CERTivity 1.3 on the same machine, without the need of requesting or registering another license file. So the above dialog will not appear in such a case if you use the same machine.

If you have a trial license you can register a commercial license without the need to uninstall the existent version. Use the menu `Help > License > Install License Key` to change the license to the one that you purchased. In the same way, without uninstalling, you can switch from a Standard to a Professional License activating all the features.

2.3.3 Running CERTivity

CERTivity can be started either from the Start Menu / Desktop shortcuts or directly using the applications startup launchers from the installation base directory `bin/certivity.exe` on Windows system, `bin/certivity` on the rest of the systems. This later option is the only one available if you installed CERTivity using a plain archive.

The CERTivity application settings are stored outside the application directory under the `EduLib/.certivity/<version>` sub-directory, located under the platform dependent user home directory. The settings are accessible through `Tools>Options` panel as described in Section 3.6, “Tools Menu”. From this panel you can export the settings values

from the old version into the new application version via an intermediary archive file. That is the recommended way.

The CERTivity application is also producing log files in the same user location. Its exact sub-path under the platform dependent user home directory is `(.)EduLib/.certivity/<version>/var/log/` . The last 3 application runs are logged in `messages.log`, `messages.log.1` and `messages.log.2`. The exact path is dependent on the target OS, for example:

- On Windows XP `C:\Documents and Settings\John\Application Data\.EduLib\.certivity\1.0\var\log`;
- On Linux `/home/john/.EduLib/.certivity/1.0/var/log`;
- On Mac `/Users/john/Library/Application Support/EduLib/certivity/1.0/var/log`

The Log level can be configured from the Application Options accessible through `Tools>Options` panel as described in Section 3.6, “Tools Menu”, via the option `Log level` which comes with the default value of `INFO`.

2.3.4 Handling Multiple CERTivity versions

Multiple CERTivity versions can be run on the same machine, each one being independent of the other. This is the normal process for upgrades. Each application version has its own installation directory, its own Start Menu links, its own settings.

The CERTivity application settings are accessible through `Tools>Options` panel as described in Section 3.6, “Tools Menu”. From this panel you can export the settings values from the old version into the new application version via an intermediary archive file. We recommend that after installing the new version and becoming familiar with it, you uninstall the previous version.

It is possible as a repairing process to install the same CERTivity version overwriting the previous installation.

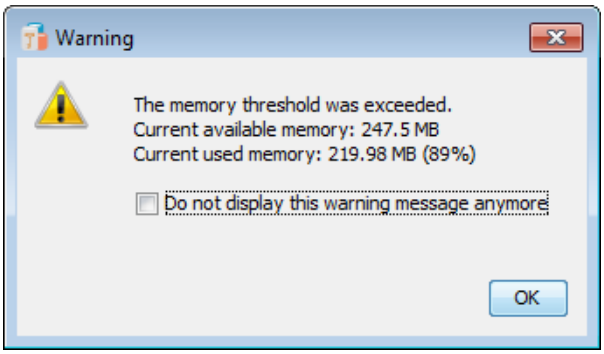
2.4 Java Virtual Machine settings

Standard Java Virtual Machine parameters can be passed through the configuration file `${certivity_home}/etc/certivity.conf(.sh)` in the `default_options` property by pre-pending `-J` to the standard parameter, for example `-J-Xmx256m`.

If you handle larger files (especially the PDF operations are memory consuming) or more files at the same time you may ran out of JVM Heap memory. By default CERTivity comes with a maximum of 256 MB configured as above. Depending on the memory available on the target machine, the operating system 32/64-bit architecture, you can specify different sizes for memory, for example by modifying `-J-Xmx256m` into `-J-Xmx512m`. After doing this setting you need to restart the application. If the heap memory amount that you specified is too high for that platform/32-bit model, upon starting up a "`JVM creation failed`" message will be present. You will need to adjust the memory settings according to your architecture - for example on Windows on 32-bit JVM distributions due to the 32-bit Windows process model you cannot specify values larger than about 1100 MB. It is also not recommended to set the maximum JVM heap to be larger than the physical memory.

Note that this file is specific to each CERTivity version, so this allows you to manage each version independently.

CERTivity has a Memory Detection system letting you know when the maximum heap memory passes a threshold. This is configured in the Application Options (accessible through `Tools>Options` panel as described in Section 3.6, “Tools Menu”) by the property Memory usage warning max threshold, which comes with the default value of 90%. In such case the following dialog will be presented:



It is possible that after this warning the JVM may issue an OutOfmemory error if the threshold is too tight. In such a case please adjust the JVM heap setting as described above.

The CERTivity application also features a memory toolbar, which by default is on - there you could see the exact memory used and even force a garbage collection.

2.5 Purchase and Licensing Model

CERTivity is a commercial product, which is also offered for Trial for 30 days. The trial registration is fully functional, except limitations on the number of KeyStores that can be opened and created during an application instance (run). In order to be able to use a certain version of CERTivity permanently and without any limitation a commercial license is needed.

Facts about the licensing model:

- There are two categories of commercial licenses - Standard and Professional - the feature differences are depicted in Appendix A, *CERTivity®'s Features Matrix*. The price difference is depicted on our web site [http://www.edulib.com/products/keystores-manager/purchase/];
- The license is per number of users and each user can use the software on more machines;
- We offer volume discounts (per number of users) according to the information on our web site;
- If the License Key (File) is for more users, as the file is the same, each user will have to use that file to register the application;
- A License Key (File) covers a major version and all its minor upgrades (e.g. 1.0, 1.1, 1.2..., but not 2.0). The same License Key (File) will unlock all the minor versions. If you purchased a license for a certain major version (e.g. 1.2) you don't have to purchase licenses for any of the next minor versions (e.g 1.4) if the software is used by the same users;
- If in less than 60 days after you purchase a License Key (File), a major release is out, you are entitled to use the new major version for the category of license you bought. You can download and use it as well as any further minor versions with the same License Key (File);

- For new major versions (e.g. 3.0) existent users of the previous major version (e.g. 2.0) will be able to upgrade their License Key (File) by purchasing that license at an important discount.
- Changing your license category ("migrade") of a certain License Key (File) is possible by paying the difference between the categories; this is available upon request by providing us the previous details of the license;
- We offer a full 30 days refund.

2.5.1 Payment details

Our orders are securely handled by our e-commerce and payment partner Avangate [http://www.avangate.com]. Avangate handles all the payment details in a safe and secure manner, starting from using SSL channels for communication up to anti-fraud screening procedures. Upon payment you will be redirected from our site to the Avangate site for the payment. You will benefit from multiple payment methods, over 49 currencies and a localized user-friendly store.

If you want to purchase a license for more users you will get a discount according to the one specified on our web site [http://www.edulib.com/products/keystores-manager/purchase/]. The price is automatically adjusted in your shopping cart depending on the quantity you select.

2.5.2 What do I get after payment

Our software delivery is electronic only. Even before the payment you can download the suitable setup package of the last CERTivity version. Shortly after the payment you will get via the e-mail the License Key (File) to activate your CERTivity copy - please store it in a secure place, as you may need it for future installations. For instant payment methods like Credit Card or PayPal, the delivery e-mail will be sent usually within a few minutes after an order has been successfully completed. For payment methods such as Bank Transfer, check or PayPal eCheck, the delivery will be done after the payment is confirmed (usually 2 to 3 business days).

For more details about the payment and the e-mail delivery you can read the Avangate - Shoppers FAQ [http://www.avangate.com/help/customer-faq.php].

You will use the License Key (File) that you will have been receiving via email to register your CERTivity application. You may be in one of the following scenarios:

- On a certain machine, you have already installed CERTivity and run it registered with an unexpired trial license. Then start the application and use the menu `Help > License > Install License Key` to change the License Key (File) to the one that you just purchased; the same scenario applies if you migrate from the Standard category to the Professional category.
- On a certain machine, you have already installed CERTivity but the trial license has expired. Then upon application start-up you will be prompted with an Install License Key dialog. There you will provide the License Key (File) that you just purchased.
- You don't have any installation of CERTivity on the target machine or you have other previous major versions of CERTivity. Then you need to download and install the new major version of CERTivity. The installation is done in parallel, not overwriting the previous version. After the installation, upon the application start-up you will be prompted with an Install License Key dialog. There you will provide the License Key (File) that you just purchased.

If, at a later time, a new minor version is released you will just download and install it in parallel and it will be registered with the same License Key (File). If you need to install the new minor version on a different machine (used by the same user) you will need to provide the License Key (File) that you initially received.

3. CERTivity®'s Menus/Tool bar

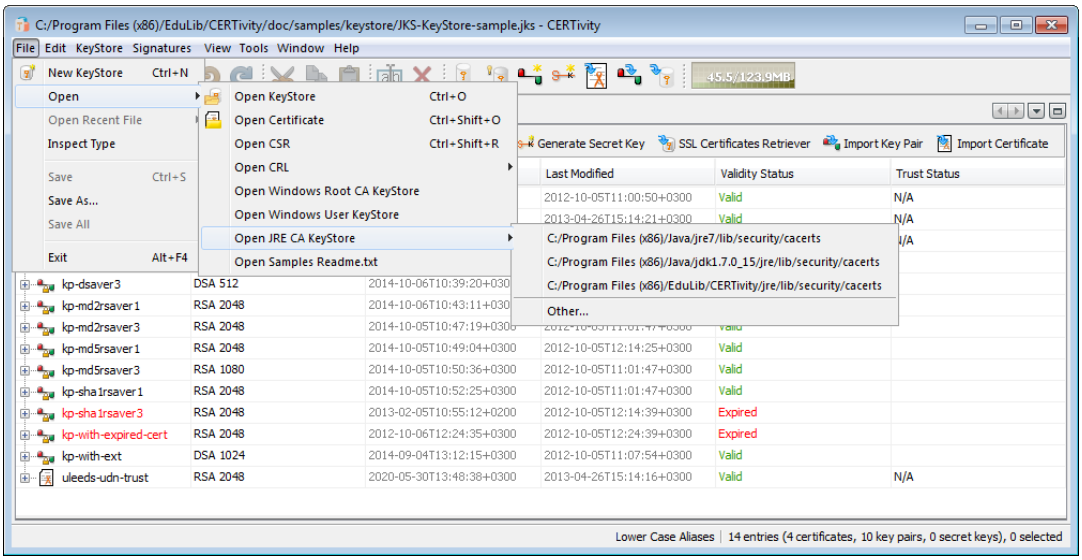
3.1 File Menu

The File menu includes the following commands:

- **New KeyStore** - Creates a new KeyStore;
- **Open** - Opens an existing KeyStore, an existing certificate, an existing CSR file, an existing CRL file, Windows Root CA KeyStore, Windows User KeyStore, JREs CA Keystores and the Samples Readme.txt file;
- **Open Recent File**- Opens a file from a list of the most recently used files;
- **Inspect Type** - when accessed, detects the type of a file. The cryptographic file types detected are:
  - JKS KeyStore;
  - JCEKS KeyStore;
  - BKS KeyStore;
  - UBER KeyStore;
  - PKCS#12 KeyStore or Key Pair;
  - Certificate;
  - Certificate Signing Request (CSR) of PKCS10 type;
  - Certificate Signing Request (CSR) of SPKAC type;
  - Certificate Revocation List (CRL);
  - Encrypted Microsoft PVK Private Key;
  - Unencrypted Microsoft PVK Private Key;
  - Unencrypted OpenSSL Private Key;
  - Unencrypted PKCS #8 Private Key;
  - Encrypted OpenSSL Private Key;
  - Encrypted PKCS #8 Private Key;
  - OpenSSL Public Key.
- **Save**- Saves the current KeyStore, if it has been modified;
- **Save As** - Saves the opened KeyStore under a different name;
- **Save All** - Saves all the opened KeyStores;
- **Exit** - Exits the application.

Note that the features regarding the opening of the native Windows related are available only if run on a Microsoft Windows platform.

A screenshot for the File Menu is given below:



3.2 Edit Menu

The Edit menu includes the following commands:

- **`Cut`** - Removes the currently selected entry from a KeyStore and places it in the clipboard;
- **`Copy`** - Copies the selected entry into the clipboard;
- **`Paste`** - Inserts the entry from the clipboard in the KeyStore;
- **`Rename`** - Renames the KeyStore entry;
- **`Delete`** - Deletes the KeyStore entry.

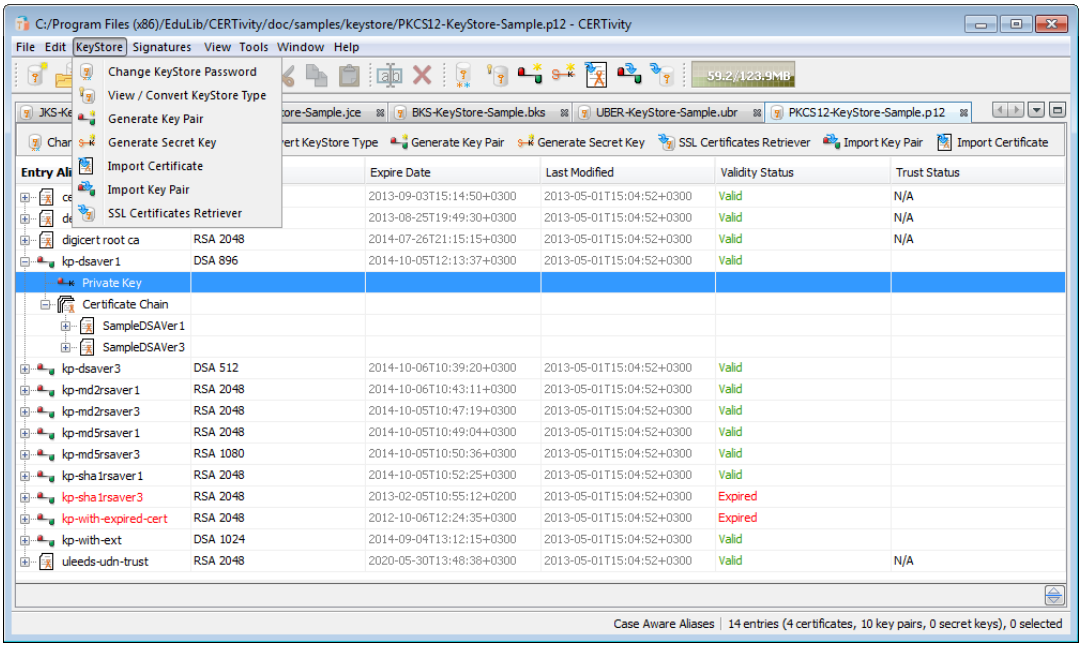
These commands are also having their usual Keyboard Shortcuts by default.

3.3 KeyStore Menu

The KeyStore menu is available when a KeyStore is opened and includes the following commands:

- **`Change KeyStore Password`** - Changes the password of the store;
- **`View/Convert KeyStore Type`** - Views the current store type and gives the possibility to change it ;
- **`Generate Key Pair`** - Generates a new Key Pair;
- **`Generate Secret Key`** - Generates a new Secret Key - note that not all of the KeyStore types are able to store Secret Keys;
- **`Import Certificate`** - Import a Certificate from a file into the KeyStore;
- **`Import Key Pair`** - Import a Key Pair from a file into the KeyStore;
- **`SSL Certificate Retriever`** - Connects to a remote SSL socket (host and port) and extracts the certificates used during handshaking. It is then possible to inspect and import the certificates into the KeyStore.

A screenshot for the KeyStore menu can be seen below:



3.4 Signatures Menu

The Signatures menu includes the following commands:

- **Verify** - validates the following types of files presenting verbose details; it is possible to view and export the certificates embedded in the verified files.
    1. **JAR file** - Validates a signed JAR file;
    2. **PDF file** - Validates a signed PDF file;
    3. **XML file** - Validates a signed XML file.
- **Sign** the following types of files using the current Key Pair entry; disabled if no Key Pair is selected. Also accessible from the contextual menu for a Key Pair. Specific sign parameters are presented depending on the file type.
    1. **JAR file** - Signs a JAR file using the current Key-Pair entry;
    2. **PDF file** - Signs a PDF file using the current Key-Pair entry;
    3. **XML file** - Signs an XML file using the current Key-Pair entry;
    4. **CSR file** - Signs a CSR file using the current Key-Pair entry.

3.5 View Menu

The View menu includes the following commands:

- **Toolbars** - Is used to show or hide toolbar groups;
- **Full screen** - Allows application to run using the whole screen. The main toolbar will be hidden during in this mode.

3.6 Tools Menu

Using the Tools menu, you can change:

- The application options (use `Main Options`);

- Trust Path Options.

3.6.1 Main Options

The Main Options of the application are the following:

- Certificate expiry notification period (default 30 days), meaning that if a certificates valid interval ends before the current date + the notification period a certain visual element will alert you;
- RSA Key Pair default size (default 1024),- the default size for RSA keys which will be used when generating a RSA Key Pair. Change it for your convenience;
- RSA Key Pair max size (default 4096) - you won't be able to generate a key Pair having more bits than this value. This prevents bigger values that would require a great CPU time to generate;
- Auto generated Certificate serial number max bit length (default 20);
- Undo level - the number of undo levels for each opened KeyStore (default 20);
- Log level;
- Memory usage warning max threshold, meaning the percentage of used memory after which a warning message will be displayed (default 90);
- KeyStore persistence - the type of persistence for opened KeyStores when exiting the application. CERTivity® can remember the KeyStores which are opened when the application exits, and reload them again when the application is launched next time. There are two options available:
  - Persist only KeyStore file name - meaning that only the name (and path) of the previously opened KeyStores will be remembered to be reopened on the next launch. The passwords of the KeyStores will not be remembered, and you will be prompted to enter the password for each of them when selecting each KeyStore tab first time (recommended);
  - Fully persist - meaning that the name and password of the KeyStores will be remembered so that the KeyStore to be reopened when launching the application, without prompting you for the passwords of the KeyStores. The passwords are stored in an encrypted way.

  Although the "Fully persist" option makes the application more friendly, use this option with care and only when you are sure the machine is exclusively accessible by you;
- Recent File list max size - sets the list maximum size for the most recently used files (default 10);
- JRE CA KeyStore list max size - sets the list maximum size for JRE CA KeyStore list (default 10);
- Certificates Retriever connection type - sets the connection type used when retrieving certificates, the combo-box being populated with all connection types available for the Java version used;
- Inspected and draggable file size limit (default 2048 KB) - sets the size limit for the files inspected using the "Inspect File" action and for the drag and drop action.

3.6.2 Trust Path Options

The user has the possibility to set the TrustStores which should be used for establishing trust when importing a certificate from different sources, when importing a CA Reply, or when

displaying the trust status for certificate entries in the KeyStore view. Also, the user has the possibility to set a series of Trust Path validation options.

The Trust Path Options have 2 main categories:

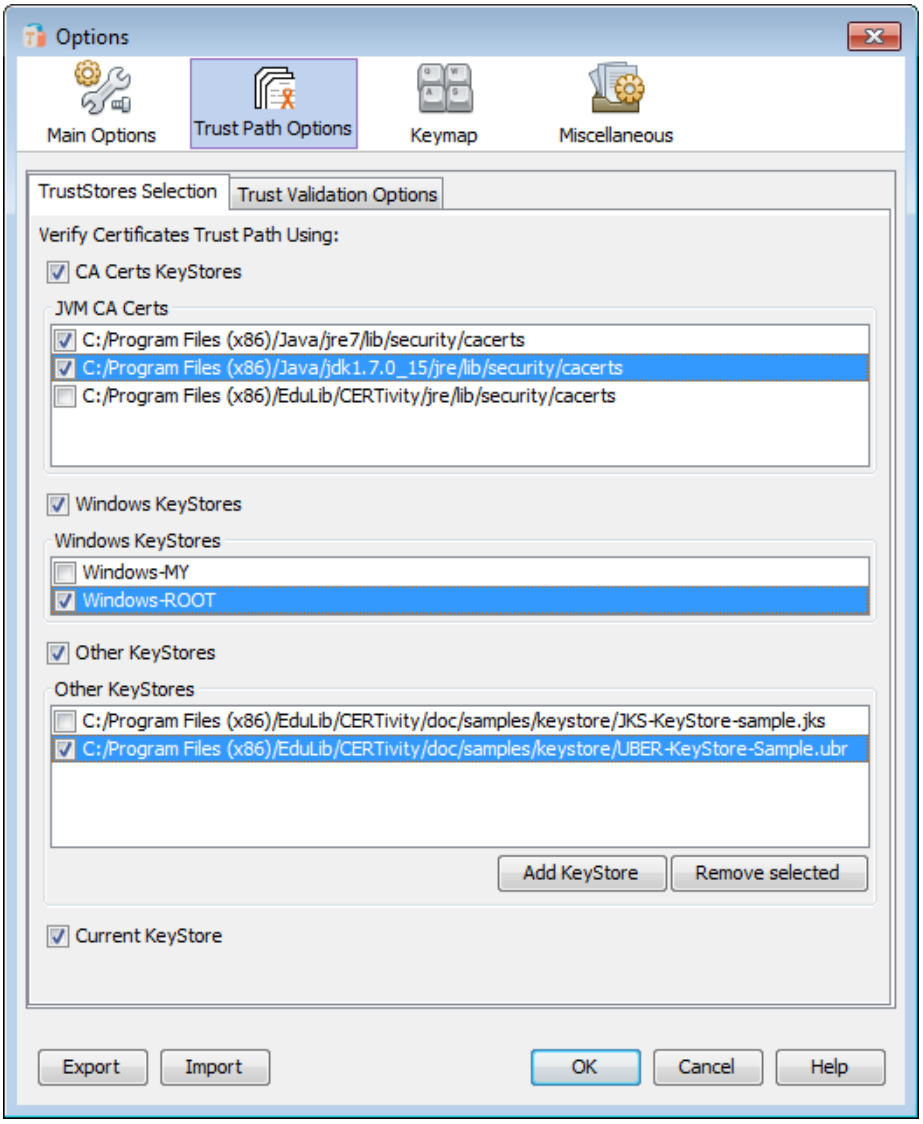
- TrustStores Selection;
- Trust Validation Options.

3.6.2.1 TrustStores Selection

A TrustStore is basically a KeyStore which contains Trusted Certificate Entries.

CERTivity allows setting more TrustStores which can be chosen from the JRE CA TrustStores discovered on the current machine, from the Windows Native KeyStores (if running on a Microsoft Windows system), or from a custom KeyStore which you can select that can act like a TrustStore. Also, you have the option to set as a TrustStore the current active KeyStore. The current active KeyStore is the KeyStore which is opened and focused at the moment of starting an operation (such as importing a certificate, importing a CA Reply, etc.).

A screenshot of the TrustStores Selection panel which allows selecting one or more TrustStores (as it can be seen), is depicted below:



As it can be seen, there are 4 categories of TrustStores: CA Certs KeyStores, Windows KeyStores, Other KeyStores and Current KeyStore. Each of these categories can be disabled or enabled by clicking on the corresponding checkbox. The selections made within each category will not be lost when unselecting the category from its checkbox.

In the situation in which some CA Certs KeyStores are not found anymore, they will not be displayed when opening the Options panel.

To add a custom KeyStore, select `Add KeyStore` button. A file chooser dialog will be opened and you will be able to select a KeyStore. Any type of KeyStore from the ones supported by CERTivity can be selected here. To remove a KeyStore, select the KeyStore from the list and press `Remove selected`.

For the new TrustStores that you add or select, you will be prompted to enter the passwords of the KeyStores only if the KeyStores have not been opened in the current run of CERTivity, or if they are not currently opened. Also, you will be prompted to enter the passwords only when they will be needed first time. For example, when closing the Options dialog by pressing

OK, if there is no KeyStore opened in background, you will not be prompted to enter the passwords of the new TrustStores that you selected right away. You will be prompted to enter them when you will open a KeyStore, or a Certificate from file, or when performing any other action which will need the TrustStores for trust validation.

Also, if the password of a TrustStore is changed from outside CERTivity, you will be prompted again to enter the password when that TrustStore will be reloaded from the file.

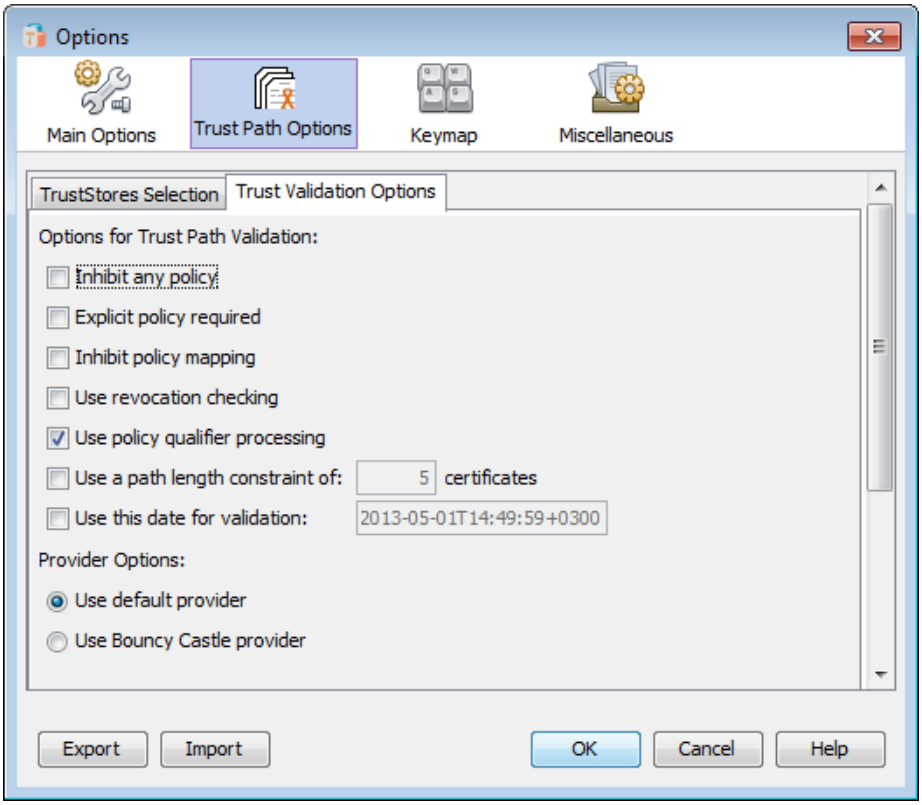
Note

When prompted to enter a password, if you select "Cancel" or you close the dialog, the TrustStore will be unselected, and it will not be used again until you select it again (going to `Options > Trust Path Options > TrustStores Selection`).

3.6.2.2 Trust Validation Options

When establishing the Trust Path for trust validation, there are more parameters which can be taken in consideration. Some of them are configurable, and the user has the possibility to set them according to his needs by going to the `Trust Validation Options` tab.

The Trust Validation Options panel looks like in the screenshot below:



As it can be seen, the user can set the following options:

- `Inhibit any policy`

  Default value: unselected;

  If selected, any policy OID will be inhibited if it's included in a certificate;

- `Explicit policy required`

  Default value: unselected;

  If selected, an acceptable policy needs to be explicitly identified in every certificate;
- `Inhibit policy mapping`

  Default value: unselected;

  If selected, policy mapping will be inhibited;
- `Use revocation checking`

  Default value: unselected (using the Default provider), unavailable (using the Bouncy Castle provider);

  If selected, the default revocation checking mechanism of the underlying service provider will be used (if the Default provider is selected and if the Default provider supports revocation checking). The Bouncy Castle provider does not support revocation checking, so this option is disabled for the Bouncy Castle provider. Also, in the situation in which the Default provider option is selected and the only available provider is Bouncy Castle, revocation checking will not work;
- `Use policy qualifier processing`

  Default value: selected;

  If selected, the most common (and simplest) strategy for processing policy qualifiers will be used.
- `Use a path length constraint of n certificates`

  Default value: unselected;

  If selected, it sets the number of non-self-issued (non self-signed) intermediate certificates that may exist in a certification path. The last certificate in a certification path is not an intermediate certificate and it is not included in this limit.

  A negative value set implies that the path length is unconstrained. This is equivalent with unselecting the "Use a path length constraint of ..." check-box.

  A value of 0 certificates implies that the path can only contain a single certificate.

  The default maximum path length is 5.

  ### Note

  If the check-box is unselected, the value from the text field will be ignored and the path length will be unconstrained.
- `Use this date for validation`

  Default value: unselected;

  If selected, it sets the time for which the validation of the certification path should be evaluated. If not selected, the current date and time (at the moment of performing the validation) will be used.

When setting a date, the entered date format must be ISO 8601.

The default value of the date field is the current date and time in ISO 8601 format.

Also, the user has the possibility to set the provider that will be used for Trust Path validation. The user can choose either the default provider (which is the first provider from the system where CERTivity runs which supports the Trust Path validation operations), either the Bouncy Castle provider. The Bouncy Castle provider supports almost all the Trust Path validation operations with some limitations. It does not support revocation checking ("Use revocation checking" option will be disabled). Also, if the default provider is selected but the only available provider is Bouncy Castle, revocation checking will not work.

3.6.3 Other Options

- Defined KeyBoard shortcuts (use **`Keymap`**);
- Appearance options (use **`Miscellaneous`**).

3.7 Window Menu

Using Window menu, you can run the following IDE related actions:

- **`Close window`**;
- **`Maximize window`**;
- **`Unlock window`**;
- **`Close all documents`**;
- **`Switch between opened documents`**;
- **`Reset window`**.

3.8 Help Menu

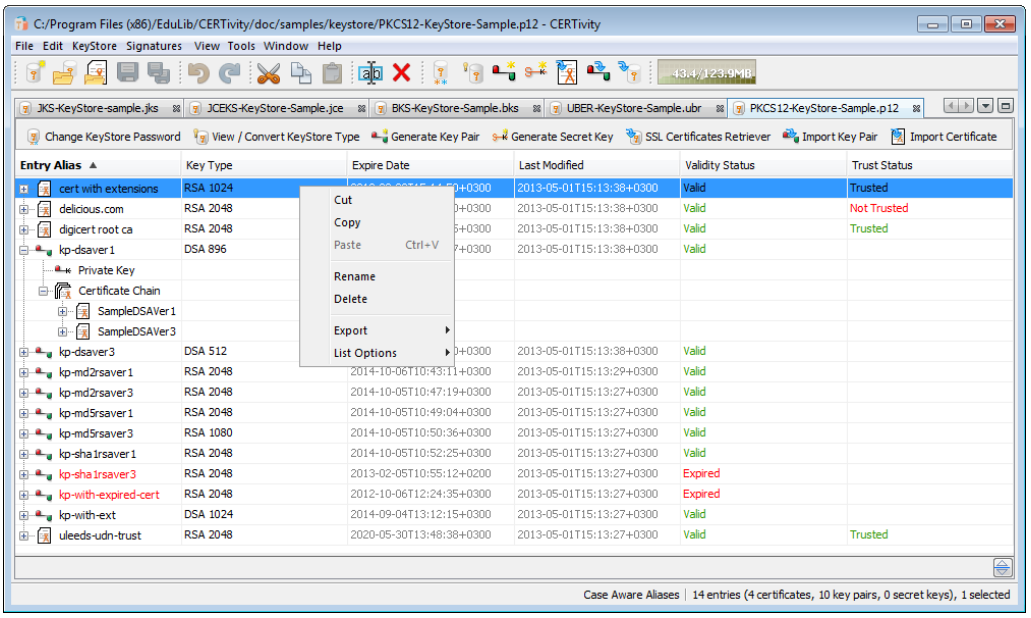
The Help menu includes the following commands:

- **`License`** - Allows installing a new license and viewing the installed license details;
- **`Help Contents`** - Access to CERTivity Help;
- **`About`** - Provides minimal information about the application and system.

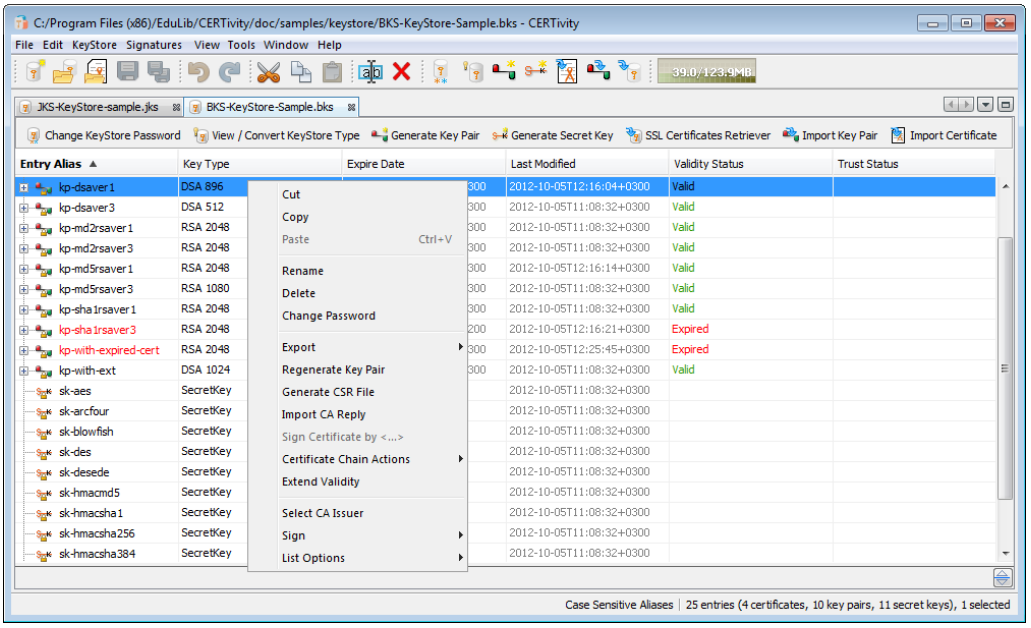
3.9 Contextual Menu

Each KeyStore entry or sub-component has a contextual menu associated with it. Items such as rename and delete are commons to most entries, but for a clear picture all the available actions are specified below depending on the entry.

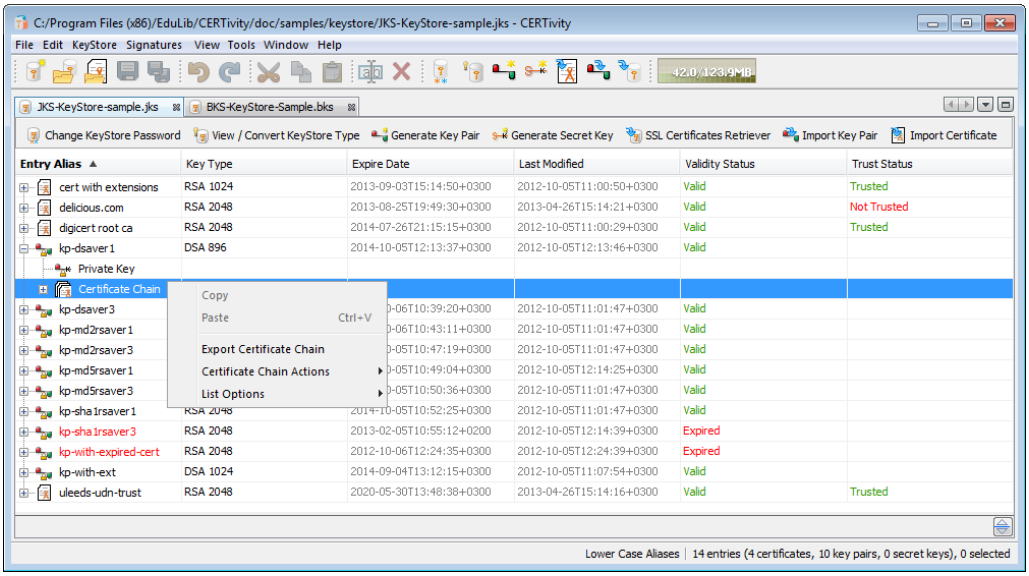
- **Certificate entry**: Cut, Copy, Paste, Rename, Delete, Export Certificate, Export Public Key;



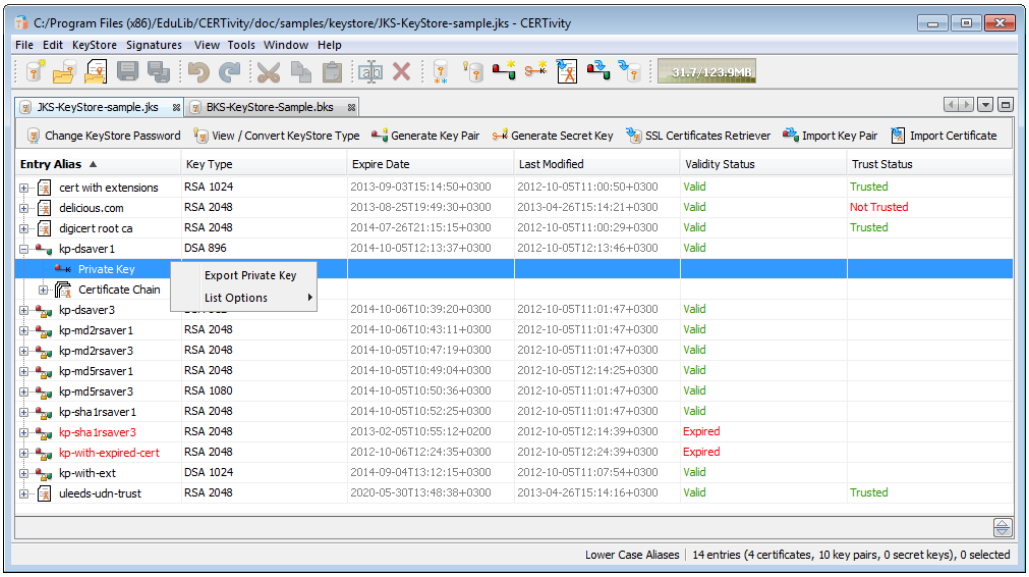
- **Key Pair entry**: Cut, Copy, Paste, Rename, Delete, Change Password, Export Key Pair, Export Private Key, Export Certificate Chain, Generate CSR File, Import CA Reply, Sign Certificate by <aliasForIssuer>, Extend Validity, Select CA Issuer, Sign CSR file, Sign XML file, Sign PDF file, Sign JAR file;



- **Certificate Chain entry**:Export Certificate Chain, Certificate Chain Actions;



- **Private Key entry**:Export Private Key.



3.10 Toolbar

The Tools toolbar provides a quick method for invoking tools for using in the application. The toolbar items are grouped in certain categories that can be disabled or enabled using the `View > Toolbars` menu option:

Icon	Action	Category
	Creates a new KeyStore.	File
	Opens an existing KeyStore.	
	Opens an existing Certificate as Standalone.	

	Saves the active document.	
	Saves all documents.	
	Undo the last action, if possible.	Undo/Redo
	Redo allows commands that have previously been undone with the Undo to be redone.	
	Removes the currently selected data from the active document and put it on the clipboard.	Clipboard
	Copies selected data onto the clipboard.	
	Inserts a copy of the clipboard contents at the insertion point.	
	Changes the alias name for the KeyStore entry.	Edit
	Deletes the KeyStore entry.	
	Changes the KeyStore password.	KeyStore
	Converts the KeyStore type.	
	Generates a Key Pair.	
	Generates a Secret Key.	
	Retrieves Certificates from SSL.	
	Imports a Key Pair.	
	Imports a trusted Certificate.	

There is also a memory usage toolbar icon showing the current memory and the current maximum memory of the Java Heap. When clicking on it, you can force the garbage collection.

Besides the main toolbar, the application provides a secondary toolbar on the KeyStore tab, for quick access.

4. CERTivity®'s Certificates

4.1 Open Certificate

A Certificate embeds a public key belonging to an entity. It certifies the public key and all the information via digitally signature of another entity (the issuer, e.g. - a person, company, etc.), saying that the embedded public key (and some other information) belongs to the declared entity (the subject) and has some specific value. That is why it is also called a Public Key Certificate. The certificate is usually signed by a trusted Certification Authority (CA) or it can be self signed.

CERTivity can handle X.509 certificates types, both version 1 and 3.

In order to open a standalone existing certificate, click on **`Menu File > Open > Open Certificate`**. After the certificate file (with .cer or .crt extension) is selected, it will be opened in a new tab which is named after the certificate's file name. There is drag and drop support for certificate files on Microsoft Windows and Linux platforms.

Most recently used certificates can be found using **`Menu File > Open Recent File`**. A simple click on the desired certificate in the menu, will open the certificate in a new tab. If the certificate has been already opened, the certificate's tab will be activated.

If the file opened using **`Menu File > Open > Open Certificate`** or **`Menu File > Open Recent File`** contains more than one certificate, then in the left part of the new tab opened these certificates will be displayed in a tree view reflecting their hierarchy. When a certificate is selected in the tree view, the information associated with it will be displayed in the right part of the window.

The following certificate details will be displayed:

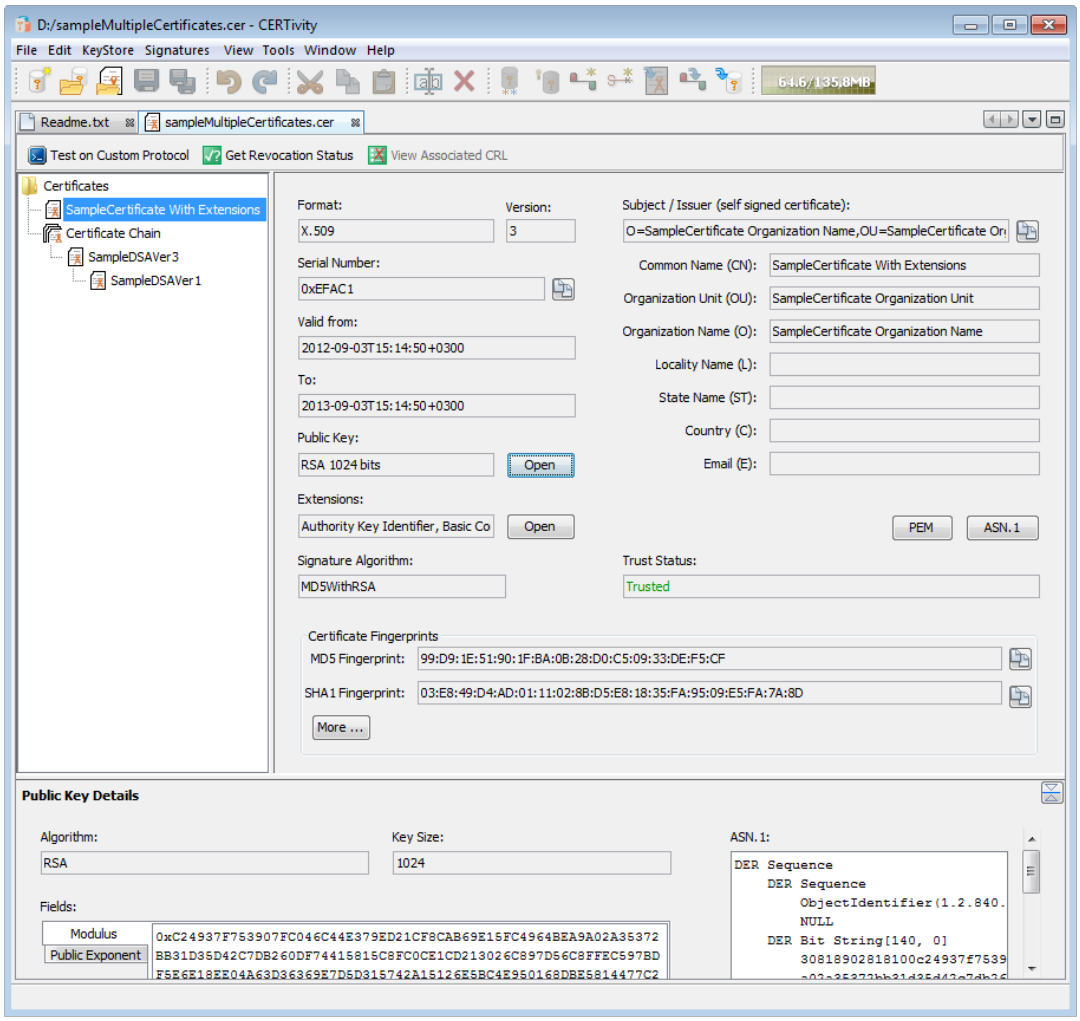
- Format;
- Version;
- Serial Number;
- Validation date period;
- Public Key;
- Signature Algorithm;
- Subject/Issuer;
- Common Name (CN);
- Organization Unit (OU);
- Organization Name (O);
- Locality Name (L);
- State Name (ST);
- Country (C);
- Email (E);
- Trust Status;
- MD5 Fingerprint;

- SHA1 FingerPrint.

In the certificate window details the following actions are available:

- Test on Custom Protocol - which will open a new window for testing the certificate against a raw TCP/IP connection with the possibility to send text requests;
- Get Revocation Status - which will open a dialog to check the revocation status;
- View Associated CRL - which will open a new tab to view the entire Certificate Revocation List associated to the certificate in case one is available in the certificate extension;
- Open public key - which will complete the window with details about the public key (algorithm, key size, modulus, public exponent, ASN.1);
- PEM - which will open a new window containing the PEM representation of the certificate;
- ASN.1 - which will open a new window containing the ASN.1 representation of the certificate.
- Display more certificate fingerprints - which will expand the list of certificate fingerprints by adding to the list the fingerprints of a certificate in the following hashes: MD2, MD4, RIPEMD-128, RIPEMD-160, RIPEMD-256, SHA-224, SHA-256, SHA-384 and SHA-512.
- Display less certificate fingerprints - which will collapse the list of certificate fingerprints by removing from the list the fingerprints of a certificate in the following hashes: MD2, MD4, RIPEMD-128, RIPEMD-160, RIPEMD-256, SHA-224, SHA-256, SHA-384 and SHA-512.

The details above, the actions and the display format are mostly the same when a Certificate is visualized from a KeyStore tab, either as a KeyStore entry or as a Key Pair entry sub-component, only that the information will appear in the Details Panel and depending on the resolution it might be scrollable and the Public Key Details will not be visible from the beginning in the view from KeyStore, but rather after opening it.

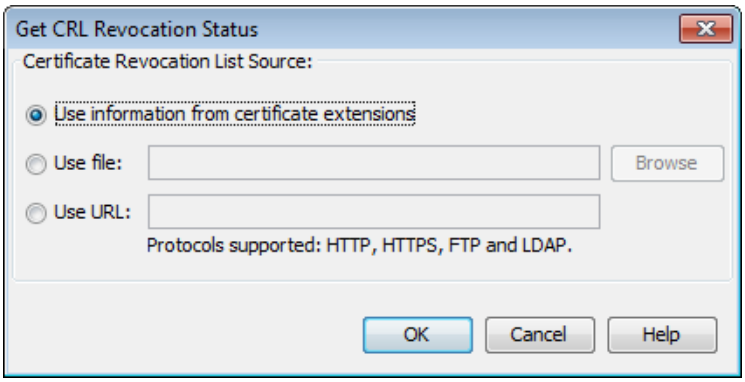


Note

You can use certificates examples provided in the distribution kit in `doc/samples/certificate` folder, to test the certificates features.

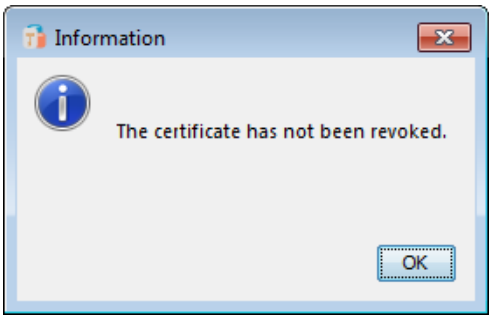
4.2 Get Revocation Status for a Certificate

Checking certificates for revocation excludes the possibility that an application or user will accept credentials that have been revoked by a Certification Authority administrator. A certificate is considered valid until its expiration date. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances include change of name, change of association between subject and Certification Authority and compromise or suspected compromise of the corresponding private key. Under such circumstances, the issuing Certification Authority needs to revoke the certificate. In order to get the revocation status of a certificate, open the certificate (click on **`Menu File > Open > Open Certificate`**) and click on **`Get Revocation Status`** button.



When checking the revocation status of a certificate, the following situations may be found:

- The certificate has been revoked;
- The certificate has not been revoked;
- No revocation information found;
- Invalid revocation information found;
- An error was encountered while trying to retrieve the revocation status of the certificate.



4.3 View Associated CRL for a Certificate

While in the certificate details window, the user can view the Certificate Revocation List associated to the selected certificate. In order to view the certificate revocation list associated to a certificate, open the certificate (click on **`Menu File > Open > Open Certificate`**) and click on **`View Associated CRL`** button. Note that the **`View Associated CRL`** button is inactive if the certificate for which the user wants to view the associated CRL does not contain extensions.

When viewing the certificate revocation list associated to a certificate, the following situations may be found:

- The certificate revocation list associated with the selected certificate is opened in a new top component window;
- An error was encountered while trying to retrieve the list of CRL distribution points URLs from the certificate;
- There are no CRL distribution points available for the selected certificate, therefore the action will end with an warning message;
- An error was encountered while trying to retrieve the certificate revocation list;
- An error occurred while trying to load the certificate revocation list;

- An I/O error occurred and the certificate revocation list could not be read;
- The list of CRL distribution points associated with the selected certificate does not contain any CRL URLs, hence the action will end with an warning message.

4.4 Test Certificate on Custom Protocol

In order to test or use a certificate against a certain TCP/IP raw text connection, you have to open it first (click on **`Menu File > Open > Open Certificate`**). After the certificate is opened, click on **`Test on Custom Protocol`** on top of the page.

This action will open a new top component window (named "Test Certificate Window"), containing the details needed for testing the certificate. The name of the tested certificate will be written in the "Currently used certificate" field. The certificate can be changed from a file chooser by clicking on **`Browse`** button.

This functionality is also available for a certificate that is part of a KeyStore.

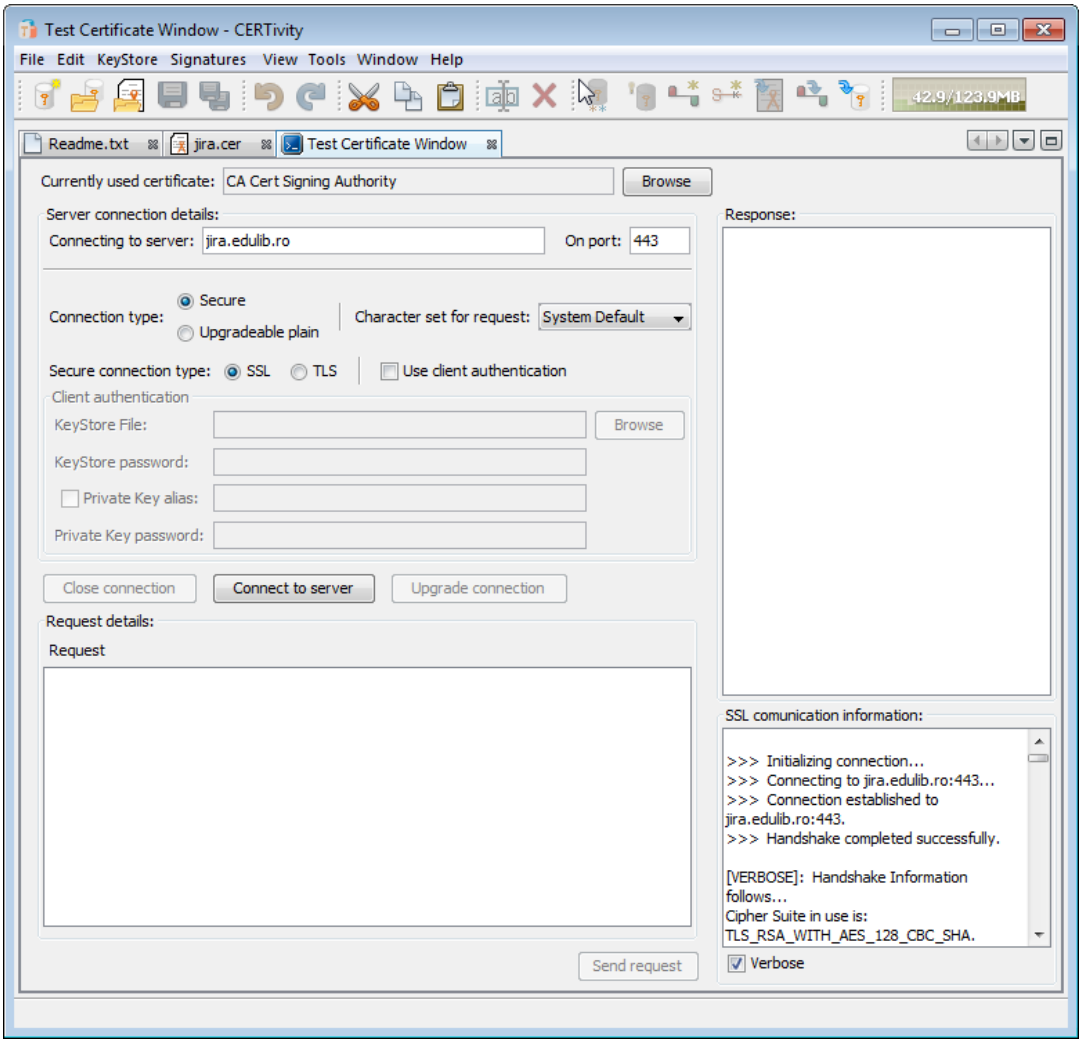
In order to use the certificate for the secure connection, the following server connection details must be filled in:

- server name;
- server port;
- connection type;
- secure connection type;
- the charset used for the request;
- client authentication (KeyStore file - can be chosen using **`Browse`** button; KeyStore password; key pair password, alias). If a Private Key alias will not be provided the default Java selection behaviour of the private key from the provided KeyStore will be used.

The connection to the server can be initiated, closed or upgraded using the corresponding buttons on the page - the buttons changed their state accordingly.

The test window has other three areas for:

- request details area - the location where will be introduced the specific request details. A request will be sent to the server only after clicking on **`Send request`** button. Note that some protocols require line terminators for delimiting the requests. These should be added manually and the button Send Request must still be pressed.
- response area - where the response from the server will be displayed;
- SSL communication information area - where SSL information will be displayed.



You can also test connections that starts on plain and then upgrades to SSL. For example testing a STARTTLS connection type for a SMTP server would be done according to the following scenario:

1. Configure server connection details;
2. Select Connection type to be Upgradeable plain. Configure client authentication details if necessary;
3. Press `Connect to Server`;
4. Start the handshaking plain messaging by issuing the necessary text commands. Don't forget to add the line terminators before pressing `Send request`;
5. Empty the request window (e.g. select all with `Ctrl+A` and write `STARTTLS` and press `Enter`, then `Send request`;
6. If the server responds with `220 2.0.0 Ready to start TLS` or similar press Upgrade Connection. The connection will be switched to secure and the certificate provided will be used for this;
7. Continue messaging on the secured connection.

4.5 Certificate's Representations

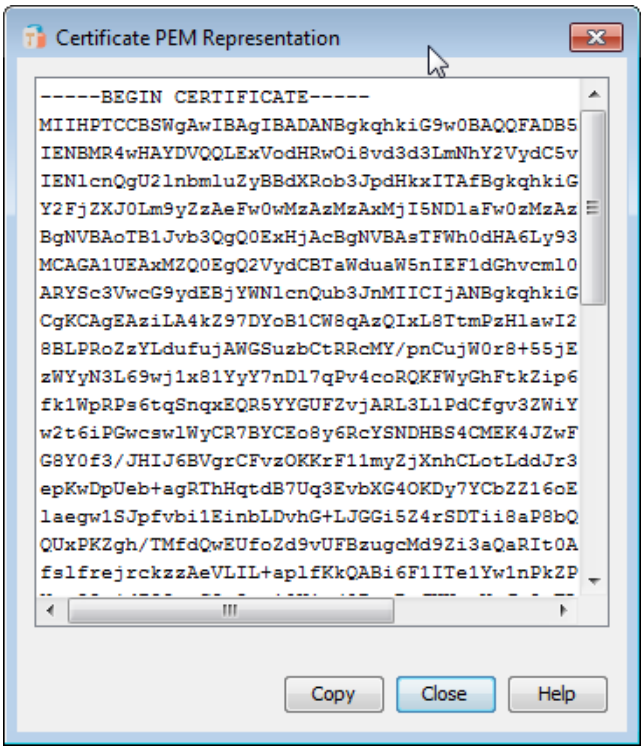
When opening an existent certificate, you can see two representations for the certificate:

- PEM;
- ASN.1 .

4.5.1 PEM

PEM (Privacy Enhanced Mail) Base64 encoded DER certificate, enclosed between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----". In order to see the PEM representation for an existing certificate, open the certificate (click on **`Menu File > Open > Open Certificate`**) and click on **`PEM`** button.

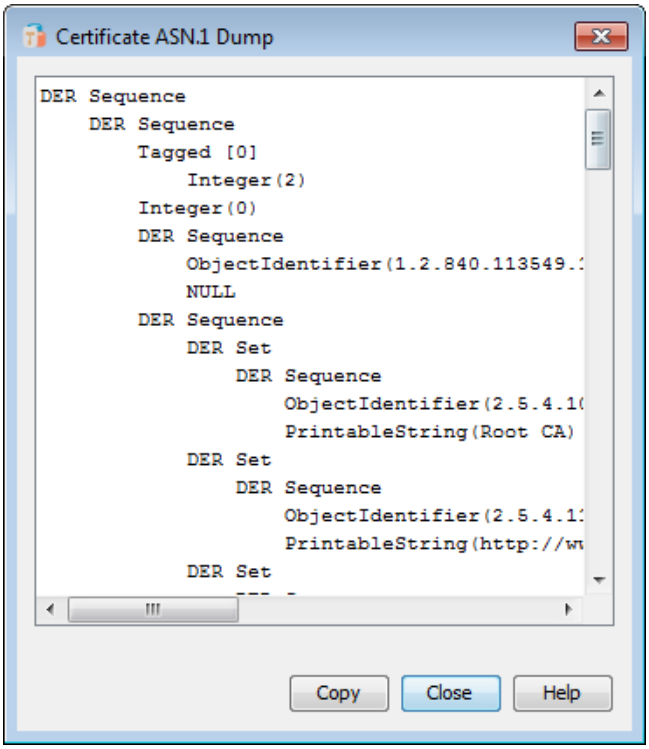
You can copy the content of the PEM certificate representation by clicking on **`Copy`** button.



4.5.2 ASN.1

Abstract Syntax Notation One (ASN.1) is a standard and flexible notation that describes data structures for representing, encoding, transmitting, and decoding data. DER(Distinguished Encoding Rules) is a subset of Basic Encoding Rules (BER) is used in situations when a unique encoding is needed, such as in cryptography and ensures that a data structure that needs to be digitally signed produces a unique serialized representation.

In order to see the ASN.1 representation for an existing certificate, open the certificate (click on **`Menu File > Open > Open Certificate`**) and click on **`ASN.1`** button. You can copy the content of the ASN.1 certificate representation by clicking on **`Copy`** button.



4.6 Certificate's Public Key

In order to see more detail information about a certificate's public key, open the certificate (click on **`Menu File > Open > Open Certificate`**) and click on **`Open`** button located near the **`Public key`** field. The Public Key Details will be displayed in the bottom part of window. There you can find information about:

- algorithm;
- key size;
- ASN.1;
- modulus;
- public exponent.

4.7 Certificate Signing Request

A Certificate Signing Request (CSR) consists of a distinguished name, a public key, and optionally a set of attributes, collectively signed by the entity requesting certification. Certification requests are sent to a certification authority, which transforms the request into an X.509 public-key certificate.

The most widely used syntax for a CSR is defined by the PKCS#10 specification. Another, far less common CSR format is the Signed Public Key and Challenge (SPKAC) format, which was defined by Netscape for use inside their browsers.

It is possible to encode a PKCS#10 CSR in binary or text formats. The text or PEM (Privacy Enhanced Mail) formatted CSR is the binary CSR after it has been Base-64 encoded to create a text version of the CSR. It also includes additional header and footer lines which enclose the Base-64 and provide an indication of the content. See below an example of an PEM encoded CSR:

```
-----BEGIN CERTIFICATE REQUEST-----
MIICWTCCAhkCAQAwbDEWMBQGA1UEAwwNU2FtcGxlRFNBVmVyMTEoMCYGA1UECwwf
U2FtcGxlRFNBVmVyMSBPcmdhbml6YXRpb24gVW5pdDEoMCYGA1UECgwfU2FtcGxl
RFNBVmVyMSBPcmdhbml6YXRpb24gTmFtZTCCAYEwggEIBgcqhkjOOAQBMIH8AnEA
iNTiphpbWQwggdfkTC9c/rggs/keJUpQuWKogbxHBrAxJYSifUpmzlejxuly03VR
gt5medn36ngZnOLPAW0P7lq9R0+xO+PoZBVD8bs+Wc/eb/SH+/kzpaR5ez5WAjtm
w/TgMf6j5YmYNdP02la1gQIVAIJQzWQ324cxbnFLquobenh30ygrAnAipIep41Z4
MXGlth5S9F3YlnpgVzF0AZyxlYEbNuP9DoRe8VJSKaQpyXMGBHNUi5h+onxuMApg
YjN4V0wa3/kg0w38wbe8h3DvKiXlKl4tjSkB0SSvLT/cN+iraO7JPIC/P5cbTQg2
SYa7rfTe/ycLA3MAAnAWIIF8yI4+o1bjpVQ+6kchHpULl8DYXh7NE4TPkBokb5dH
ZlsEbDbF50I7mSSKPSLrHh5cfj001aYxXYoCkLM1ua+FNKnnIcEEN4z/CL0G1ZLx
76Wo/a+z10WFrzaZ+LhZal5QvXVPoiKn8zZSGIVfoCEwHwYJKoZIhvcNAQkHMRIM
EHNhbXBsZS1jaGFsbGVuZ2UwCQYHKoZIzjgEAwMvADAsAhQTY7AYtpdMX3aJMy4h
8F2dNOXa/QIURPf9efxkDwPbdbWl3ldAgeFjEmM=
-----END CERTIFICATE REQUEST-----
```

See also a CSR file in SPKAC format:

```
SPKAC=MIICQzCCASswggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCdYpr9Vja/ui3
oyaJpRbgXXNBYSy2x33iyM2JjHTagQLNu6Wq/LV9SOb0DQZtM5H/2D/fHrl8Bpmjq3LZHW0PbclT8fj+yziVkadCfnNkk
QGUuEx+Dc8aSjHukIf3wxYPtnEPZTbxfmPfwog1oLGxVk3IJIEPmoCz9gGIFMRGIagoBgWe9aTDOTu8XvmBHcG9wNFfBn
Q+PMpvLwGfUKMWUVHwPYR1jAz4EQ4YhXp108MS2AuV1q2TfHuNWRBk4fvfO4fVn7EdpV4dGPprMllegn9SY+45M/ITwfR
slAg8PEd2Pr/r5Lrytk1ODRbYvOPOuk9A2pr4cTEsrvxUvokpfAgMBAAEWAzEyMzANBgkqhkiG9w0BAQIFAAOCAQEAWI8
ImIxzO00sN1d6VFAcbVGyxg+mQAl5p/eTpBny2mfiC/eM0uKE58t2VnrV7j1QCNq/v2AdmfZ0uFO8Swby2sPLipWqF9ji
vSn1HGI8bTgiw2KTLOf0QfRzppFr1ROp5Ljlz4cOBoPfIP/JDUWpIbkehOx+H/Ej1TZ5R5Bd82IGvDObya14jcsNMbSWa
ihoikotRreK0akfZzdttTw0f6/NuaDN+tfGuPKR31Qc7hW2yneWRedJYXLhvSLv7RlHJd9JIIb6TwQbeNFwvIJFi2/c93
t4HteaFib1M502jErVOFc72am8jMjxg0rsYM69S1KZ4iX3GI8s7qSTNaz9ZA==
CN=SampleMD2RSAVer1
OU=SampleMD2RSAVer1 Organization Unit
O=SampleMD2RSAVer1 Organization Name
```

4.7.1 Open Certificate Signing Request

In order to open a standalone existing CSR file, click on **`Menu File > Open > Open CSR`**. After the CSR file (with .p10, .csr, .pem or .spkac extension) is selected, it will be opened in a new tab which is named after the CSR's file name. There is also drag and drop support for CSR files on Microsoft Windows and Linux platforms.

Most recently used CSR files can be found using **`Menu File > Open Recent File`**. A simple click on the desired CSR file in the menu, will open the CSR in a new tab. If the CSR file has been already opened, the CSR's tab will be activated.

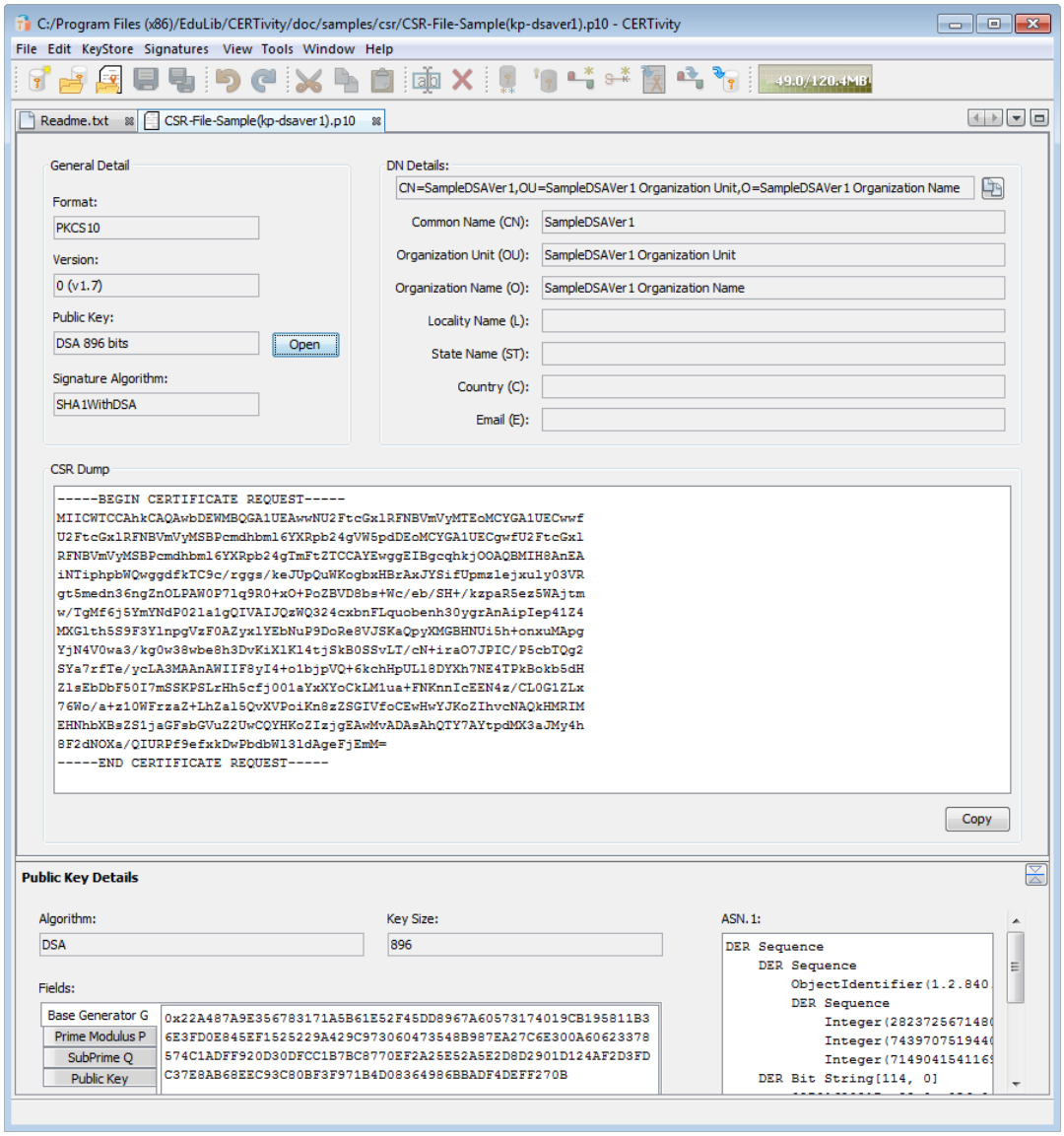
4.7.2 Certificate Signing Request Details

The following CSR details will be displayed:

- Format;
- Version;
- Public Key;
- Signature Algorithm;
- DN Details;
- Common Name (CN);
- Organization Unit (OU);
- Organization Name (O);
- Locality Name (L);
- State Name (ST);
- Country (C);
- Email (E);
- CSR Dump;

In the CSR window details the following actions are available:

- Open public key - which will complete the window with details about the public key (algorithm, key size, modulus, public exponent, ASN.1);
- Copy - which will copy into the clipboard the content of the CSR file;



Note

You can use CSR examples provided in the distribution kit in `doc/samples/csr` folder, to test the CSRs features.

4.8 Certificate Revocation Lists (CRL)

A Certificate Revocation List (CRL) is a list of certificates that have been revoked. This list contains, more exactly, the serial numbers of the certificates which have been revoked together with other information such as revocation date and additional extensions which

contain more details about the revoked certificates and the revocation reasons. The CRL also contains some global information attributes such as the version, signature algorithm, issuer name, issue date of the CRL and next update date.

The most common type of Certificate Revocation Lists is X.509 v2 and are usually encoded in DER (binary) or PEM (text) formats.

An example of a PEM encoded CRL can be seen below:

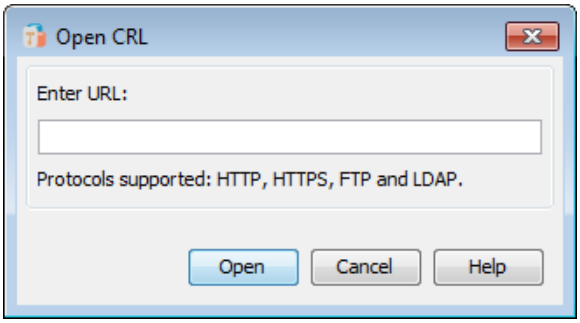
```
-----BEGIN X509 CRL-----
MIIDFDCCAfwCAQEwDQYJKoZIhvcNAQEFBQAwXzEjMCEGA1UEChMaU2FtcGxlIFNp
Z25lciBPcmdhbml6YXRpb24xGzAZBgNVBAsTElNhbXBsZSBTaWduZXIgVW5pdDEb
MBkGA1UEAxMSU2FtcGxlIFNpZ25lciBDZXJ0Fw0xMzAyMTgxMDMyMDBaFw0xMzAy
MTgxMDQyMDBaMIIBNjA8AgMUeUcXDTEzMDIxODEwMjIxMlowJjAKBgNVHRUEAwoB
AzAYBgNVHRgEERgPMjAxMzAyMTgxMDIyMDBaMDwCAxR5SBcNMTMwMjE4MTAyMjIy
WjAmMAoGA1UdFQQDCgEGMBgGA1UdGAQRGA8yMDEzMDIxODEwMjIwMFowPAIDFHlJ
Fw0xMzAyMTgxMDIyMzJaMCYwCgYDVR0VBAMKAQQwGAYDVR0YBBEYDzIwMTMwMjE4
MTAyMjAwWjA8AgMUeUoXDTEzMDIxODEwMjI0MlowJjAKBgNVHRUEAwoBATAYBgNV
HRgEERgPMjAxMzAyMTgxMDIyMDBaMDwCAxR5SxcNMTMwMjE4MTAyMjUxWjAmMAoG
A1UdFQQDCgEFMBgGA1UdGAQRGA8yMDEzMDIxODEwMjIwMFqgLzAtMB8GA1UdIwQY
MBaAFL4SAcyq6hGA2i6tsurHtfuf+a00MAoGA1UdFAQDAgEDMA0GCSqGSIb3DQEB
BQUAA4IBAQBCIb6B8cN5dmZbziETimiotDy+FsOvS93LeDWSkNjXTG/+bGgnrm3a
QpgB7heT8L2o7s2QtjX2DaTOSYL3nZ/Ibn/R8S0g+EbNQxdk5/la6CERxiRp+E2T
UG8LDb14YVMhRGKvCguSIyUG0MwGW6waqVtd6K71u7vhIU/Tidf6ZSdsTMhpPPFu
PUid4j29U3q10SGFF6cCt1DzjvUcCwHGhHA02Men70EgZFADPLWmLg0HglKUh1iZ
WcBGtev/8VsUijyjsM072C6Ut5TwNyrrthb952+eKlmxLNgT0o5hVYxjXhtwLQsL
7QZhrypAM1DLYqQjkiDI7hlvt7QuDGTJ
-----END X509 CRL-----
```

4.8.1 Open a Certificate Revocation List

CERTivity allows opening Certificate Revocation Lists which are stored in local files, or from a remote location, using a given URL address which identifies the location of a CRL. To open a CRL the following actions have to be performed:

- **For a CRL stored in a local file:** Click on **`Menu File > Open > Open CRL > From File`**. A file chooser will appear allowing to select one or more CRL files (having either **`.crl`** or **`.pem`** extension). If a CRL that has to be opened has a different extension, an "All files" filter is available in the file chooser which allows selecting any file. After selecting one or more files, press Open, and each selected CRL will be opened in a different tab which will display the content details of the CRL. There is also drag and drop support for CRL files on Microsoft Windows and Linux platforms.
- **For a CRL from a remote location:** Click on **`Menu File > Open > Open CRL > From URL`**. A dialog will appear requesting to enter the URL of the CRL. The CRL found at the location denoted by the given URL will be opened into a new tab. If the CRL is large, a progress bar will be displayed on the status bar until the CRL content is retrieved from the remote location. If the URL is invalid, an error message will be displayed informing that, and the user can enter another URL.

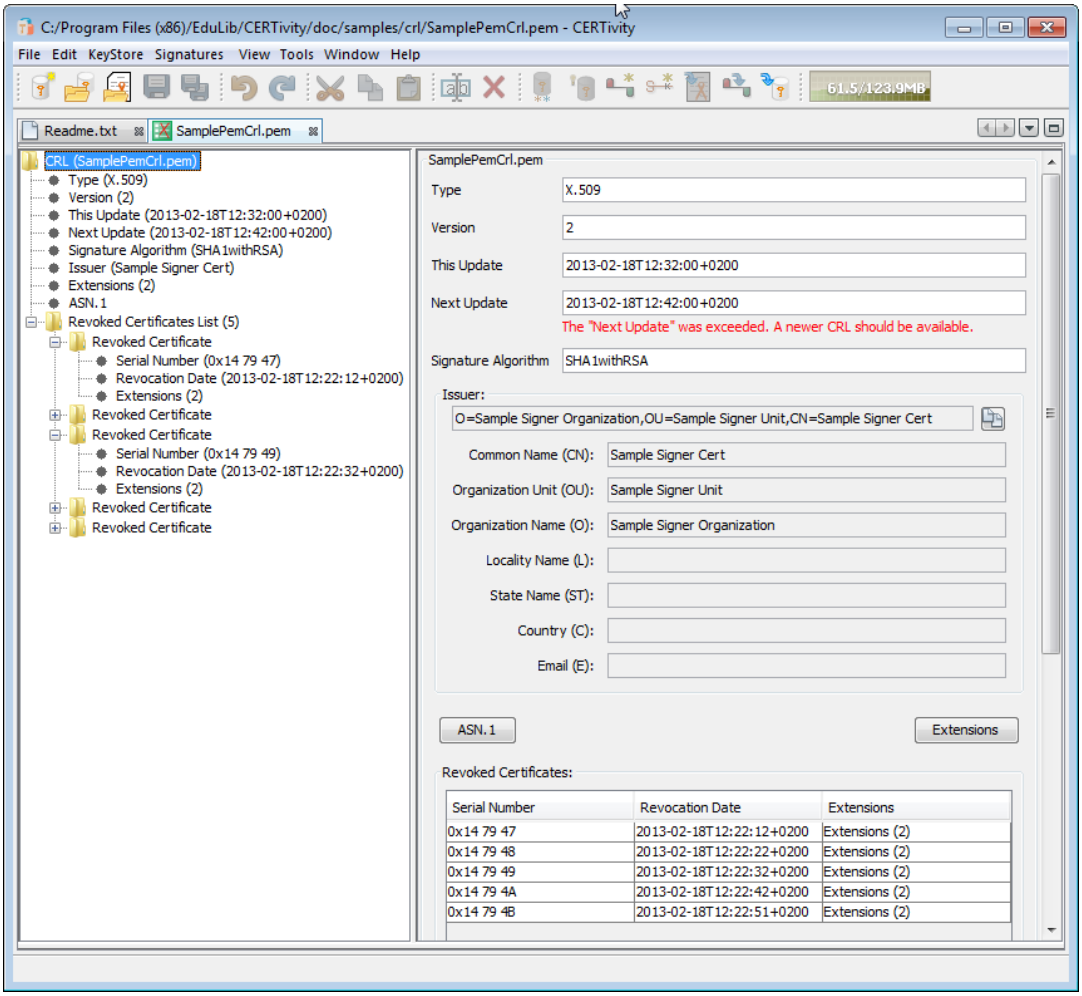
  The dialog for entering the URL can be seen below:



After opening and closing more Certificate Revocation Lists, the most recently used CRLs can be found using **`Menu File > Open Recent File`**. For the CRLs which were opened from local files the entire file path will be displayed in the menu that appears, while for the the ones opened from URLs, the URL will be displayed. A simple click on the desired CRL in the menu, will open it in a new tab. If the CRL has been already opened, the CRL's tab will be activated.

4.8.2 CRL Details

CERTivity displays the content of the CRL using a tree like structure for each field or group of fields of the CRL as it can be seen in the screenshot below:



Each node of the CRL tree contains the name of the field and its value in brackets, if the value is short enough to be displayed, like for **`Type`**, **`Version`**, **`This Update`**, **`Next Update`**.

For each selected node in the CRL tree, the content of the selected node will be displayed in the right panel. When the root of the tree is selected (selected by default when opening the CRL), the right panel will display the entire content of the CRL (as it can be seen in the example screenshot from above).

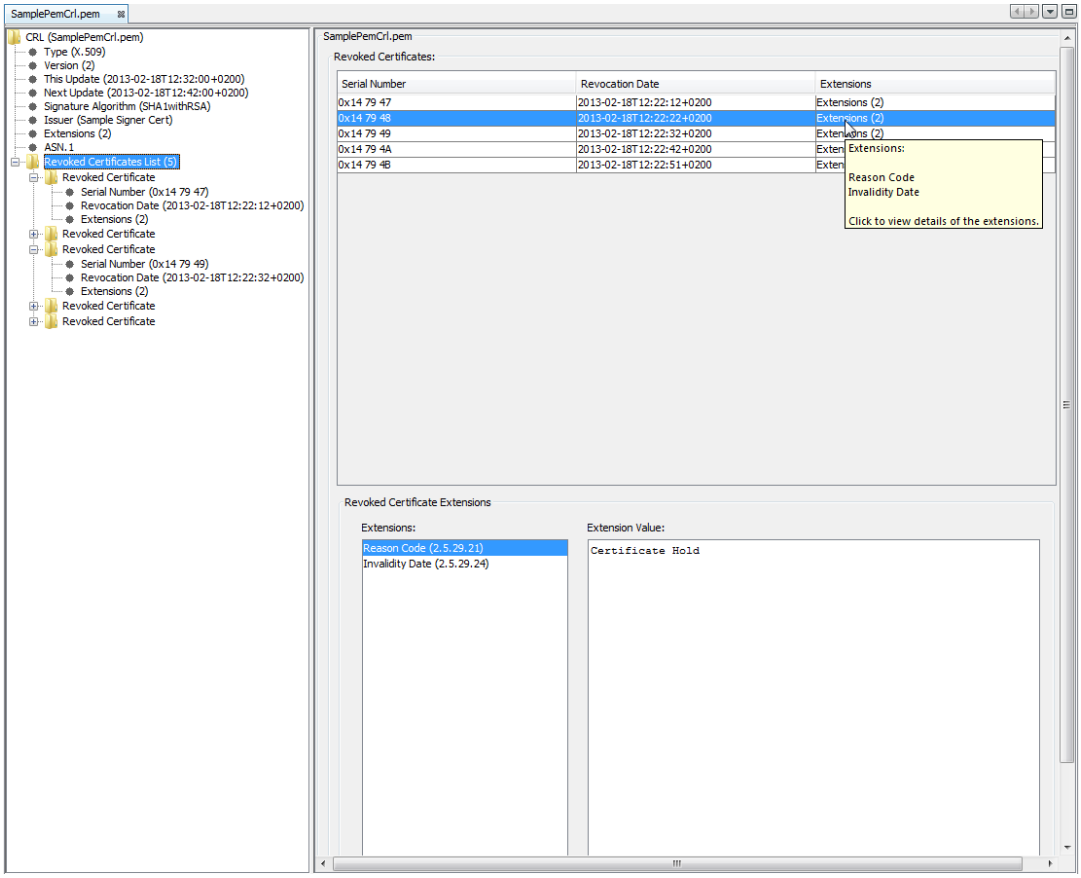
In this full display mode (selecting the root node of the tree), the ASN.1 representation and the CRL extensions are not displayed by default but the user can make them visible by clicking

on the **ASN.1** and **Extensions** buttons, which will expand the panel with the corresponding additional content.

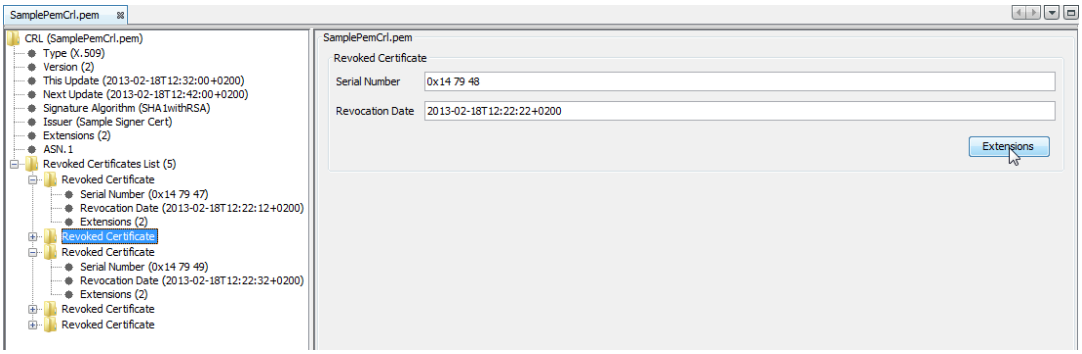
Also, when the **root node** of the CRL tree is selected, the revoked certificates are displayed at the bottom of the right panel as a list containing for each revoked certificate the **Serial Number**, **Revocation Date**, and **Extensions**. The Extensions column displays informations only about the number of extensions if the revoked certificate has extensions. To view the extensions of a certain revoked certificate, select the corresponding row of the table, and an additional panel will appear at the bottom of the table containing details about the extensions of the selected revoked certificate. The names of the available extensions of a revoked certificate can be viewed faster in the tooltip which appears when positioning the cursor over the revoked certificate row.

Note

The same view of the revoked certificates and their extensions can be obtained by selecting the **Revoked Certificates List node** from the tree, as it can be seen in the screenshot below:



If the revoked Certificates List node is expanded, each revoked certificate can be visible as a child node, which can also be expanded further to see the fields of the Revoked Certificate (`Serial Number`, `Revocation Date` or `Extensions`). If the `Revoked Certificate` node is selected, the right panel will display the fields contained by the selected revoked certificate, as it can be seen below:



Note

The extensions of the revoked certificate can also be seen by clicking the Extensions button, which will trigger the displaying of the revoked certificate's extensions. Also, each field of the revoked certificate including the extensions, can be seen individually by selecting the corresponding field in the child nodes of the revoked certificate in the CRL tree.

4.8.2.1 CRL Fields

The CRL Viewer from CERTivity allows viewing the content of the following CRL fields:

- **`Type`**

  The type of the CRL. In most cases, this is X.509;

- **`Version`**

  The version of the CRL. In most cases, the version is 2;

- **`This Update`**

  This field indicates the issue date of the current CRL;

- **`Next Update`**

  This field indicates the date by which the next CRL will be issued. As mentioned in RFC 5280 [http://www.ietf.org/rfc/rfc5280.txt], the next CRL could be issued before the indicated date, but it will not be issued any later than the indicated date.

  When displaying this field is selected in the CRL tree and is displayed on the panel in the right side, the date held by it is verified against the current date, and if the date is exceeded, a red notice message will be displayed under the field as a reminder that maybe a newer CRL has been issued;

- **`Signature Algorithm`**

  The algorithm that was used for the signature of the current CRL;

- **`Issuer`**

  The name of the entity that signed and issued the CRL. In this field, the issuer identity is carried. Alternative name forms may also appear in the `Iissuer Alternative Name` extension.

The value of this field is not displayed entirely in the CRL tree next to the node between brackets, only a part of it is shown. The full issuer details can be seen in the right panel when clicking on the node;

- **`Extensions`**

  The extensions of the CRL. According to the RFC 5280 [http://www.ietf.org/rfc/rfc5280.txt], this field may only appear if the version is 2. If present, this field contains one or more CRL extensions.

  The value of this field is not displayed entirely next to the node between brackets. Only the number of extensions is displayed. The full details about the extensions can be seen in the right panel when clicking in the extensions node;

- **`Revoked Certificates`**

  This field contains the list of revoked certificates. When there are no revoked certificates, this list is absent.

  The revoked certificates are displayed as child nodes of this CRL tree node. The `Revoked Certificates List` node on its own displays only the number of revoked certificates. Also, the list of revoked certificates can be seen in the right panel by clicking on the `Revoked Certificates List` node.

The CRL tree contains one more node, **`ASN.1`**, which contains the ASN.1 representation of the current CRL. The value of this node can be viewed in the right panel when it is selected.

4.8.2.2 CRL Extensions

CRL extensions provide methods for associating additional attributes with CRLs. These extensions can be marked as critical or non-critical.

CERTivity can display the following CRL extensions, defined in the RFC 5280 [http://www.ietf.org/rfc/rfc5280.txt]:

- **`Authority Key Identifier`**

  This extension provides a means of identifying the public key corresponding to the private key used to sign a CRL;

- **`Issuer Alternative Name`**

  This extension allows additional identities to be associated with the issuer of the CRL;

- **`CRL Number`**

  This extension contains a monotonically increasing sequence number for a given CRL scope and CRL issuer. This extension allows users to easily determine when a particular CRL superseedes another CRL. CRL numbers also support the identification of complementary complete CRLs and delta CRLs;

- **`Delta CRL Indicator`**

  The delta CRL indicator identifies the CRL as being a delta CRL. Delta CRLs contain updates to revocation information previously distributed rather than all the information that would appear in a complete CRL;

- **`Issuing Distribution Point`**

This extension identifies the CRL distribution point and scope for a particular CRL and it indicates wether the CRL covers revocation for end entity certificates only, CA certificates only, attribute certificates only or a limited set of reason codes;

- **`Freshest CRL`** (or **`Delta CRL Distribution Point`**)

  This extension identifies how delta CRL information for this complete CRL is obtained.

- **`Authority Information Access`**

  This extension defines the use of the Authority Information Access extension in a CRL.

Also, the following CRL Entry extensions can be displayed by CERTivity:

- **`Reason Code`**

  This extension identifies the reason for the certificate revocation. The possible reason codes are:

  - `unspecified;`
  - `keyCompromise;`
  - `cACompromise;`
  - `affiliationChanged;`
  - `superseded;`
  - `cessationOfOperation;`
  - `certificateHold;`
  - `removeFromCRL;`
  - `privilegeWithdrawn;`
  - `aACompromise.`

- **`Invalidity Date`**

  This extension provides the date on which it is known or suspected that the private key was compromised or that the certificate otherwise became invalid. This date may be earlier than the revocation date in the CRL entry;

- **`Certificate Issuer`**

  This extension identifies the certificate issuer associated with an entry in an indirect CRL, that is, a CRL that has the `indirectCRL` indicator set in its issuing distribution point extension. When present, the certificate issuer CRL entry extension includes one or more names from the issuer field and/or issuer alternative name extension of the certificate that corresponds to the CRL entry.

More information about the CRL and CRL Entry extensions can be found in the RFC 5280 [http://www.ietf.org/rfc/rfc5280.txt].

4.8.2.3 Revoked Certificates

The revoked certificates (if present) can be viewed in CERTivity either by selecting the `Revoked Certificates List node`, which will display a table in the right panel containing the information about these certificates, either by expanding the `Revoked`

`Certificates List node` and selecting its child nodes, which will display the available fields of the revoked certificate in the right panel. Also, each revoked certificate node can be expanded to see individual fields.

For a revoked certificate, the following fields will be displayed:

- **`Serial number;`**
- **`Revocation Date;`**
- **`Extensions.`**

The number of revoked certificates present in the CRL can be seen next to the `Revoked Certificates List` node, in brackets.

5. CERTivity®'s KeyStore

5.1 KeyStores Capabilities

A KeyStore is a protected database of cryptographic keys - private, public, secret. Private keys in a KeyStore have a certificate chain associated with them, which authenticates the corresponding public key - together they form a Key Pair entry - you cannot have just a private key by its own. On the other hand a KeyStore can contain just the certificates from trusted entities.

A Certificate embeds a public key belonging to an entity. It certifies the public key and all the information via digitally signature of another entity (the issuer, e.g. - a person, company, etc.), saying that the embedded public key (and some other information) belongs to the declared entity (the subject) and has some specific value. That is why it is also called a Public Key Certificate. The certificate is usually signed by a trusted Certification Authority (CA) or it can be self signed.

CERTivity can handle X.509 certificates types, both version 1 and 3.

Besides Key Pair and Certificate entries (asymmetric keys) some types of KeyStores can store Secret Keys (symmetric keys) as well.

Hence a KeyStore is a protected collection of Key Pair, Certificate and Secret Keys entries and each such entry is addressable via an unique alias or entry name. KeyStores are stored according to their standards and they are protected by a general password while the Private Keys and Secret Keys are protected by different individual passwords.

CERTivity asks for these passwords when operations are requiring access to the keys. Once a Private key or Secret Key is unlocked it will stay unlocked while the KeyStore is loaded.

CERTivity can manage the following KeyStore types - their main capabilities according to their standard are described below.

Table 5.1. KeyStores capabilities

Keystore type	Keystore password protection	Supports Secret Key	Aliases Case Sensitive	Provider
jks - Java KeyStore (Oracle's KeyStore format)	Yes	No	No - use lower case	Default JCE
pkcs12 - Public-Key Cryptography Standards #12 KeyStore (RSA's Personal Information Exchange Syntax Standard)	Yes (for password that is greater than 7 characters, you may need to download and install the Java Cryptography Extension (JCE) Unlimited	No	Half - Case aware	Bouncy Castle

Keystore type	Keystore password protection	Supports Secret Key	Aliases Case Sensitive	Provider
	Strength Jurisdiction Policy Files)			
jceks - Java Cryptography Extension KeyStore (More secure version of JKS)	Yes	Yes	No - use lower case	Default JCE
bks - Bouncy Castle KeyStore (Bouncy Castle's version of JKS);	Yes. Note the empty string ("") universal password. If the KeyStore is unlocked using the universal password, and if the password is not changed until saving the KeyStore, the empty string will be set as the KeyStore password when saving.	Yes	Yes	Bouncy Castle
uber - Bouncy Castle UBER KeyStore (More secure version of BKS)	Yes	Yes	Yes	Bouncy Castle
Windows Root CA	Yes	No	Yes	Default JCE (on Oracle - SunMSCAPI)
Windows User	Yes	Yes	Yes	Default JCE (on Oracle - SunMSCAPI)

Please note that PKCS12 KeyStores have no password protection for their key pair entries.

"Case aware" means that an alias can be defined both with low case and upper case, will be saved as this, but there cannot be two aliases which differ just by the case of their letters.

Working with Windows Root CA KeyStore and Windows User KeyStore are available only on Windows platform and additional confirmation and warning panels will be displayed by the Windows system when installing, deleting, renaming a KeyStore entry. Hence, the second confirmation dialogs are not under the control of CERTivity application.

The BKS type of KeyStore allows for being accessed both with the KeyStore password, as well as with the empty string password - this is not under the control of the CERTivity application .

Note

You can use KeyStore examples provided in the distribution kit in the folder `doc/samples/keystore`, to test the KeyStore features.

5.2 KeyStore Interface Organization

The KeyStore Interface contains two main sections:

- Tree Table panel allowing the navigation between the entries;
- Details Panel for visualizing the current selection.

The tree table contains the following sortable columns:

- **Entry Alias** - e.g. the name of the certificate, the name of the key pair;
- **Key Type** - e.g. RSA 2048, DSA 512 (the type of the key pair or of the certificate);
- **Expire Date** - the expiration date of the certificate or key pair - e.g. 26.10.2011 17:43:03 (DD.MM.YYYY HH:MM:SS);
- **Last Modified** - the date of the last modification of the certificate or key pair - e.g. 26.10.2011 17:43:03 (DD.MM.YYYY HH:MM:SS);
- **Validity Status** - the status of the entry (e.g. - valid, expired, about to expire);
- **Trust Status** - the trust status of the certificate entry. It can take one of the following 3 values:
  - `Trusted` - if a Trust Path could be established using the provided TrustStores and Trust Path options;
  - `Not Trusted` - if a Trust Path could not be established using the provided TrustStores and the Trust Path options;
  - `N/A` - if no TrustStores were set.

Each entry type has a specific icon.

	Certificate entry.
	Certificate Chain entry.
	Unlocked Key Pair entry.
	Locked Key Pair entry.
	Unlocked private key entry.
	Locked private key entry.
	Public key entry.
	Certificate Extensions entry.

	Unlocked Secret Key entry.
	Locked Secret Key entry.

After selecting a KeyStore entry, when clicking on the right button of the mouse, you can use the contextual menu. More details about the contextual menus are available in **CERTivity's Menus / Tool bar / Contextual menu**.

The Details Panel is polymorphic, changing according to the current selection type. Using the Details Panel section, you can get more details about the selected entry:

- for certificate entry, you can see the Certificate Details;
- for key pair entry, you can see the Private Key Details and also Certificate Chain Details;
- for certificate chain entry, you can see Certificate Chain Details where you can select a certificate and see its details;
- for Public Key entry, you can see Public Key Details like ASN.1, algorithm, key size;
- for Private Key entry, you can see Private Key Details like ASN.1, algorithm, key size.

The Details Panel can be minimized and maximized, by clicking on the top right corner button of the panel.

5.3 Create a New KeyStore

In order to create a new KeyStore, click on **Menu File > New KeyStore** or use the default keyboard shortcut **CTRL+N**. A new window for the new created KeyStore will be opened. The Create New KeyStore File dialog is more complex than in a standard MDI application, because there are more settings to be bound from the beginning such as the KeyStore password and type. Protection being an important factor it is important to bind the KeyStore file name with the password from the beginning.

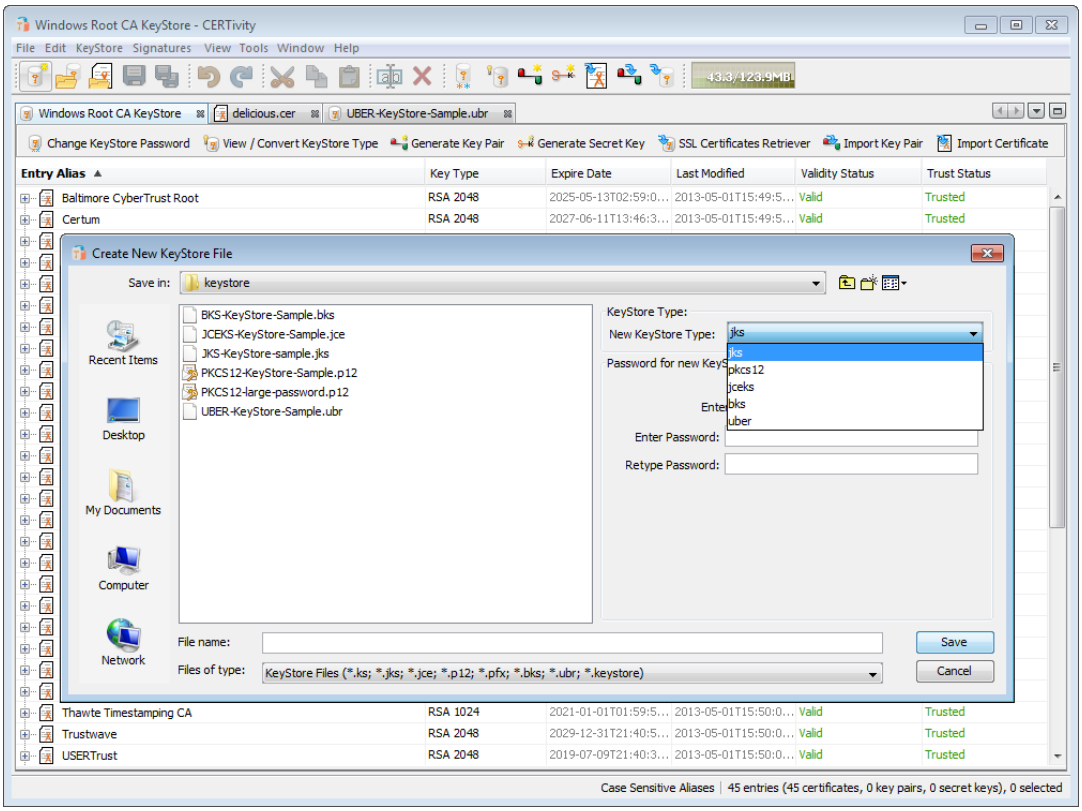
The available KeyStore types are:

- jks - Java KeyStore (Oracle's KeyStore format);
- pkcs12 - Public-Key Cryptography Standards #12 KeyStore (RSA's Personal Information Exchange Syntax Standard);
- jceks - Java Cryptography Extension KeyStore (More secure version of JKS);
- bks - Bouncy Castle KeyStore (Bouncy Castle's version of JKS);
- uber - Bouncy Castle UBER KeyStore (More secure version of BKS).

When creating (and in general handling) a pkcs12 or a uber type KeyStore, longer passwords either for the KeyStore of the Key Pairs requires that you have the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files installed. If you are not allowed to install it then you will need to use smaller passwords (e.g. maximum 7 characters). Otherwise you may encounter "Illegal Key Size" errors when accessing pkcs12 or uber files or keys inside them.

Note

This is a matter of import control restrictions in most of the countries and is not related to technical reasons.



5.4 Open an Existing KeyStore

In order to open an existing KeyStore, click on **`Menu File > Open > Open KeyStore`** or use the default keyboard shortcut **`CTRL+O`**. A file chooser dialog box will be opened in order to select the desired KeyStore. The supported files have the following default filters: `cacerts; *.ks; *.jks; *.jce; *.p12; *.pfx; *.bks; *.ubr; *.keystore.`. An `All Files` filter is available, too.

After selecting the KeyStore file, the KeyStore password is required.

Most recently used KeyStores can be found using **`Menu File > Open Recent File`**. A simple click on the desired KeyStore in the menu, will open the KeyStore in a new tab. The KeyStore's password is required only in the "Persist only KeyStore file name" mode. If the KeyStore has been already opened, the KeyStore's tab will be activated.

On Microsoft Windows and Linux platforms, dragging a KeyStore from a file explorer into CERTivity will also open it.

Note

You can see a sample of :

- jks KeyStore type in the `samples` directory: `doc/samples/keystore/JKS-KeyStore-sample.jks`;
- pkcs12 KeyStore type in the `samples` directory: `doc/samples/keystore/PKCS12-KeyStore-Sample.p12`;

- jceks KeyStore type in the `samples` directory:`doc/samples/keystore/JCEKS-KeyStore-Sample.jce`;
- bks KeyStore type in the `samples` directory: `doc/samples/keystore/BKS-KeyStore-Sample.bks`;
- uber KeyStore type in the `samples` directory: `doc/samples/keystore/UBER-KeyStore-Sample.ubr`.

5.5 Open JREs CA KeyStores

An efficient way to open CA KeyStores (TrustStores) of the JREs on the current system is to use **`Menu File > Open > Open JRE CA KeyStore`**. There you have a list of the CA Truststores discovered on your system. The discovery of the JREs is done by compiling a list of paths in the following way:

- The Java property `${java.home}` of the JRE CERTivity started with;
- The system environment variables `JAVA_HOME` and `JRE_HOME`;
- For Windows platforms searching the installed Java JDKs and JREs in the Windows registry;
- For Unix and Mac we are looking for traditional Java installation directories such as `/usr/java` for Unix, `/usr/lib/jvm` for Linux (Debian, RedHat) and for Mac `/Library/Java/Home/`, `/System/Library/Java/JavaVirtualMachines/`. Various patterns are then used.

You can select a KeyStore from the TrustStore list discovered by CERTivity on your system, or you can select another one by using **`Menu File > Open > Open JRE CA KeyStore > Other...`** menu item. In this menu item you have to select the JDK's or JRE's home path, and CERTivity will open the Truststore for you. This new selected Truststore will be added to the menu list, so you will not have to make the selection steps again next time. The maximum list size of JREs CA Keystore can be set in the `Tools > Options` menu.

Before opening the selected JRE CA KeyStore CERTivity will ask for its password. The password depends on the JRE distribution, but generally it has a well known default - `changeit`.

5.6 KeyStore Persistence (Reloading opened KeyStores)

CERTivity® offers KeyStore persistence between runs by remembering the KeyStores which are opened at the time the application exits. If the KeyStores are not closed before exiting the application, their names and locations are remembered so that on the next launch they will be reloaded (if the KeyStore files still exist, and they can be loaded). When reloading the KeyStores, you can set CERTivity® to either ask you for the password of each KeyStore when each KeyStore tab is selected for the first time, or, you can set it to remember the passwords of each KeyStore as well, so that you won't be prompted for them. All passwords will be encrypted to increase safety.

Although the full persistence option (remembering the KeyStore name and encrypted password) makes the application more friendly we are not recommending the full persistence, unless you are sure the machine is exclusively accessible by you. Otherwise, it is recommended to use the option which only remembers the name and location of each KeyStore.

To change the persistence type, click on **`Menu Tools > Options`** or use the default keyboard shortcut **`ALT+T+O`**. The preferences dialog will be opened. In the **`Certificates`**

Options tab, look for the field **KeyStore persistence**. This field has a combo list which allows you to select one of the following two options:

- Fully persist (file name & encrypted password) - this is the default value when starting CERTivity® for the first time. When this option is selected, the application will save both the KeyStore name and the encrypted password of each KeyStore which is opened when exiting the application. This is not recommended if the machine is not exclusively accessed by you.
- Persist only KeyStore file name (without password) - When this option is selected, the application will only save the name and location of the KeyStore file, and you will be prompted to enter the password for each KeyStore which was previously opened when selecting the KeyStore tab for the first time after launching the application.

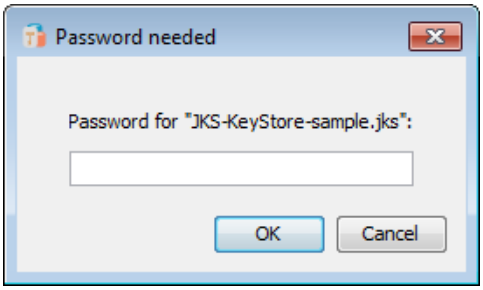
  The KeyStore is not loaded until you provide the correct password. If you enter a wrong password, you will be prompted to enter the password again. Also, if the file of the KeyStore has become corrupted since the last run, you will be informed regarding the issue. If you choose to cancel entering a password, the KeyStore tab will be closed, and the KeyStore will not be remembered for opening on the next launch anymore.

Changing the KeyStore persistence type will only take effect when clicking on OK on the preferences dialog. If the dialog is closed otherwise, the new selected persistence type will be disregarded.

Note

When changing from a lower level of persistence (from "Persist only KeyStore file name" option) to a higher level of persistency ("Fully persist" option), if you have KeyStores which have not yet been unlocked (for which you have not yet entered the password), and you still do not enter the password before exiting the application, on the next run, you will still be prompted to enter the passwords for the KeyStores for which you did not provide them when having the previous level of persistency.

A screenshot showing the dialog which prompts you to enter a password for opening the KeyStore when the application is launched, can be seen below:



5.7 Open Microsoft Windows KeyStores

These operations are functional only on Microsoft Windows platforms. The Windows system native KeyStores are opened and similar visualising and editing actions can be performed on these KeyStores with some limitations:

- Private Key Fields are not available for inspecting;
- DSA Key Pairs cannot be generated;
- Key Pairs cannot be exported or copied;

- Private Keys cannot be exported;
- Undo/Redo functionality is not available due to the fact that all the actions are persistent, no save is needed, and so, it is possible that the KeyStore gets modified from outside between undo and redo, and then the behavior may be unexpected.

Especially for the Root KeyStore a native confirmation dialog will also appear for editing actions. This is not under the control of CERTivity. It is advisable to do the same logical confirmation both in the CERTivity confirmation dialog as well as in the Windows native one. As these are the Operating System KeyStores take care when editing, especially for the Root CA KeyStore. For example when renaming a certificate entry (key pairs can not be renamed), there are 2 native pop-ups appearing: First to confirm deleting of the certificate, and the second to confirm the import of the certificate with the new alias. If on the delete dialog "YES" is selected and on the import dialog "NO" is selected, then the node gets deleted. There is no way to recover the node back.

Due to a JRE 1.6 64-bit distribution limitation opening the Windows KeyStores is not functional on Microsoft Windows 64-bits Releases. JRE 1.7 resolves this issue, as well as using a 32-bit distribution of JRE 1.6.

5.7.1 Open Windows Root KeyStore

The Windows-ROOT KeyStore contains all root CA certificates trusted by the machine.

In order to open the Windows Root KeyStore, click on **`Menu File > Open > Open Windows Root CA KeyStore`**. A new tab will be opened containing the Windows Root KeyStore entries.

Native confirmation dialogs will be displayed upon, adding, deleting.

If you want to add an entry, but the current KeyStore already contains an entry with the same SHA1 fingerprint, you will have to choose to overwrite the old entry or not because Windows Root CA KeyStore do not allow more entries with the same content. The operating system, will ask for a confirmation of deleting the entry from the Root Store and also a Security Warning from the operating system will be displayed, informing about the installing of a new entry.

5.7.2 Open Windows User KeyStore

This operation is functional only on Microsoft Windows platforms. In order to open Windows User KeyStore, click on **`Menu File > Open > Open Windows User KeyStore`**. A new tab will be opened containing the Windows User KeyStore entries.

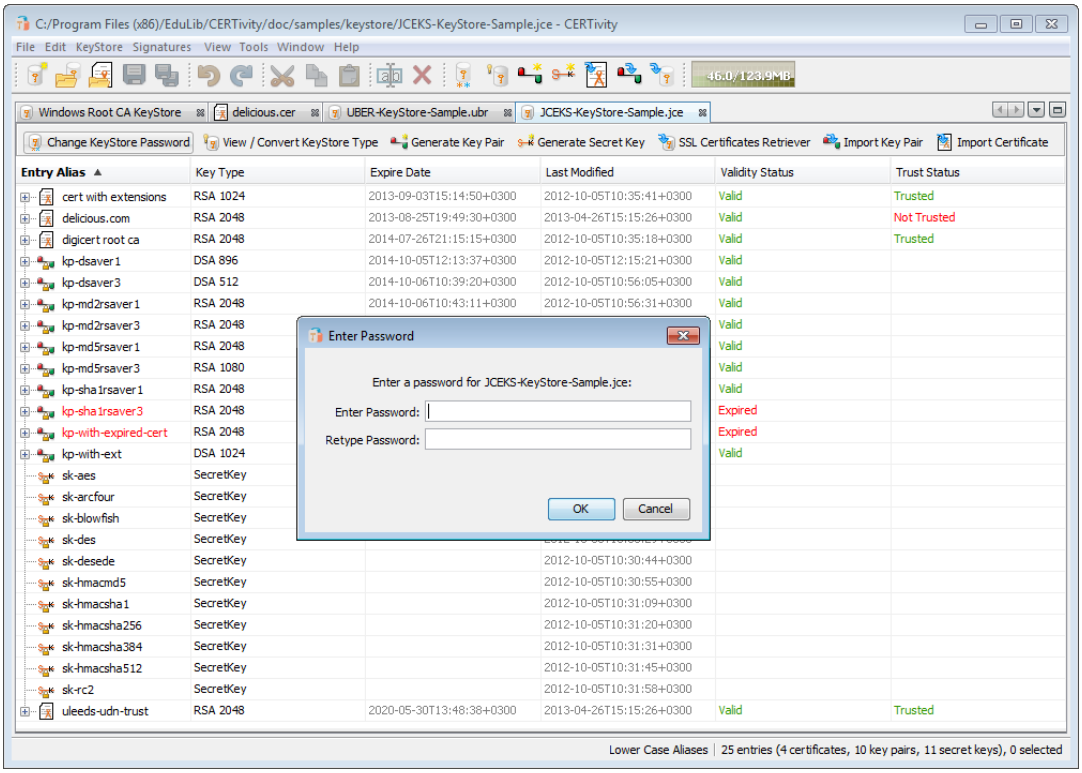
Due to a JRE 1.6 64-bit distribution limitation opening the Windows KeyStores is not functional on Microsoft Windows 64-bits Releases. JRE 1.7 resolves this issue, as well as using a 32-bit distribution of JRE 1.6. For this reason the bundled CERTivity setup is using the 32-bit distribution of JRE 1.6.

5.8 Change KeyStore Password

In order to change a KeyStore password, open the desired KeyStore (**`Menu File > Open > Open KeyStore`**) and click on "**Change KeyStore Password**" or use the toolbar

A new dialog will be opened, for entering the new password. The password must be retyped. In case of error the message "`Entered passwords do not match`" will appear.

A screenshot for the Change KeyStore Password action can be seen below:



5.9 View/Convert KeyStore Type

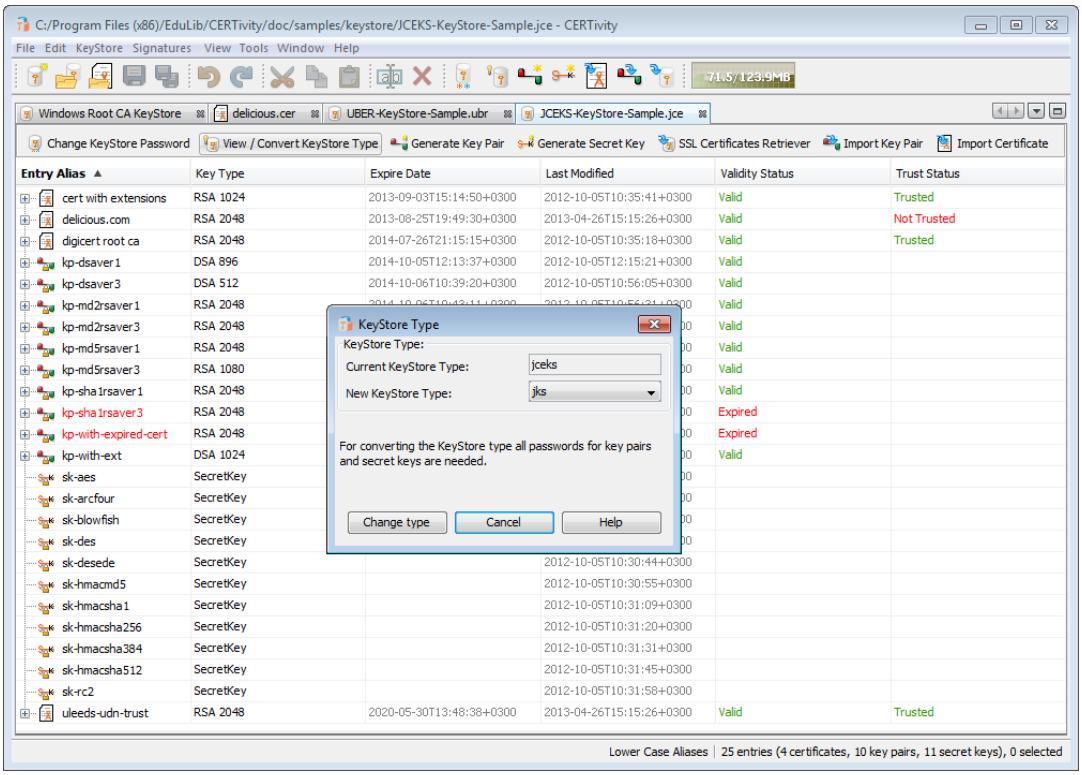
In order to view/convert a KeyStore type, click on **`View/Convert KeyStore Type`** of the opened KeyStore window. The available KeyStore types are:

- jks - Java KeyStore (Oracle's KeyStore format);
- pkcs12 - Public-Key Cryptography Standards #12 KeyStore (RSA's Personal Information Exchange Syntax Standard);
- jceks - Java Cryptography Extension KeyStore (More secure version of JKS);
- bks - Bouncy Castle KeyStore (Bouncy Castle's version of JKS);
- uber - Bouncy Castle UBER KeyStore (More secure version of BKS).

After selecting the new desired KeyStore type, click on **`Change type`** button.

For converting the KeyStore type all passwords for key pairs and secret keys are needed!

A screenshot for the change KeyStore password action can be seen below:



Note

When converting to a pkcs12 KeyStore type, the KeyStore password and the entry passwords will be lost (because the pkcs12 has no passwords.

When converting from pkcs12 to any other KeyStore type, passwords will be required.

Note

In uber KeyStore type the alias name is case sensitive.

5.10 View Certificate Details

From the KeyStore window, you can view specific details for the selected certificate in the Details Panel. The following certificate details will be displayed, similar to opening a standalone certificate:

- Format;
- Version;
- Serial Number;
- Validation date period;
- Public Key;
- Extensions;
- Signature Algorithm;

- Subject/issuer;
- Common Name (CN);
- Organization Unit (OU);
- Organization Name (O);
- Locality Name (L);
- State Name (ST);
- Country (C);
- Email (E);
- Trust Status (only for Certificate entries, not for the Certificate part of Key Pairs);

  ### Note

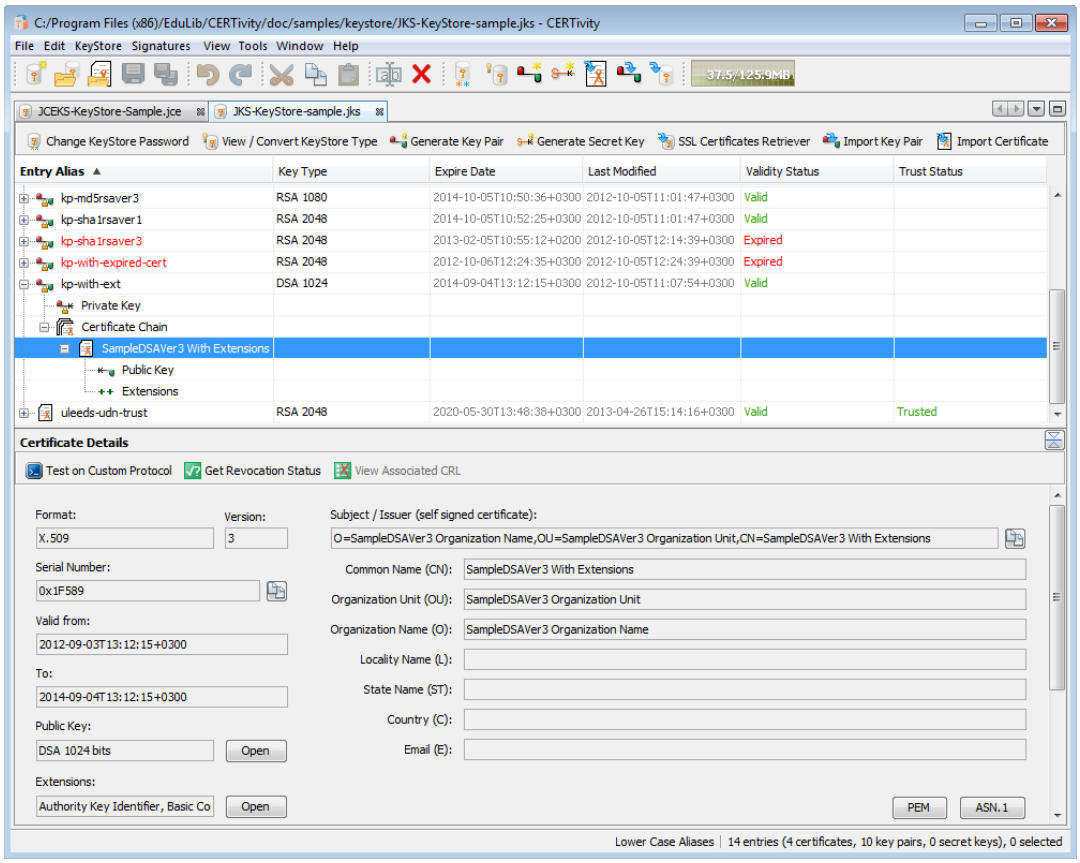
  This field is not visible when the Certificate details are displayed for user verification in import operations.

- MD5 Fingerprint;
- SHA1 FingerPrint.

The Certificate Details part of the KeyStore window contains also the following actions available:

- Test on Custom Protocol - which will open a new window for testing the certificate against a raw TCP/IP connection with the possibility to send text requests;
- Get Revocation Status - which will open a dialog to check the revocation status;
- View Associated CRL - which will open a new tab to view the entire Certificate Revocation List associated to the certificate in case one is available in the certificate extension;
- Open Public Key - which will position the TreeTable on the Public Key node under the Certificate entry and will populate the Details Panel with details about the Public Key (algorithm, key size, modulus, public exponent, ASN.1);
- PEM - which will open a new window containing the PEM representation of the certificate;
- ASN.1 - which will open a new window containing the ASN.1 representation of the certificate.
- Display more certificate fingerprints - which will expand the list of certificate fingerprints by adding to the list the fingerprints of a certificate in the following hashes: MD2, MD4, RIPEMD-128, RIPEMD-160, RIPEMD-256, SHA-224, SHA-256, SHA-384 and SHA-512.
- Display less certificate fingerprints - which will collapse the list of certificate fingerprints by removing from the list the fingerprints of a certificate in the following hashes: MD2, MD4, RIPEMD-128, RIPEMD-160, RIPEMD-256, SHA-224, SHA-256, SHA-384 and SHA-512.

A screenshot for Certificate's Details in the KeyStore window can be seen below:

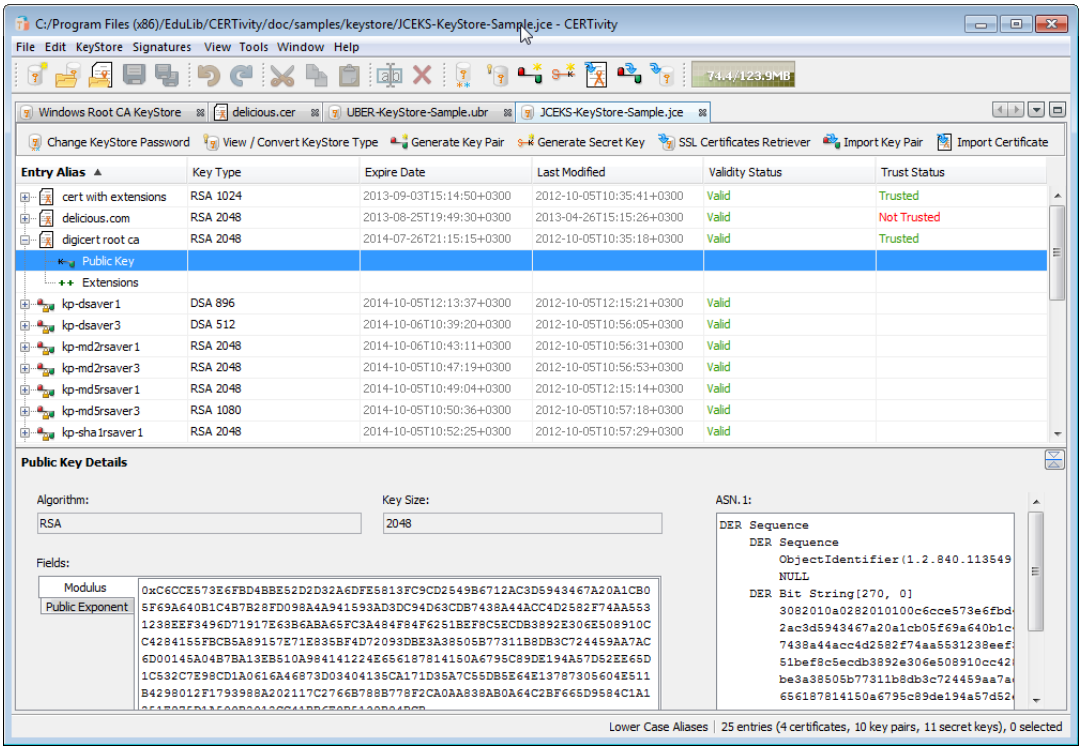


5.11 View Public Key Details

When selecting a Public Key entry alias in the KeyStore window, the Public Key Details are displayed at the bottom of the window. The Public Key Details that can be retrieved this way, are:

- Algorithm;
- Key size;
- ASN.1 representation;
- Modulus;
- Public Exponent;

A screenshot for a certificate's Public Key Details can be seen below:



5.12 View Certificate Extensions Details

Certificate extensions offer more information about the certificate by extending the original X.509 certificate standard information with additional identification information or information about the cryptographic capabilities and restrictions in usage of the certificate. Only V3 certificates can have extensions.

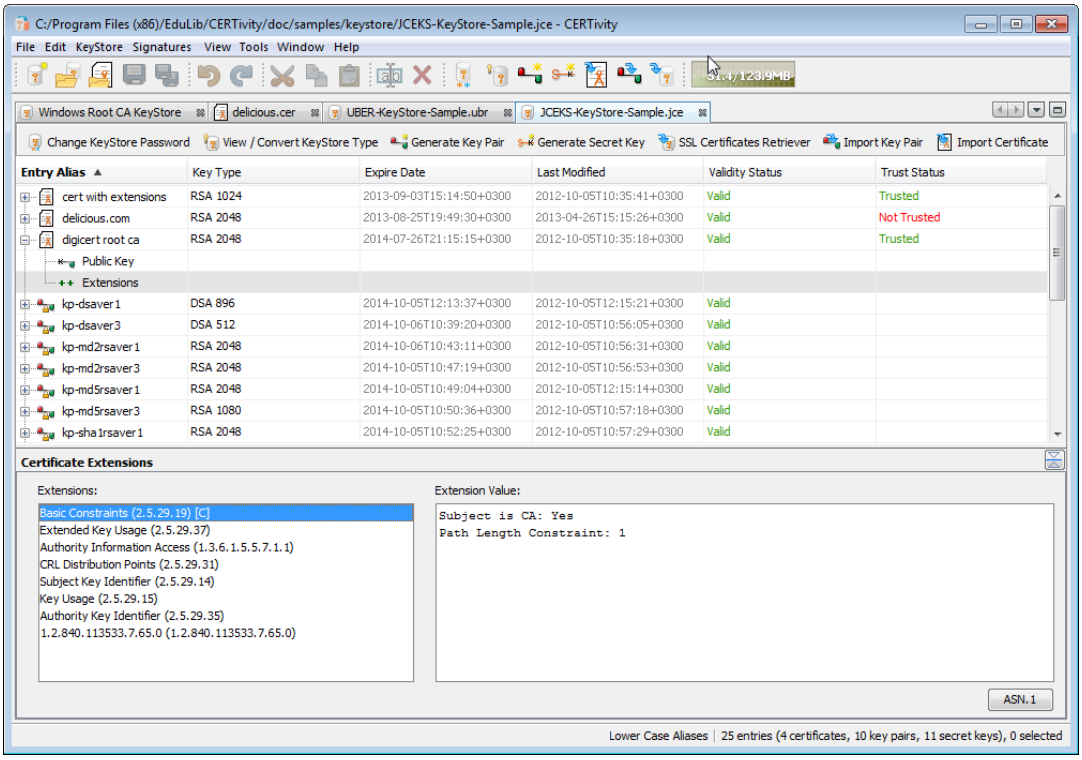
When selecting an Extensions entry alias in the KeyStore window, the certificate extensions details are displayed at the bottom of the window. The extensions details panel can also be obtained by clicking on the "Open" button next to the extensions text field in the certificate panel.

CERTivity allows viewing the extensions contained in a certificate and can display the content of the following extensions:

- Authority Information Access;
- Authority Key Identifier;
- Basic Constraints;
- CRL Distribution Points;
- Extended Key Usage;
- Issuer Alternative Name;
- Key Usage;
- Netscape Cert Type;
- Private Key Usage Period;
- Subject Alternative Name;

- Subject Information Access;
- Subject Key Identifier.

A screenshot for certificate extensions can be seen below:



The extensions that exist in a certificate are displayed in a list (left side of bottom panel) using their name and their Object Identifier (OID). For the ones which are not recognized, only the OID will be displayed.

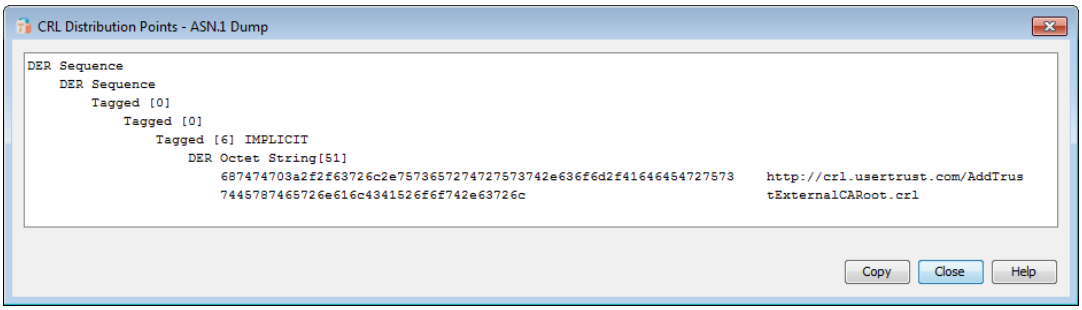
Also, extensions which are marked as critical can be identified by the symbol "[C]" at the end of the name.

The value of each extension is displayed in the right side of the bottom panel when selecting it from the list. The content is displayed in a text format, having indentation where needed (when some fields have other sub fields) for a better representation and to be easy to read.

5.12.1 View Certificate Extensions ASN.1 Representation

CERTivity allows displaying the ASN.1 representation for each extension (even for those which are not yet recognized). To see the ASN.1 representation click on the button "ASN.1" from the bottom right corner of the extensions panel. A new dialog will open displaying the ASN. 1 content.

This dialog can be seen in the screenshot below:



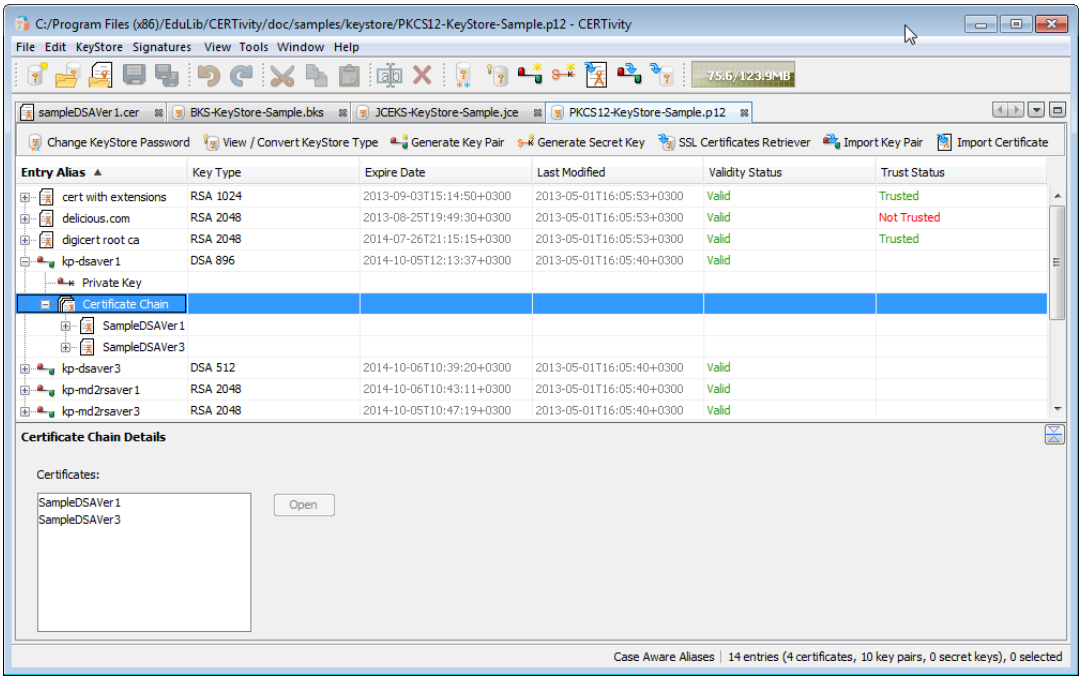
Note

When clicking the "ASN.1" button, the dialog will display the ASN.1 content of the selected extension from the extensions list (in the left of the extensions panel).

5.13 View Certificate Chain Details

In order to view certificate chain details, open the desired KeyStore (Menu **`File > Open > Open KeyStore`**) and click on the entry alias **`Certificate Chain`**.

In the bottom part of the window, you can see the list with all the certificates. More details for all the certificates from the list can be obtained by selecting one certificate and clicking on **Open** as well as selecting the certificate directly from the Tree view. It is also possible to select a certificate from the chain and copy it into the clipboard in order to paste it into another KeyStore or even in the current one.

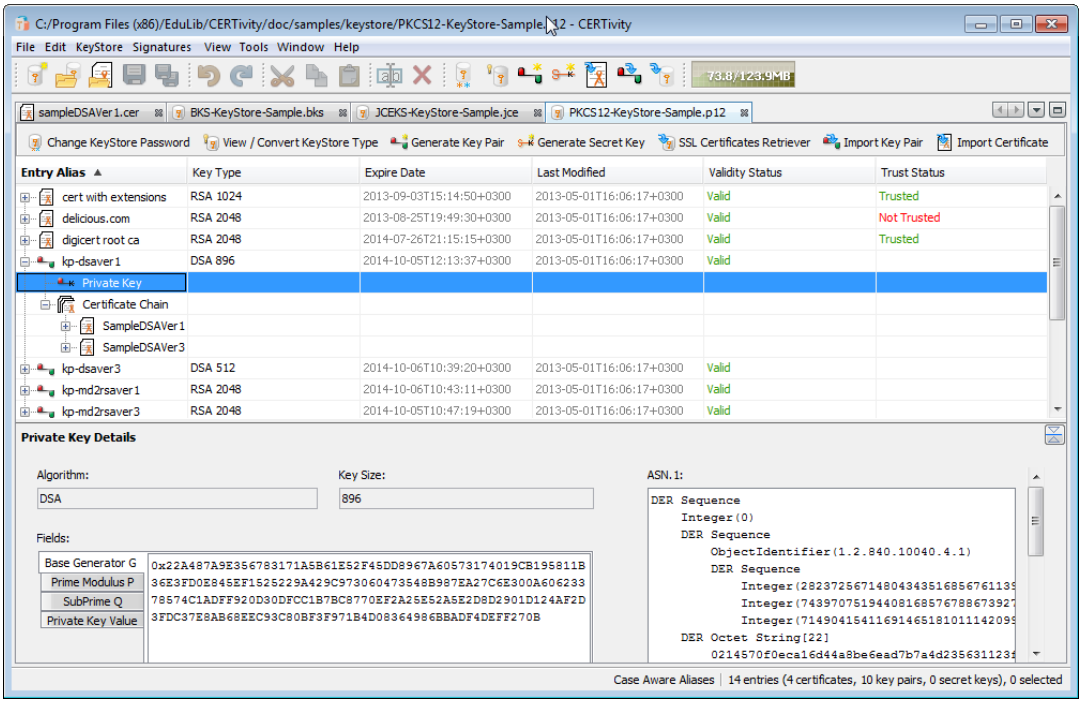


5.14 View Private Key Details

When selecting a Private Key entry alias in the KeyStore window, after introducing the correct password, the Private Key Details are displayed at the bottom of the window. The Private Key Details that can be retrieved this way, are:

- Algorithm;
- Key size;
- ASN.1 representation;
- Base Generator G;
- Prime Modulus P;
- SubPrime Q;
- Private Key value.

A screenshot for Private Key Details can be seen below:

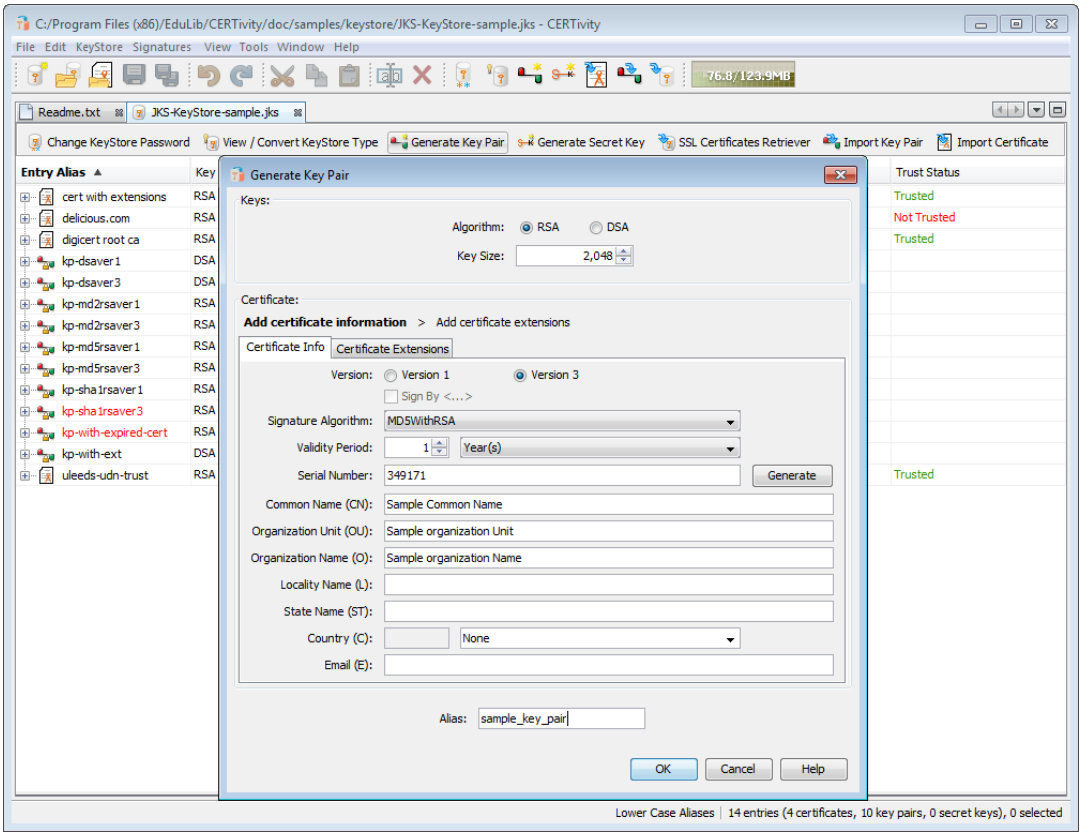


5.15 Generate Key Pair

In order to generate a Key Pair and add it into the current KeyStore, click on **`Generate Key Pair`**. A new window will be opened, containing:

- a section for Keys, where you have to select the desired algorithm and size for the future Key Pair;
- a section for the Certificate, where you have to complete the certificate related fields. This section contains 2 tabs:
  - A tab which allows adding information to the certificate such as the version of the certificate, if it will be signed by the selected CA Issuer, signature algorithm, validity period, serial number, common name, and other optional fields such as organization, organization unit, locality name, state name, country or email address;
  - A tab which allows adding extensions to the certificate for version 3 certificates. This tab is enabled when the "Version 3" option is selected in the first tab.
- an alias for the Key Pair entry in the current KeyStore.

A screenshot for generate key pair action can be seen below:



Depending on the algorithm selected the key size and the signature algorithm are different.

- For RSA the minimum and maximum key sizes are configurable. The minimum key size can be set from Menu **`Tools > Options > RSA Key Pair min size`**. The minimum key size allowed is `1024` bits. It can be set past this value to impose a higher minimum key size, to avoid generating keys under a certain size. The maximum key size is also configurable from Menu **`Tools > Options > RSA Key Pair max size`**. The reason for this is that for higher RSA key size values the processing time for creation, as well as for using that key for encrypting/decrypting will be too big which will be very unsuitable in production. The key-size has to be a multiple of 8 - this is also the spinner increment.

  If a value smaller than the minimum key size or larger than the maximum key size is provided, then a warning will be issued upon pressing `OK`.

  The default key size value that appears initially in the Generate Key Pair dialog can be set from Menu **`Tools > Options > RSA Key Pair default size`**.

  The signature algorithm for RSA can be one of the followings: `MD5WithRSA`, `MD2WithRSA` or `SHA1WithRSA`. The default signature algorithm is `MD5WithRSA`.

- For DSA the minimum key size is 512 bits and the maximum is 1024 bits. The key size must also be a multiple of 64 - this is ensured by the spinner. If a value out of the range `512-1024` is provided then a warning will be issued upon pressing `OK`.

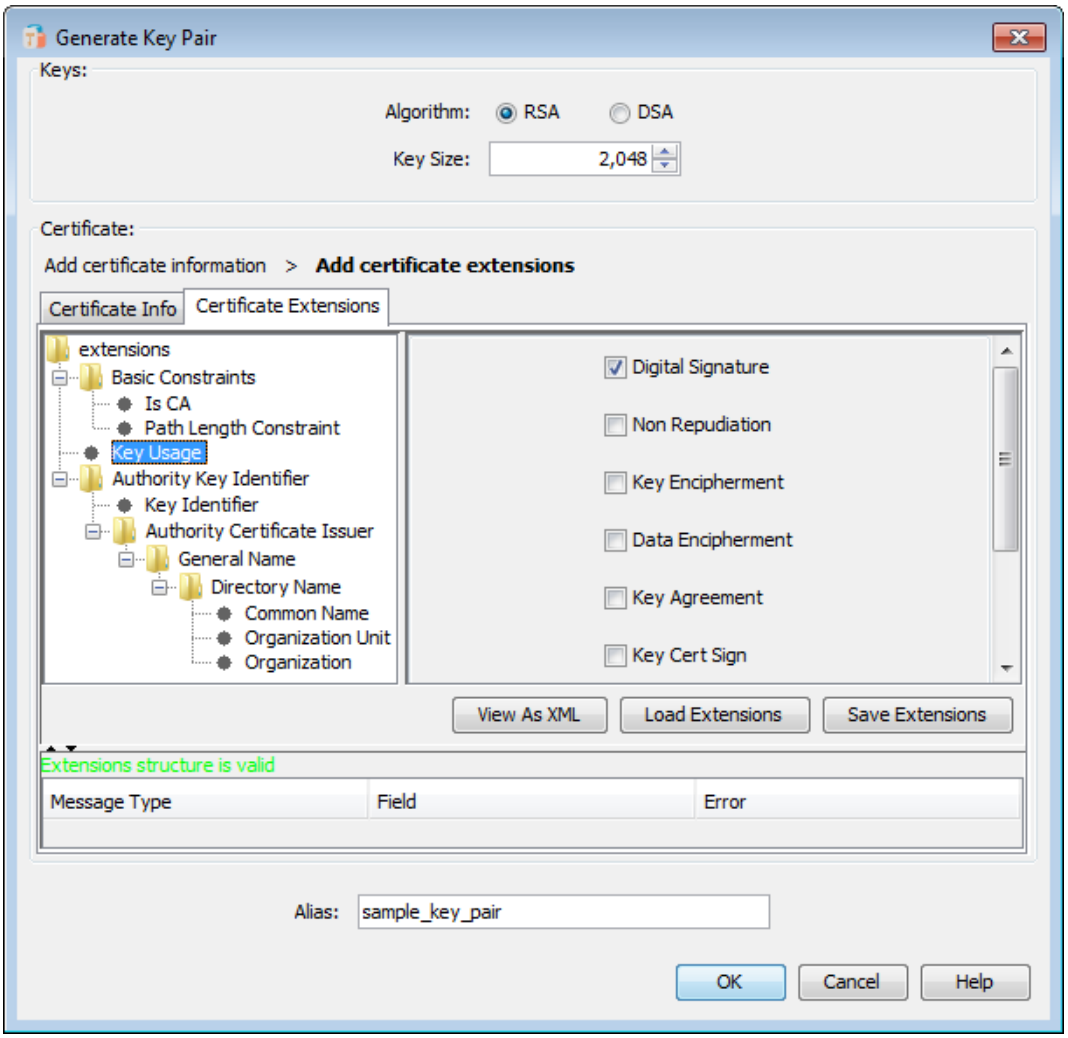
  The signature algorithm for DSA can only be `SHA1WithDSA`.

5.15.1 Manage Certificate Extensions

Certificate extensions are used to offer more information about the certificate by extending the original X.509 certificate standard information with additional identification information or information about the cryptographic capabilities and restrictions in usage of the certificate.

In order to be able to add extensions to the certificate, select the **`Certificate Extensions`** tab (the Version 3 option has to be selected in the **`Certificate Info`** tab. Version 1 certificates do not accept extensions, therefore the tab is disabled for version 1 certificates).

This tab offers the possibility to create a new set of extensions for the certificate, or load a set of extensions from a previously saved template. The extensions are represented in this dialog using a tree-like list, as it can be seen below:

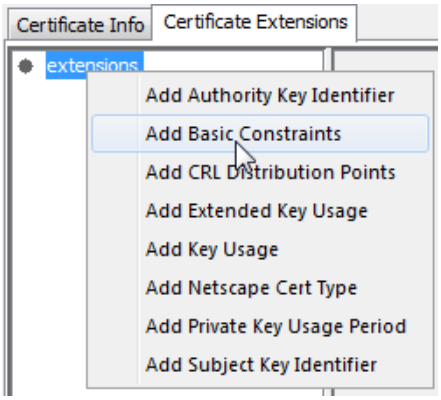


The extensions of the certificate together with their items can be added in the tree list from the left side (as it can be seen in the above picture), while on the right side, the values of each added extension item can be set after selecting the item for which the value should be set from the left side tree.

5.15.1.1 Creating an extension

To create an extension, the following steps must be performed:

- Right click on the extensions item from the extensions list;
- From the contextual menu that appears, select one of the available extensions;



CERTivity allows adding the following 8 extensions to a certificate:

- *Authority key identifier*:

  This extension is used for identifying the public key corresponding to the private key used to sign a certificate. It is used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover). The identification may be based on either the key identifier (the subject key identifier in the issuer's certificate) or on the issuer name and serial number. For more details please see RFC 5280 - 4.2.1.1 Authority Key Identifier [http://tools.ietf.org/html/rfc5280#section-4.2.1.1].

  CERTivity allows computing the value of the keyIdentifier field by deriving it from the public key (issuer public key) used to verify the certificate's signature. There are two available methods used by CERTivity for generating the key identifier:

  - 160-bit hash of the value of the bit string of the public key;
  - 64-bit hash of the value of the bit string of the public key.

- *Basic constraints*:

  This extension identifies whether the subject of the certificate is a CA (using the isCA field) and the maximum depth of valid certification paths that include this certificate (using the pathLengthConstraint field). For more details please see RFC 5280 - 4.2.1.10 Basic Constraints [http://tools.ietf.org/html/rfc5280#section-4.2.1.9].

- *CRL distribution points*:

  This extension identifies how CRL information is obtained. The cRLDistributionPoints extension can contain one or more DistributionPoint items. A Distribution Point consists of three fields, each of which is optional: `distributionPoint`, `reasons`, and `cRLIssuer`. Each of these fields is optional, but a DistributionPoint must not consist of only the reasons field; either distributionPoint or cRLIssuer must be present. For more details please see RFC 5280 - 4.2.1.14 CRL Distribution Points [http://tools.ietf.org/html/rfc5280#section-4.2.1.13].

- *Extended key usage*:

This extension indicates one or more purposes for which the certified public key may be used, in addition to or in place of the basic purposes indicated in the key usage extension. In general, this extension will appear only in end entity certificates. For more details please see RFC 5280 - 4.2.1.13 Extended Key Usage [http://tools.ietf.org/html/rfc5280#section-4.2.1.12].

- *Key usage*:

  This extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted. For example, when a RSA key should be used only to verify signatures on objects other than public key certificates and CRLs, the digitalSignature and/or nonRepudiation bits would be asserted. Likewise, when a RSA key should be used only for key management, the keyEncipherment bit would be asserted. For more details please see RFC 5280 - 4.2.1.3 Key Usage [http://tools.ietf.org/html/rfc5280#section-4.2.1.3].

- *Netscape Cert Type*:

  This extension is used for limiting the applications for a certificate. If the extension exists in a certificate, it will limit the uses of the certificate to those specified. The following values are available:

  - `SSL Client`: the certificate is selectable when a server requests a certificate;
  - `SSL Server:` the Netscape Communicator will talk to a server (otherwise it will complain that the certificate is invalid);
  - `S/MIME Client:` the certificate can be used for S/MIME signing and encryption;
  - `Object Signing`: the certificate can be used for signing objects such as Java applets and plugins;
  - `Reserved`: this bit is reserved for future use;
  - `SSL CA`: the certificate can be used for issuing certificates for SSL use;
  - `S/MIME CA`: the certificate can be used for issuing certificates for S/MIME use;
  - `Object Signing CA`: this certificate is certified for issuing certificates for Object Signing.

- *Private key usage period*:

  This extension allows the certificate issuer to specify a different validity period for the private key than the certificate. This extension is intended for use with digital signature keys. This extension consists of two optional components, notBefore (the date from when the certificate can be used) and notAfter (the date until the certificate can be used). For more details please see RFC 3280 - 4.2.1.4 Private Key Usage Period [http://tools.ietf.org/html/rfc3280#section-4.2.1.4].

- *Subject key identifier*:

  This extension provides a means of identifying certificates that contain a particular public key. For more details please see RFC 5280 - 4.2.1.2 Subject Key Identifier [http://tools.ietf.org/html/rfc5280#section-4.2.1.2].

CERTivity allows computing the value of the keyIdentifier field by deriving it from the public key (subject public key) used to verify the certificate's signature. There are two available methods used by CERTivity for generating the key identifier:

- 160-bit hash of the value of the bit string of the public key;
- 64-bit hash of the value of the bit string of the public key.

Each extension can be added only once. Therefore, after an extension is added, on the next right click on the `extensions` root node, on the pop-up menu only the remaining available extensions which have not already been added will be displayed.

- If the new added extension contains subitems (like the Authority Key Identifier extension for example which contains Key Identifier, Authority Certificate Issuer, and Authority Certificate Serial Number) these can be added in the same way, by right clicking on the list item of the new added extension. A popup menu containing the available subitems will be displayed. In the same way as for extensions, some of the subitems can be added only once, so they will appear only once on the first popup menu when right clicking on the item for which they have to be added.

  Some extensions contain mandatory subitems (for example the "Is CA" field, from Basic constraints extension), which will be added automatically when adding the new extension item. Also, after creating the extension item, the popup menu containing the available subitems will be automatically displayed as a form of autocompletion.

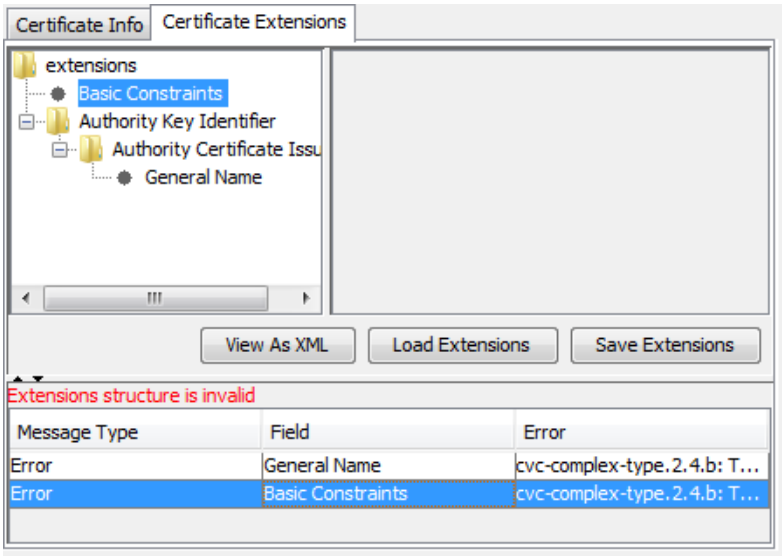
  If the selected extension or subitem does not contain any more mandatory or optional subitems, in the right side of the dialog, a panel will be displayed allowing the input of the value of the extension or subitem. This panel will contain either a text field (for extensions and items which allow various textual input) or a list of checkboxes for the ones which only have certain defined values (like the Key usage extension).
- For items which allow text input, to set the typed in value, press the **`Set value`** button.

After each extension or subitem added, as well as after each value set for an extension or subitem, the structure of the extension and the values are verified to ensure that they are valid according to the RFC 3280 [http://tools.ietf.org/html/rfc3280] and RFC 5280 [http://tools.ietf.org/html/rfc5280] standards. For the extensions to be able to be created, their structure and value has to be valid according to this standard. Otherwise, when trying to create the certificate, a message will be displayed informing that the structure or the values of the extensions are not valid, and asking if the certificate should be created without adding the extensions or to return to the certificate generation dialog for correcting errors.

Note

The validity state of the structure and content of the extension can be seen at any time at the bottom of the certificate structure list. For a valid structure, a green note will be displaying the message "Extensions structure is valid".

If the structure is not valid, due to missing mandatory subitems or invalid values set, the note will be displaying a red warning containing the message "Extensions structure is invalid". Also, under this note a table will become visible containing the additional information about each item which is invalid. The table contains the type of message (Error or Warning), the name of the extension item or subitem which is invalid, and an additional message containing the details of the error, as can be seen below:



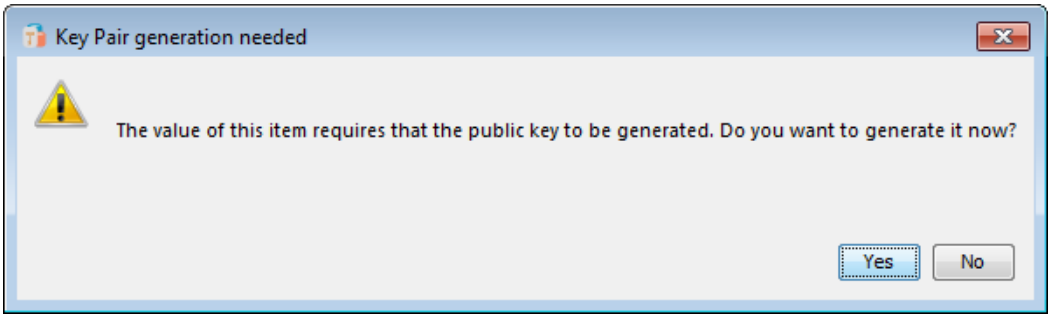
Selecting a row in the table will trigger the selection of the extension item or subitem to which the table row refers to. If the selection of the row is done having the pointer over the last column ("Error" column) a popup dialog will also be displayed showing the full details of the error.

Field value auto completing

For some extension fields, the value can be obtained from the information from the certificate or which is to be set to the certificate in case of generating a new Key Pair (for example from the serial number, or the Issuer/Subject name), or by computing hash values on the public key from the certificate or the one which is created when generating a new Key Pair.

- For Authority Key Identifier extension:
  - The value of the Key Identifier field is computed using one of the two methods (mentioned above) for generating the hash over the issuer public key.

    For generating a new Key Pair (containing a new self-signed certificate), the private - public Key Pair is usually generated when pressing the OK button. But, if the Key Identifier item is added for the Authority Key Identifier, a message will be displayed requesting to generate the public key earlier, so that its hash can be generated and used as a value for this field.



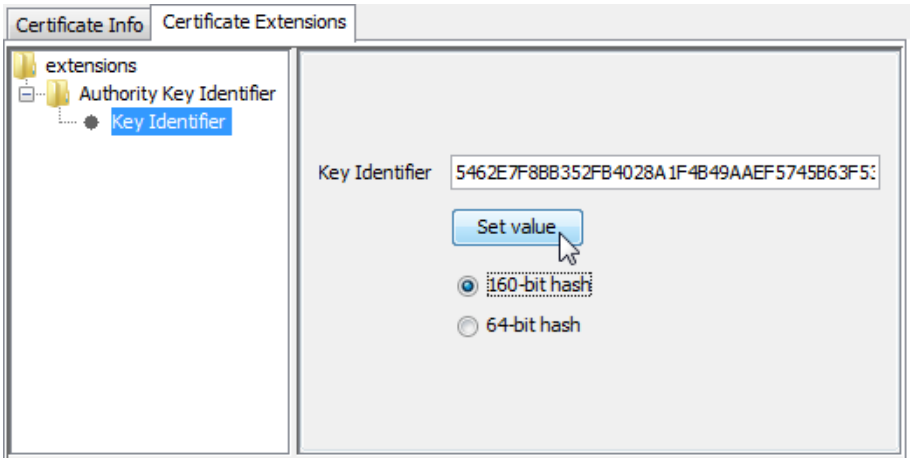
    If this request is denied, the Authority Key Identifier can still be added, but the value will have to be set by the user.

Note

This message will not be displayed when signing CSR files, because in that case both the public key of the issuer certificate and the public key of the resulting CA reply are generated at the time of adding extensions to the CA reply.

By default when generating the public key and computing the hash over it, the hash will be 160-bit. To use the 64-bit hash, switch to the 64-bit radio button. The value from the text field will be replaced with the 64-bit value. Also, to use back the 160-bit value, select the 160-bit radio button.

The radio buttons can be seen in the screenshot below:



- The Directory Name of the Authority Certificate Issuer, in the case in which the Authority Certificate Issuer should be represented using a Directory Name, can be autocompleted at generation time using the name values of the certificate Issuer DN (which for self-signed certificates is the same with the Subject DN), if these fields have been completed in the `Certificate Info` tab. Only the name items for which there is a value added in the previous tab will be added. The rest can be added manually by right-clicking on the `Directory Name` node, and adding a name component from the remaining items in the popup menu.
- The Authority Certificate Serial Number extension item can have it's value taken from the `Serial Number` field in the `Certificate Info` tab. If the serial number was not set, when creating the Authority Certificate Serial Number a message will be displayed asking to generate and set the Serial Number and also use that value for the Authority Certificate Serial Number. If this request is denied, then the value of the Authority Certificate Serial Number must be set manually.

- For CRL Distribution Points extension:

  For this extension, the same mechanism for autocompleting values applies where Directory Name items are used, like the one mentioned for the Authority Key Identifier extension. The names are taken from the Issuer DN if they were added before creating the Directory Name item.

- For Subject Key Identifier extension:

The value of this extension can be generated automatically in the same way the Key Identifier item from the Authority Key Identifier extension can be generated (presented above). The only differences are that for this extension the hash is computed over the public key of the user certificate, not the public key of the issuer certificate.

5.15.1.2 Save extensions template

After creating one or more extensions, the structure of the extensions together with their values can be saved to a file as a template, so that it can be reused for other certificate generation. The template will be represented by an XML document.

To save an extensions template the following actions must be performed:

- Click on the **`Save Extensions`** button;
- If the extensions structure is not valid or the items contain invalid values, a warning message will be displayed asking if the template should be saved anyway although it is invalid. If the "No" button is pressed, it will return to the panel for adding extensions.

- In the file chooser dialog that appears, type the name for the template. It will be saved in an XML document file.

5.15.1.3 Load extensions template

Extension structure templates previously created together with their values can also be loaded back from a file to be used for creating the same extensions for more certificates.

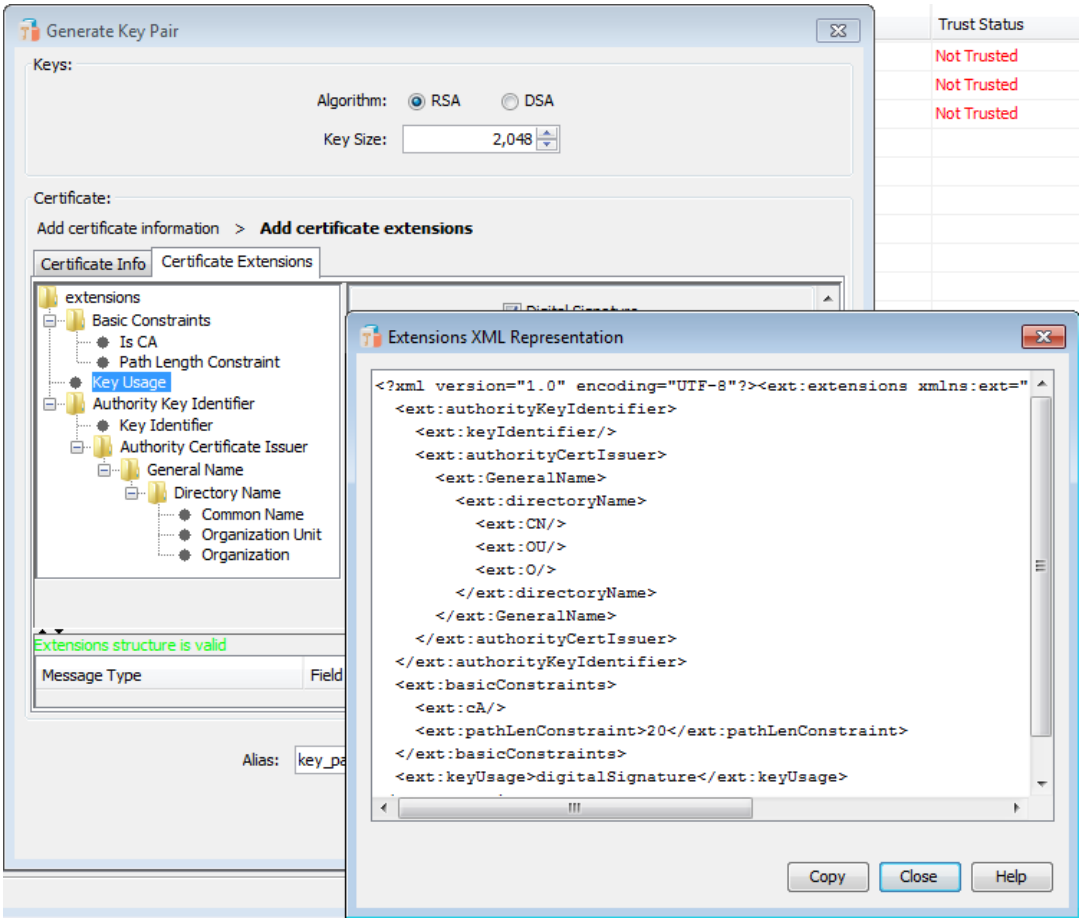
To load an extension template the following actions must be performed:

- Click on **`Load Extensions`** button;
- From the file chooser dialog that appears, select the template file to be loaded. The loaded extension structure will appear in the list on the left of the panel and will be validated.

5.15.1.4 View As XML

The extension structure can also be viewed in XML format. This XML format is the one in which the extension structure and values will be saved in the template file if this option is chosen.

The structure can be viewed in XML format at any time (either if the structure is valid or invalid), by clicking on the **`View As XML`** button. A dialog will open like the one below:



The XML structure can be copied to clipboard with a simple click on the **Copy** button, or by selecting the entire text and using the copy shortcut (CTRL-C).

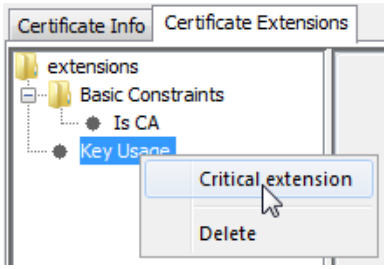
The XML structure follows the tree like structure of the extensions list. The direct XML child nodes of the **extensions** root node represent the extensions. The items of each extension (when they are present) are represented as child nodes of each extension node. The name of the XML nodes are similar to the names of the extensions and their subitems in the ASN.1 representation (which can be seen for each extension in RFC 5280 [http://tools.ietf.org/html/rfc5280]).

5.15.1.5 Mark extensions as critical

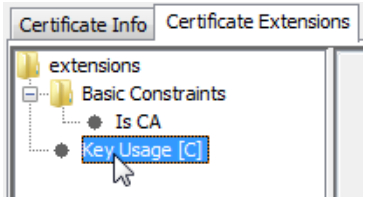
Extensions can be marked as critical or non critical. When an extension is marked as critical, it indicates that its value contains information of such importance that an application cannot ignore it. If a particular certificate-using application cannot process a critical extension, the application should reject the certificate.

CERTivity allows marking the extensions as critical. By default, after the extension item is created in the extensions list, the extension is considered to be non-critical.

To make an extension critical after its structure was created in the list, right click on the extension item, and from the popup menu that appears select **Critical extension** as it can be seen below:



After the extension is marked as critical, next to the extension item in the tree list the "[C]" symbol will be added, informing that the extension was marked as critical, as it can be seen below:



Also, in the right click menu for the extension item, the "Critical extension" menu will be checked.

To unmark an extension that has been marked as critical, right click on the extension item again, and from the popup menu that appears select again **`Critical extension`** (which now is checked) to uncheck it.

Note

Although all extensions can be marked as critical, the "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" (RFC 3280 and RFC 5280) recommends that some extensions should not be marked as critical. If an extension is marked as critical, the application which uses the certificate has to be able to process the extension, or otherwise to reject the certificate.

The extension creation may fail in the situation in which some of the extensions which should not be marked critical are marked so. For example, if the Authority Key Identifier extension is marked as critical (although the above mentioned profile does not recommend to be marked as so), if for the Authority Certificate Issuer an Uniform Resource Identifier is set, if the value of this URI does not have a valid form, the creation of the extension will fail. The same situation applies to the CRL Distribution Points extension.

The other extensions supported by CERTivity which the profile does not recommend marking as critical are Private Key Usage Period and Subject Key Identifier.

5.15.1.6 Delete extensions

The extensions and their subitems can also be deleted from the extensions structure. To delete an individual extension or subitem the following steps can be performed:

- Select the extension or subitem to be deleted;
- Right click on it, and from the popup menu, select **`Delete`**.

Or:

- Select the extension or subitem to be deleted;
- Press **"Delete"** key.

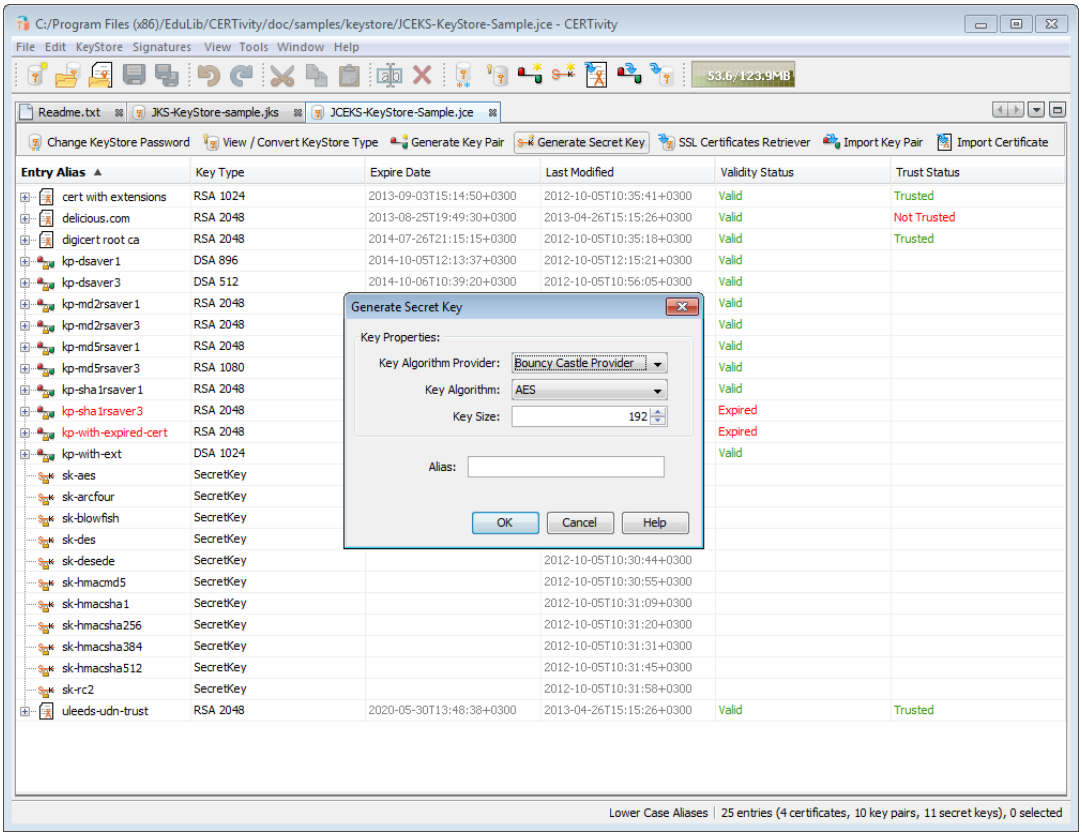
Also, to delete the entire structure, right click on the extensions item node, and select **Delete all**.

5.16 Generate Secret Key

In order to generate a Secret Key and add it into the current KeyStore, click on **Generate Secret Key**. In the new window, the user has the option to select from a wide range of key algorithms and sizes. The algorithms are defined for 2 providers: for the Bouncy Castle Provider and for the Sun JCE Provider (if it exists on the system where CERTivity is running), allowing the user to select only the supported key sizes for each algorithm depending on the algorithm type and provider. In case the Sun JCE Provider is not available, the Default provider will be used which means that all the Secret Key algorithms (that CERTivity supports) will be displayed with the key sizes starting from 1 for each algorithm. For this case, if the algorithm or the key size is not supported by the Default provider, an error will be displayed.

To generate a Secret Key, the user has to select a Provider, then to select an algorithm, then a key size, and finally to enter an alias for the Secret Key which will be generated.

A screenshot for generate secret key action can be seen below:



Note

JKS and PKCS#12 KeyStore types do not support storing Secret Keys. This is a limitation of the standards, not of the CERTivity application.

The key algorithms are dictating the JCE provider and the key sizes supported. These are depicted in the following table.

Table 5.2. Size and Provider for Secret Keys

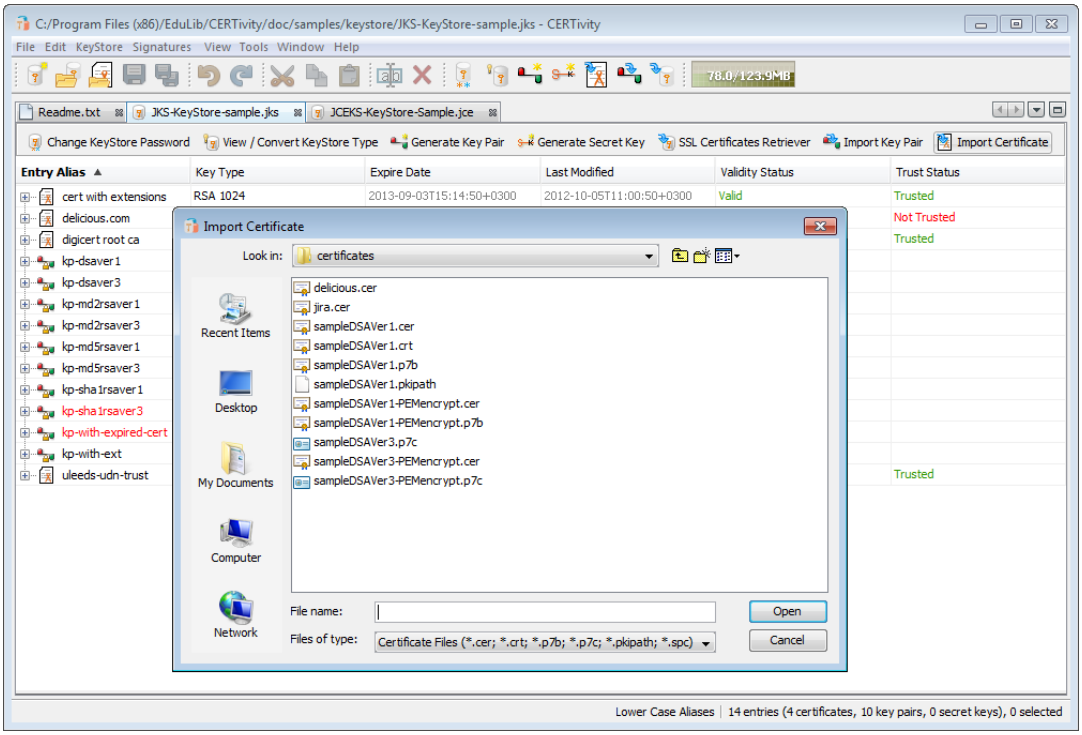
Key Algorithm	Key Size	Provider
AES	1 - 256	Bouncy Castle
	128 - 256, multiple of 64	Sun JCE
AESWrap	1 - 256	Bouncy Castle
ARCFOUR	40 - 1024	Sun JCE
Blowfish	1 - 448	Bouncy Castle
	32 - 448, multiple of 8	Sun JCE
Camellia	128 - 256, multiple of 64	Bouncy Castle
Cast5	1 - 128	Bouncy Castle
Cast6	1 - 256	Bouncy Castle
DES	64	Bouncy Castle
	56	Sun JCE
DESede	128, 192	Bouncy Castle
	112, 168	Sun JCE
DESedeWrap	128, 192	Bouncy Castle
GOST28147	256	Bouncy Castle
Grainv1	80	Bouncy Castle
Grain128	128	Bouncy Castle
HC128	128	Bouncy Castle
HC256	256	Bouncy Castle
Noekeon	128	Bouncy Castle
RC2	1 - 1024	Bouncy Castle
	40 - 1024	Sun JCE
RC4	40 - 2048	Bouncy Castle
RC5	1 - 128	Bouncy Castle
RC5-64	1 - 256	Bouncy Castle
RC6	1 - 256	Bouncy Castle
Rijndael	1 - 256	Bouncy Castle
Salsa20	128, 256	Bouncy Castle
SEED	128	Bouncy Castle
Serpent	128 - 256, multiple of 64	Bouncy Castle
Skipjack	1 - 128	Bouncy Castle

Key Algorithm	Key Size	Provider
TEA	128	Bouncy Castle
Twofish	128 - 256, multiple of 64	Bouncy Castle
VMPC	128, 6144	Bouncy Castle
VMPC-KSA3	128, 6144	Bouncy Castle
XTEA	128	Bouncy Castle
HmacMD2	1 -	Bouncy Castle
HmacMD4	1 -	Bouncy Castle
HmacMD5	1 -	Bouncy Castle
	1 -	Sun JCE
HmacRIPEMD128	1 -	Bouncy Castle
HmacRIPEMD160	1 -	Bouncy Castle
HmacSHA1	1 -	Bouncy Castle
	1 -	Sun JCE
HmacSHA224	1 -	Bouncy Castle
HmacSHA256	1 -	Bouncy Castle
	40 -	Sun JCE
HmacSHA384	1 -	Bouncy Castle
	40 -	Sun JCE
HmacSHA512	1 -	Bouncy Castle
	40 -	Sun JCE
HmacTIGER	1 -	Bouncy Castle

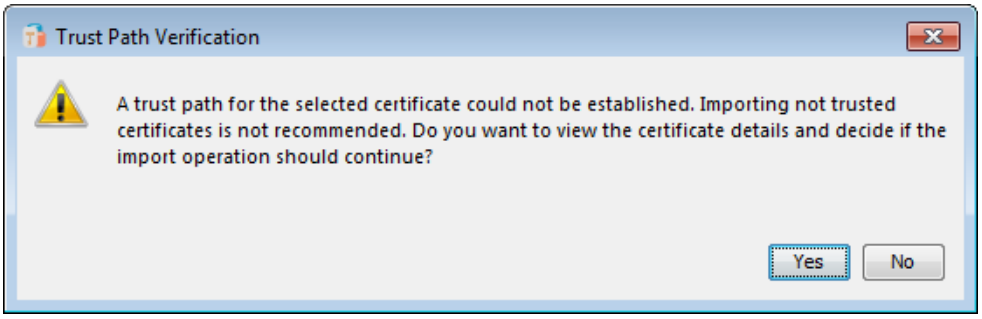
5.17 Import Trusted Certificate

In order to import a trusted certificate, click on **`Import certificate`** in the KeyStore window. Then, the desired certificate can be selected and an entry alias can be associated with it.

A screenshot for import trusted certificate action can be seen below:



If for the certificate which was selected to be imported a Trust Path can not be established, a warning message will be displayed informing that trust could not be established and asking if the user would like to view the certificate and decide if the import operation should be done or not. The warning message can be seen below:

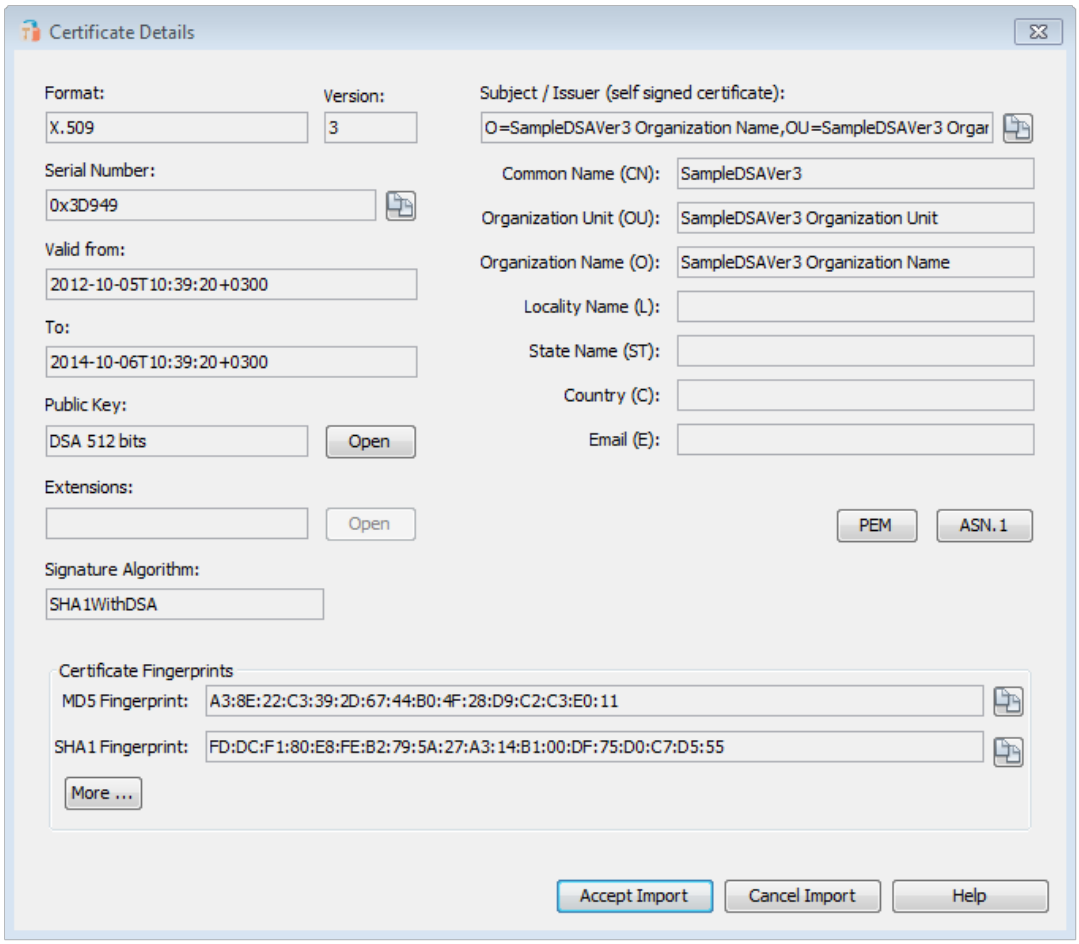


If "No" is selected or the dialog is closed, the import operation will be aborted.

If "Yes" is selected, the certificate will be displayed in a dialog with the options "Accept Import" to continue the import, or "Cancel Import" to abort the operation which can be seen in Certificate Trust Established by User.

5.17.1 Certificate Trust Established by User

In the situations when trust can not be established for a certificate, if the user decides to view the certificate to establish if it can be trusted or not, the following dialog will be displayed containing the details of the certificate:



If the user does not trust the certificate after viewing its details, he can abort the import operation by pressing "Cancel Import", or by closing the dialog.

If the user decides that the certificate can be trusted after viewing its details, he can continue the import operation by pressing "Accept Import".

5.18 Import Key Pair

In order to import a Key Pair, click on **`Import Key Pair`** in the KeyStore window. Three types of Key Pairs can be imported:

- PKCS#12 - which defines a file format commonly used to store private keys with accompanying public key certificates, protected with a password-based symmetric key;
- PKCS#8 - which is used to carry private certificate key pairs (encrypted or unencrypted);
- OpenSSL - with which a public key doesn't need to be generated separately because the private key contains the public key information as well;

The "Input File(s) Information" area contains the following fields, which are enabled depending on the previously selected Key Pair type:

- the decryption password;
- the Key Pair File;

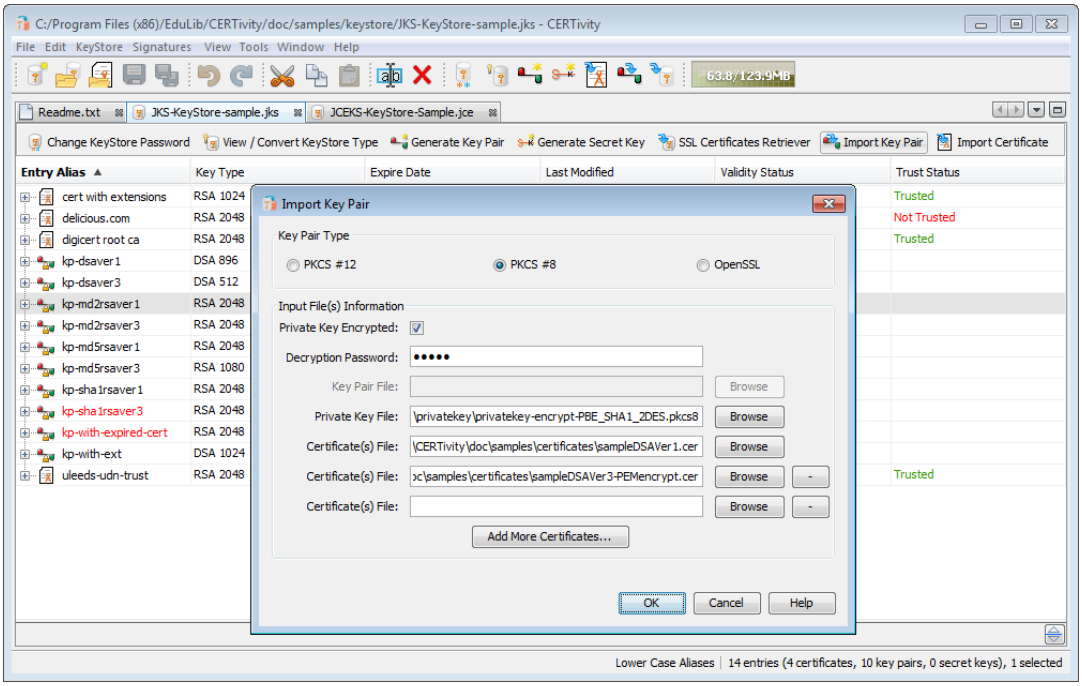
- the Private Key File;
- one or more Certificate(s) File(s);
- an "**`Add More Certificates...`**" button which creates more Certificate(s) file input fields.

The files for each Certificate(s) field can be selected using the **`Browse`** button. Also, the Certificate input fields which are not needed (except for the first one) can be removed using the "-" (minus) button next to the **`Browse`** button of each Certificate input field. The Certificate(s) input fields which are empty will be ignored, but at least one of the Certificate input fields must contain a valid value.

The Certificate input fields can accept also files which contain more certificates (such as Certificate Chain files - ".p7b", ".p7c" files). The order in which the certificate files are provided in the input fields is not important, as the certificates will be ordered (with the user certificate first and the signer or root certificate last) using a sorting algorithm based on subject - issuer matching.

When pressing OK, the resulting certificate chain will be validated (using a validation algorithm based on signature verification and subject - issuer matching), and the importing operation will continue only if the certificate chain is valid. Otherwise the user will be informed that the resulting chain is not valid and the Import Key Pair dialog will reappear to allow modifications.

A screenshot for Import Key Pair action (using a private key and Certificate(s) files) can be seen below:



5.19 SSL Certificates Retriever

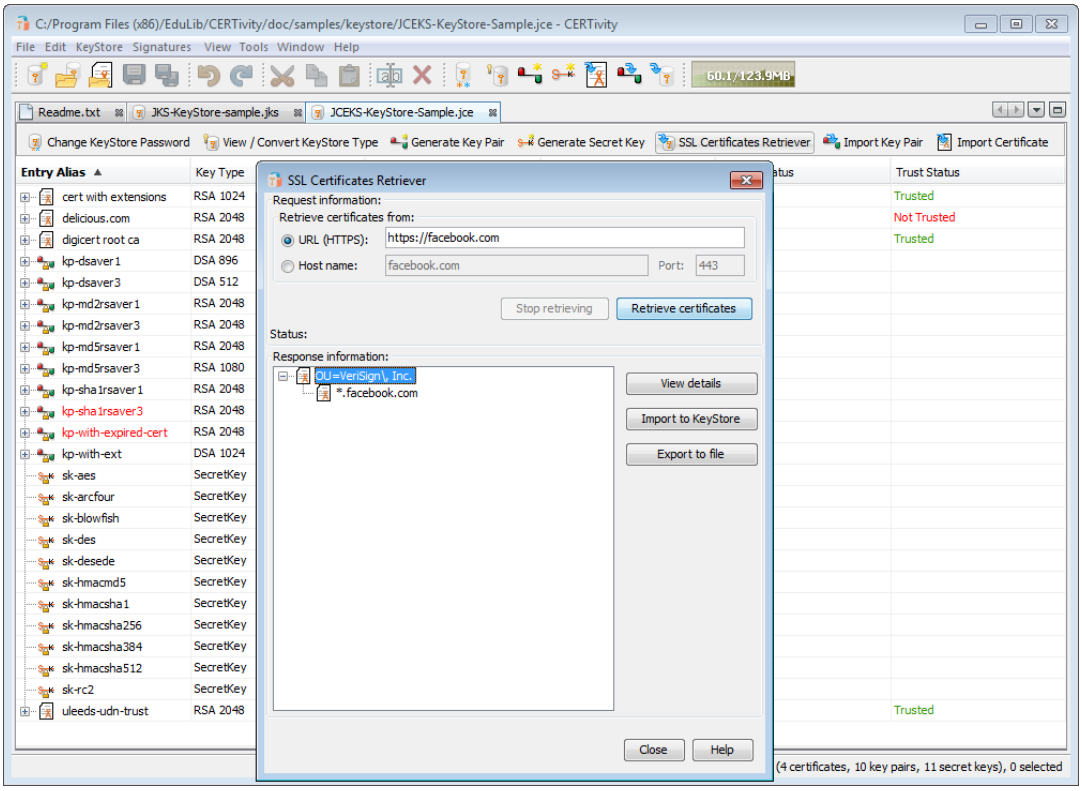
In order to import a trusted certificate, click on **`SSL Certificates Retriever`** in the KeyStore window. For retrieving certificates,in the host/port fields you must specify the server from which the certificates will be retrieved. If you have a HTTPS URL and you want to retrieve

the certificates then you need to enter the host without any path and the port separately. By default the HTTPS port is 443.

The server response is available in the "Response information" area. For the retrieved certificates, the available actions are:

- View details;
- Import to KeyStore;
- Export to file.

A screenshot for SSL Certificates Retriever action can be seen below:

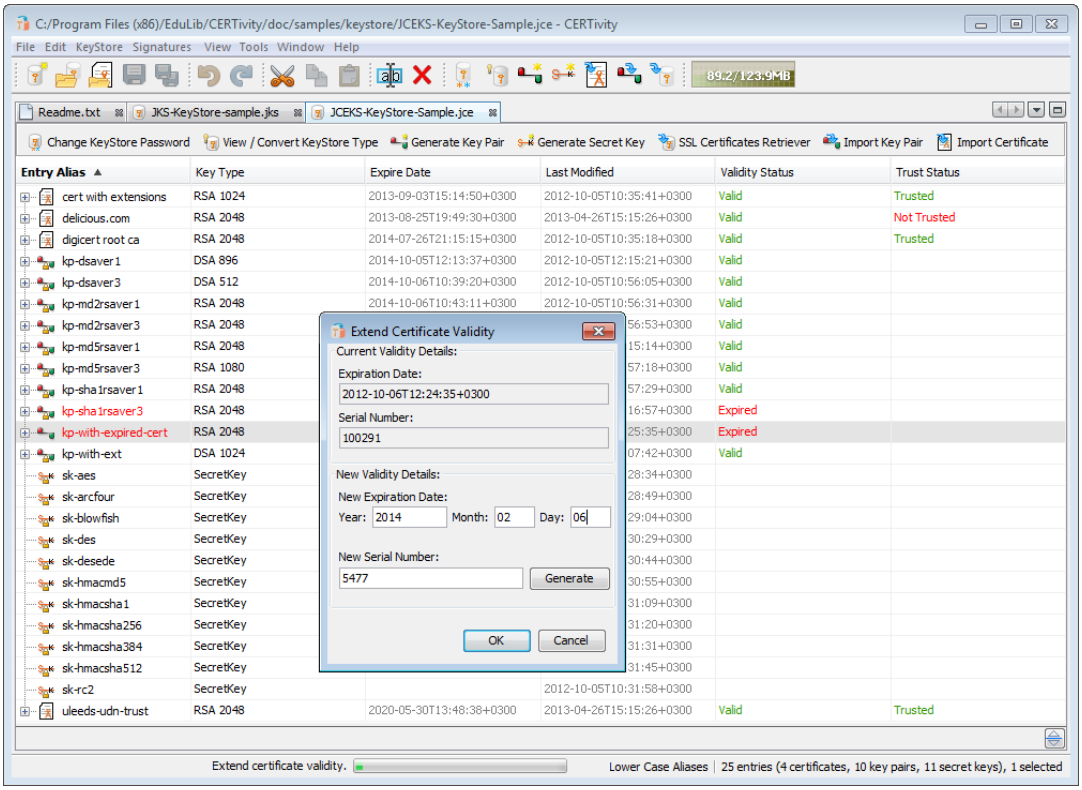


When importing a selected certificate into the active KeyStore, the certificate trust will be verified in the same way it is verified when importing a trusted certificate into the active KeyStore. If a Trust Path can not be established using the provided TrustStores and the Trust Path validation options (which can be set from `Tools > Options > Trust Path Options`), a message will be displayed informing about that and asking if the certificate should be displayed for user verification. If "No" is selected, or the dialog is closed, the import operation is aborted. If "Yes" is selected, the certificate details will be displayed, and the user will have the option to continue the import operation (by selecting the "Accept Import" button) or to abort it (by selecting the "Cancel Import" button).

5.20 Extend Validity

In an opened KeyStore window, select the key pair entry and click on the right mouse button. From the popup menu select **`Extend Validity`**. A new expiration date must be selected. This functionality makes sense only for the Key Pairs containing only a self signed certificate.

A screenshot for extend certificate's validity action can be seen below:



As it can be seen in the screenshot, the user can also set a new serial number for the new resulting certificate. The serial number can be generated randomly by using the `Generate` button, or the user can set a certain serial number by typing it in the text box. The new serial number has to be a pozitive integer value. Otherwise, a red message will be displayed under the text box informing that the new serial number is invalid and the operation will not continue.

5.21 Regenerate Key Pair

CERTivity offers the possibility to regenerate a Key Pair, more exactly to generate a new Key Pair using part of the information from an existing certificate and key information from a Key Pair.

In order to regenerate a Key Pair, in an opened KeyStore window, select a Key Pair entry and invoke the contextual menu (usually by performing a right click on the entry). From the menu that appears, select **`Regenerate Key Pair`**. A dialog similar to the one from `Generate Key Pair` action will appear which will have some fields pre-filled with the information obtained from the certificate of the selected Key Pair. The information which is taken from the certificate is:

- Key Algorithm (of the public key);
- Key Size (of the public key);
- Certificate Version;
- Certificate Signature Algorithm;
- Certificate Subject/Issuer distinguished name components (Common Name (CN), Organization Unit (OU), Organization Name (O), Locality Name (L), State Name (ST), Country (C), Email (E)).

If some of the information mentioned above is missing, or can not be extracted from the certificate, the defaults will be used. For example, if the key algorithm type can not be parsed, the default selection will be RSA, or if the Certificate Signature Algorithm can not be obtained (or is of an unsupported type), the default value will be used (MD5WithRSA, for RSA keys, and SHA1WithDSA for DSA keys). Also, the fields representing the subject distinguished name components will be filled only if these components are present in the certificate and contain a non empty value.

The Serial Number field will not be pre-filled with the value from the certificate from the selected Key Pair, because each certificate must have a unique serial number. Thus, this has to be provided (or generated using the Generate button) by the user.

Also, the new certificate will be valid for the period mentioned in the Validity Period field (which by default is 1 year) and the validity period will start from the moment of generation, regardless of the validity of the initial certificate from which the information is obtained.

The Regenerate Key Pair dialog, allows adding extensions to the certificate as well, but these are not pre-filled with the ones from the initial certificate.

A new alias name for the new Key Pair that will be generated is required. The new Key Pair will not replace the initial one in the KeyStore.

After filling all the required information, press OK. You will be prompted to enter a password for the new Key pair, and the Key Pair will be generated.

5.22 Generate CSR File

In order to generate a Certificate Signing Request file for a Key Pair entry, in an opened KeyStore window, select that Key Pair entry and invoke the contextual menu (usually by clicking the right mouse button). In this contextual menu select **Generate CSR File**. For generating a CSR file, you have to specify:

- the name of the generated file;
- the algorithm to use for signing the request; this is automatically adjusted based on the key algorithm used in the Key Pair. For DSA keys the possible selections are: SHA1withDSA, SHA1WithDSA, SHA224WithDSA, SHA256WithDSA, SHA384WithDSA and SHA512WithDSA, while for RSA keys the possible selections are: MD2WithRSA, MD5WithRSA, SHA1WithRSA, SHA1WithRSAandMGF1, SHA224WithRSA, SHA224WithRSAandMGF1, SHA256WithRSA, SHA256WithRSAandMGF1, SHA384WithRSA, SHA384WithRSAandMGF1, SHA512WithRSA, SHA512WithRSAandMGF1, RIPEMD128WITHRSA, RIPEMD160WITHRSA andRIPEMD256WITHRSA.
- the challenge - the challenge-password attribute, which specifies a password by which the entity may later request certificate revocation.

The supported CSR formats are PKCS #10 and SPKAC (Signed Public Key and Challenge). These are selectable from the file chooser filter list.

5.23 Import CA Reply

In an opened KeyStore window, select a key pair entry and invoke the contextual menu (usually by clicking the right mouse button). In this menu select **Import CA Reply** . The CA Reply can be chosen from a file chooser.

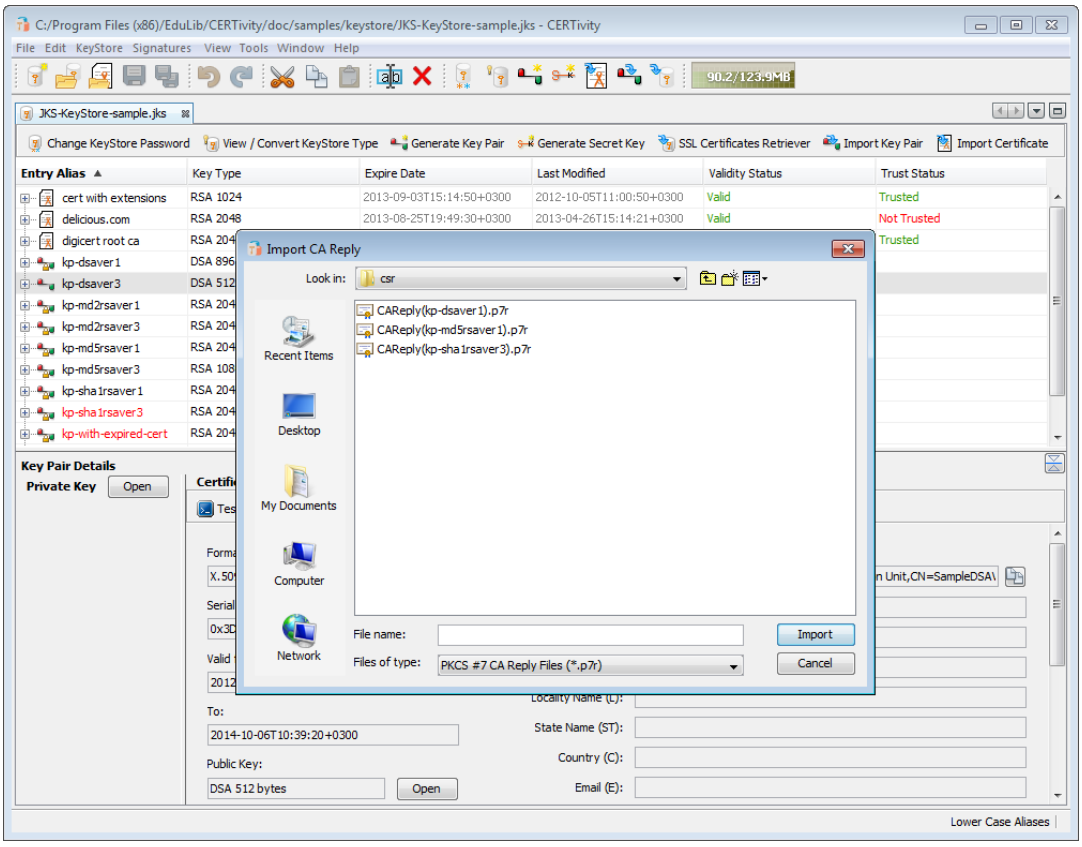
Importing the CA Reply will replace your self-signed certificate with a certificate chain. This chain will be either the one returned by the CA in response to your request (if the CA reply is a chain) or one constructed (if the CA reply is a single certificate) by establishing a Trust Path using the CA Reply certificate and the trusted certificates available in the given TrustStores (which can be set from `Tools > Options > Trust Path Options`). It can also be a single certificate which is signed by a signing authority, if the Trust Path could not be established but the user accepts the import.

The process of importing a CA Reply is more detailed and implies a series of validations and steps for establishing trust or constructing the chain from the CA Reply if it is a single certificate. There are two types of validations performed: one type which is critical and stops the validation process if it fails (if the CA Reply contains a chain and the chain is not valid, or other errors occur durring the validation and import process), and one type which will inform the user that the CA Reply chain is not trusted or that a Trust Path could not be established for the given CA Reply (if it is a single certificate) and lets the user choose if the import process should continue or not by displaying the details of the top certificate of the CA Reply.

The chain of certificates representing the received CA Reply is considered to be valid if the signature of each certificate is verified by the public key of the certificate on the next higher level in the chain. Also, for the import process to be able to be performed, it is necessary that the chain of the CA Reply to correspond to the entry for which the import is being made. This means that the public key of the first certificate in the chain to be equal to the public key of the self-signed certificate which it should replace in the Key Pair selected for performing the import.

A CA Reply (either containing a certificate chain or a single certificate) has to be trusted. The received chain is considered to be trusted if the top certificate is trusted, which means, to be present in the TrustStores set by the user. Also, a CA Reply containing a single certificate is considered to be trusted if a Trust Path can be established for it using the trusted certificates in the TrustStores set by the user.

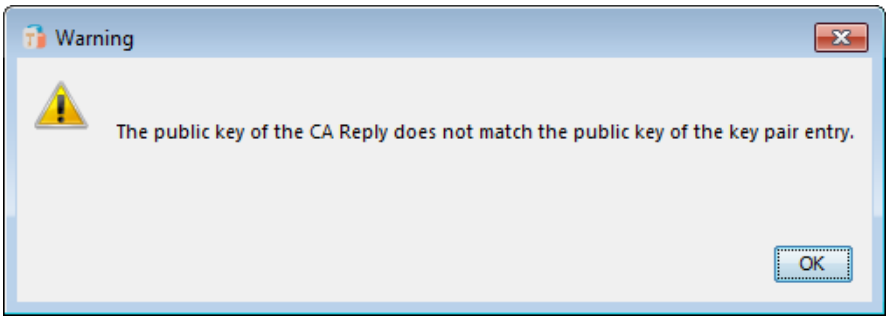
A screenshot for importing a CA Reply is depicted below:



The steps and validations for importing a CA Reply and the order in which they are performed in CERTivity® are as following:

- First, if the CA Reply contains a chain, the chain is verified to not contain any loops. If any loop is detected you will be informed by a warning message that the CA Reply contains a loop and you will be asked to decide if the import operation should continue or not. If this loop is not a mutual trust loop, we advise you not to import the CA Reply;
- Then, the CA Reply is verified to belong to the entry for which it should be imported. This means that the public key of the first certificate from the chain is tested to be equal to the public key of the certificate from the Key Pair for which the import attempt is performed. If the CA Reply does not belong to this entry, the import process will stop and you will be informed by an error message that the CA Reply does not belong to that entry.

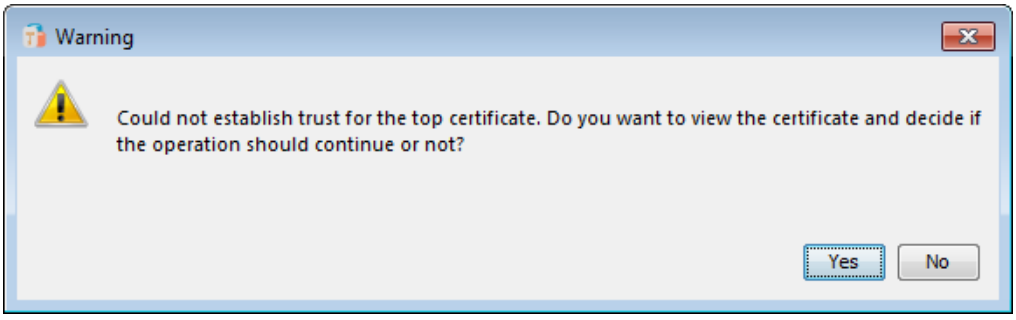
  The error message will contain the information "`The public key of the CA Reply does not match the public key of the key pair entry`", as it can be seen in the screenshot below:



- If the CA Reply contains only a single certificate, a valid trusted certificate chain (a Trust Path) is attempted to be established using the certificates present in the available TrustStores (set from `Tools > Options > Trust Path Options`). If this is not possible, the certificate from the CA Reply will be displayed and you will be prompted to take a decision if the CA Reply should be trusted and imported as it is or not;
- If the CA Reply contains a chain of certificates, the chain is sorted to have the root certificate last and the user certificate first (if this is not already sorted in this way); The chain is then verified for validity which means that for each certificate is checked that its signature is verified by the public key of the certificate at the next higher level in the chain and that its issuer is equal to the subject of the higher level certificate; if the chain is not valid, the import process will stop and you will be informed by an error message that the CA Reply does not contain a valid certificate chain;

  If the chain is valid, then the top certificate of the chain is verified if it is trusted by searching it in the the available TrustStores (set from `Tools > Options > Trust Path Options`). If it is, then the CA Reply is imported. Else, the top certificate of the chain will be displayed and you will be prompted to take a decision if the CA Reply should be trusted and imported or not;

  For example, if the top certificate of a CA Reply is not found within any of the available TrustStores, the following message will be displayed:



  If "No" is selected or the dialog is closed, the import operation will be aborted.

  If "Yes" is selected, the certificate will be displayed in a dialog with the options "Accept Import" to continue the import, or "Cancel Import" to abort the operation which can be seen in Certificate Trust Established by User.

A CA Reply file can be obtained by sending a CSR (Certificate Signing Request) to a Certificate Authority, which will sign it and send back a CA Reply file (usually a file of the type PKCS#7 CA Reply File, having the extension .p7r). Creating a CSR file can be done using CERTivity® as it is described in the section Generate CSR File.

The CA Reply can also be obtained using CERTivity® to sign the CSR file, by performing the following steps:

- Select a Key Pair entry, and generate a CSR file (as described in the section Generate CSR file). A CSR file will be obtained;
- Sign the CSR file obtained at the previous step. The process for signing CSR files is explained in the section Signing CSR Files. The resulting file will be the actual CA Reply file which can then be imported for the Key Pair for which the CSR file was generated;
- Import the CA Reply for the corresponding Key Pair entry.

5.24 Select CA Issuer

In an opened KeyStore window, select a key pair entry and invoke the contextual menu (usually by clicking the right mouse button). From this menu use **`Select CA Issuer`**. If the key pair is not unlocked, you will be prompted to enter the password for the private key associated to the key pair entry. If another key pair found in any of the opened key stores was previously selected as the CA Issuer, that selection will be lost, being replaced with the current one.

When a key pair is selected as the CA Issuer, in the contextual menu the **`Select CA Issuer`** option will be checked. Also, the Status Bar will display information regarding this selection in the following format: `CA Issuer: keystore name / alias for selected issuer`.

5.25 Sign Certificate by <aliasForIssuer>

In an opened KeyStore window, select a key pair entry and invoke the contextual menu (usually by clicking the right mouse button). The following situations can occur:

- If the contextual menu will display an active **`Sign Certificate by <aliasForIssuer>`** option, this means that a CA Issuer was previously selected and you can proceed with the signing operation by selecting the option.
- If the contextual menu will display an inactive **`Sign Certificate by <...>`** option, this usually means that no CA Issuer was previously selected. In order to use this option you must first select a CA Issuer, using the **`Select CA Issuer`** option.
- If the contextual menu will display an inactive **`Sign Certificate by <aliasForIssuer>`** option, this could also mean that you accessed the contextual menu for the same key pair that was previously selected as CA Issuer. This is a precaution measure to make sure the user will not attempt to sign a generated CSR with the same key pair that generated it.

The **`Sign Certificate by <aliasForIssuer>`** option is automatically replicating in one step the effect of the following individual actions:

- Generate a CSR for a selected key pair;
- Sign the previously generated CSR using another key pair and obtain a CA Reply;
- Import the obtained CA Reply in the initial key pair used to generate the CSR.

The workflow of the **`Sign Certificate by <aliasForIssuer>`** action is the following:

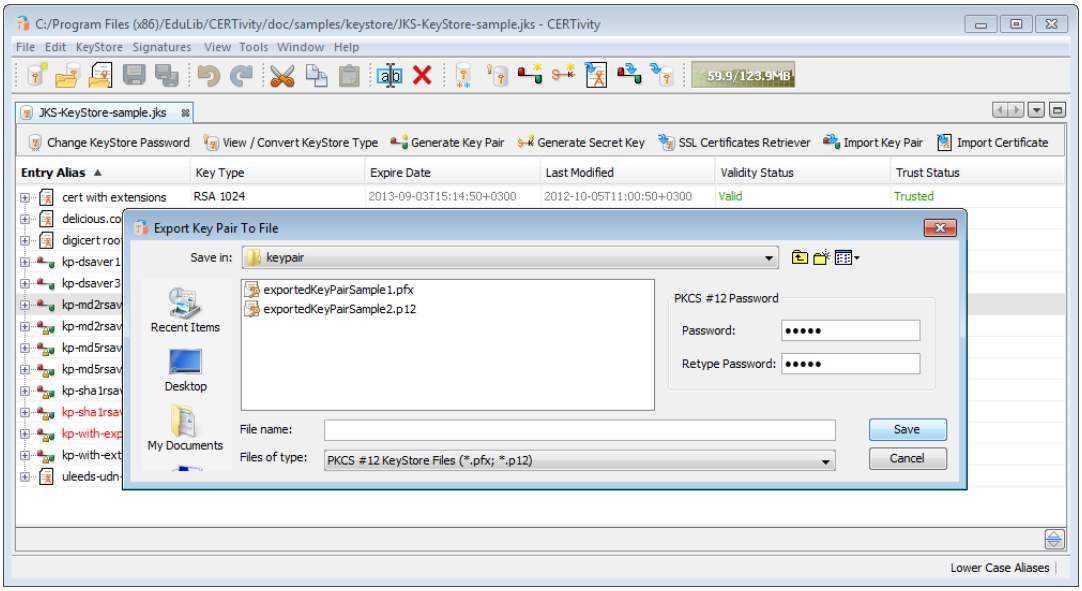
- After you select the **`Sign Certificate by <aliasForIssuer>`** option, you will be prompted to enter the password for the private key associated to the selected key pair entry if need be. If the password entered is correct you may proceed to the next step.

- After the key pair selected was unlocked, the certificate details from the CSR will be shown in a newly opened dialog requiring to provide a Serial Number and double checking the validity period. Additionally, when signing the CSR, certificate extensions can be added to the certificate.
- After the second step was completed and the OK button was pressed, a temporary, internal CA Reply will be created and transparently imported in the selected key pair, thus the target Key Pair will now contain a signed user Certificate and the issuer Certificate as part of the Certificate Chain.

5.26 Export Key Pair

In an opened KeyStore window, select the key pair entry and click on the right mouse button. From the popup menu select **`Export > Export Key Pair`** action. The Key Pair will be exported in the selected file. The Key Pair can be exported in PKCS #12 KeyStore Files format. Also, a PKCS #12 password is required. While exporting a Key Pair, an error might occur if the password length is too long - this has to do with the out of the box Java Cryptography Extension (JCE) limited Strength Jurisdiction Policy Files.

A screenshot for the export Key Pair action is depicted below:



5.27 Export Certificate Chain

In an opened KeyStore window, select the key pair entry and click on the right mouse button. From the popup menu select **`Export > Export Certificate Chain`** action. The Certificate Chain will be exported in the selected file, having the following formats:

- PKCS #7 Certificate Files;
- PKCS #7 Certificate Files (PEM encrypted);
- PKI Path Certificate Files.

5.28 Export Certificate

In an opened KeyStore window, select the certificate entry and click on the right mouse button. From the popup menu select **`Export > Export Certificate`**. The certificate will be exported in the selected file. The certificate can be exported in the following formats:

- X.509 Certificate Files;
- X.509 Certificate Files (PEM encrypted);
- PKCS #7 Certificate Files;
- PKCS #7 Certificate Files (PEM encrypted);
- PKI Path Certificate Files.

Note that certificates can also be exported from signed files or from a SSL source.

5.29 Export Public Key

In an opened KeyStore window, select the certificate entry and click on the right mouse button. From the popup menu select **`Export > Export Public Key`** action. The Public Key will be exported in the selected file. The Public key can be exported in the following formats:

- OpenSSL;
- OpenSSL (PEM encrypted).

5.30 Export Private Key

In an opened KeyStore window, select the key pair entry and click on the right mouse button - if the private key is not unlocked you will be prompted for the private key password. From the popup menu select **`Export > Export Private Key`** action. The exported Private Key type can be PKCS #8 (both binary and PEM) or OpenSSL - this is selectable from the right side of the file chooser window. The exported key can also be encrypted (in this case an encryption algorithm and a password must be provided) or not. For PKCS #8 one can further chose to PEM encode the file, by using the File Chooser filter (Files of type: PKCS #8 Private Key Files (PEM encoded) (*.pkcs8).

The Private Key will be exported in the selected file, having the following formats, depending on the export Private Key Type and file chooser filter selection:

- PKCS #8 Private Key Files;
- PKCS #8 Private Key Files (PEM encoded);
- OpenSSL Private Key Files (PEM encoded).

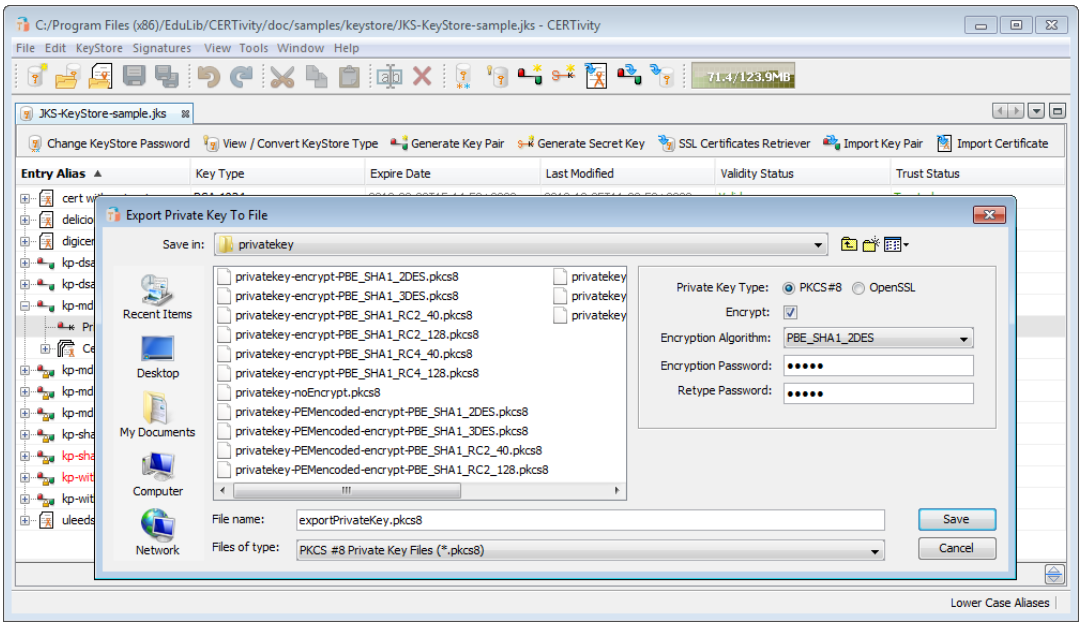
The encryption algorithm is dependent on the selected export Private Key Type and CERTivity automatically changes this in the list.

Table 5.3. Encryption Algorithm for Private Keys

Type	Encryption Algorithm
PKCS #8	PBE_SHA1_2DES
	PBE_SHA1_3DES
	PBE_SHA1_RC2_128
	PBE_SHA1_RC2_40
	PBE_SHA1_RC4_128
	PBE_SHA1_RC4_40
OpenSSL	AES-128-CBC

Type	Encryption Algorithm
	AES-128-CFB
	AES-128-ECB
	AES-128-OFB
	BF-CBC
	BF-CFB
	BF-ECB
	BF-OFB
	DES-CBC
	DES-CFB
	DES-ECB
	DES-EDE-CBC
	DES-EDE-CFB
	DES-EDE-ECB
	DES-EDE-OFB
	DES-EDE
	DES-EDE3-CBC
	DES-EDE3-CFB
	DES-EDE3-ECB
	DES-EDE3-OFB
	DES-EDE3
	DES-OFB
	RC2-40-CBC
	RC2-64-CBC
	RC2-CBC
	RC2-CFB
	RC2-ECB
	RC2-OFB

A screenshot for the export Private Key action is depicted below:



5.31 Rename a KeyStore Entry

You can rename an entry contained in a KeyStore. There are more ways to do the renaming action:

- using **`Edit Menu > Rename`** ;
- using the context menu (by clicking the right mouse button and then on **`Rename`**);
- using the Rename toolbar icon;
- use the keyboard shortcut, by default the **`F2`** key (exactly as the Windows Explorer renaming shortcut) or the **`CTRL+R`** combination.

Note that the alias will not be changed if the alias already exists in the KeyStore. An error message will notify the user in this cases, so the user can give another alias.

5.32 Delete KeyStore Entry

You can delete an entry (either certificate, key pair or secret key) contained in a KeyStore. There are more ways to do the deleting action:

- using **`Edit Menu > Delete`** ;
- using the context menu (by clicking the right mouse button and then on **`Delete`**);
- using the Delete toolbar icon;
- use the keyboard shortcut, by default **`Delete`** key.

Note that a "Confirm Entry Delete" window will be displayed first, so the user has the possibility to change his mind regarding the delete action. In case of the native Windows Root KeyStore the OS will prompt for a native confirmation dialog as well.

5.33 Copy KeyStore Entry

You can copy KeyStore entries (Certificates, Key Pairs and Secret Keys) as well as certificates part of a Key Pair's Certificate Chain into the clipboard. There are more ways to do the copying action:

- using **Edit Menu > Copy** ;
- using the context menu (by clicking the right mouse button and then on **Copy**);
- using the Copy toolbar icon;
- use the keyboard shortcut, by default **CTRL+C** key.

5.34 Cut KeyStore Entry

You can remove the currently selected entry from a KeyStore and place it in the clipboard. There are more ways to do the cut action:

- using **Edit Menu > Cut**;
- using the context menu (by clicking the right mouse button and then on **Cut**);
- using the Cut toolbar icon;
- use the keyboard shortcut, by default **CTRL+X** key.

5.35 Paste KeyStore Entry

You can insert the KeySstore entry (including a Certificate part of a Certificate Chain) from the clipboard in another KeyStore or even in the same KeyStore (an overwrite/rename confirmation dialog is invoked in this case). The paste action is active when, in the target KeyStore, the selection is not on a sub-entry, but on a main entry or there is nothing selected. There are more ways to trigger the paste action:

- using **Edit Menu > Paste**;
- using the context menu (by clicking the right mouse button and then on **Paste**);
- using the Paste toolbar icon;
- use the keyboard shortcut, by default **CTRL+V** key.

In case the alias already exists in the KeyStore, the user has to choose between the following actions:

- Overwrite - in case of pasting a single entry and overwriting the existing entry;
- Overwrite all- in case of pasting more entries and overwriting the existing entries;
- Rename - in case of pasting a single entry;
- Rename all- in case of pasting more entries;
- Skip - in case of skipping the paste for a single entry;
- Skip all - in case of skipping the paste of all entries.

6. CERTivity®'s Signatures

6.1 Verify

Using CERTivity, you can verify signatures for:

- JAR files;
- XML files;
- PDF files.

using **`Menu Signature > Verify`** command.

Note

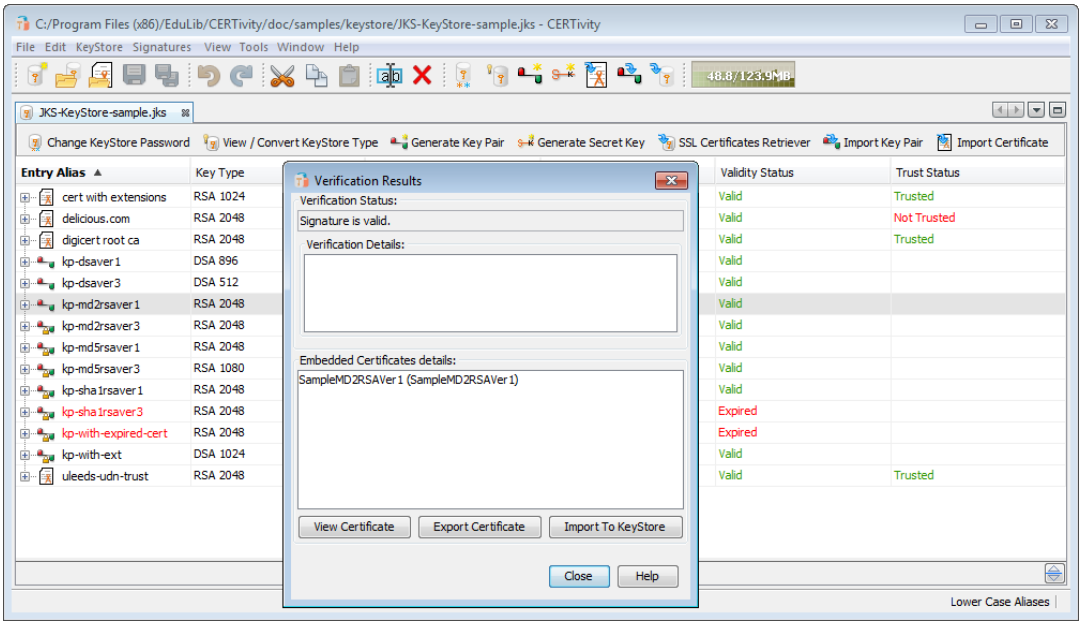
You can use the examples provided in the distribution kit in `doc/samples` folder, to test verify and sign features.

6.1.1 Verify JAR Signatures

When **verifying a JAR signature**, a KeyStore entry can be selected for verifying the entry certificates. In case there is no KeyStore selected, you can continue verification of the JAR signature without checking the existence of the certificates from the JAR entries in the KeyStore. An error will be displayed if KeyStore file could not be loaded or if the KeyStore password is wrong or the file is corrupt. A successful JAR file verification occurs if the signature(s) are valid, and none of the files that were in the JAR file when the signatures were generated have been changed since then. After the JAR signature verification operation, the messages that will be displayed are:

- "`The JAR file was verified.`" in case of successful JAR signature verification;
- "`The JAR file was not verified.`" in case the JAR file has not a valid signature.

The embedded certificate(s) can be viewed, exported into an external file or directly imported into the active KeyStore.



Note

You can use JAR examples provided in the distribution kit in `doc/samples/jar` folder, to test the verify JAR features.

When importing a selected embedded certificate into the active KeyStore, the certificate trust will be verified in the same way it is verified when importing a trusted certificate into the active KeyStore. If a Trust Path can not be established using the provided TrustStores and the Trust Path validation options (which can be set from `Tools > Options > Trust Path Options`), a message will be displayed informing about that and asking if the certificate should be displayed for user verification. If "No" is selected, or the dialog is closed, the import operation is aborted. If "Yes" is selected, the certificate details will be displayed, and the user will have the option to continue the import operation (by selecting the "Accept Import" button) or to abort it (by selecting the "Cancel Import" button).

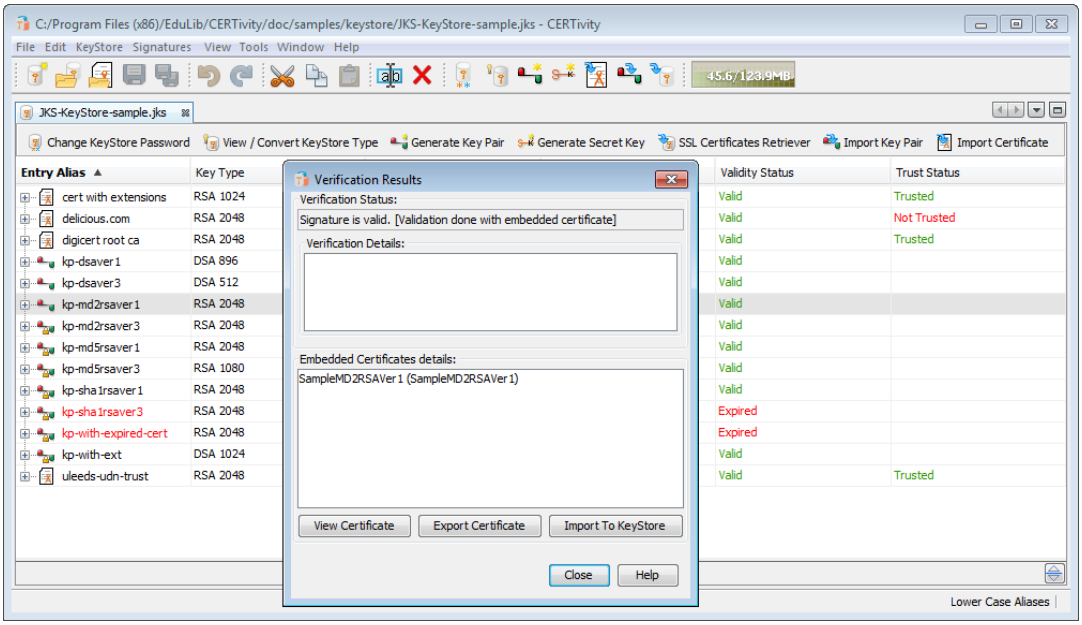
6.1.2 Verify XML Signatures

XML signatures can be used as authentication credentials or as a way to check data integrity. XML signatures can be applied to XML file, HTML pages, gif files, XML-encoded data. When validating an XML signature, an XML file must be chosen first. If there is no certificate embedded, the certificate identified by the current selected entry is used to validate the XML signature.

After the XML signature verification process, the messages that will be displayed are:

- "`File not signed.`" in case the XML file was not signed;
- "`Signature is invalid.`" in case the XML file signature is not valid;
- "`Signature is valid.`" in case the XML file signature is valid. The trusted state of the embedded certificate is not checked.

If the certificates are embedded these will be shown under the "`Certificates details`" panel and details can be viewed or Certificates exported or directly imported into the active KeyStore.



Note

You can use XML examples provided in the distribution kit in `doc/samples/xml` folder, to test the verify XML features.

When importing a selected embedded certificate into the active KeyStore, the certificate trust will be verified in the same way it is verified when importing a trusted certificate into the active KeyStore. If a Trust Path can not be established using the provided TrustStores and the Trust Path validation options (which can be set from `Tools > Options > Trust Path Options`), a message will be displayed informing about that and asking if the certificate should be displayed for user verification. If "No" is selected, or the dialog is closed, the import operation is aborted. If "Yes" is selected, the certificate details will be displayed, and the user will have the option to continue the import operation (by selecting the "Accept Import" button) or to abort it (by selecting the "Cancel Import" button).

6.1.3 Verify PDF Signatures

The **Portable Document Format (PDF)** allows to digitally sign a document by inserting a cryptographic signature value in the file. A signature is in most cases represented by a signature field containing the name and other attributes of the signer. When verifying a PDF signature, a PDF file must be chosen first. The digital signatures CERTivity understands for PDF verification are the public/private-key encrypted document digest with the standard SubFilter values `adbe.x509.rsa_sha1`, `adbe.pkcs7.detached`, and `adbe.pkcs7.sha1`. The exact specified handler (the `Filter` value) is ignored when verifying the signature according to the PDF Reference “An application may substitute a different handler when verifying the signature, as long as it supports the specified SubFilter format.”

After verifying the PDF signature, a dialog called "`Verification Results`" is presented for the Document containing the global document status and details for each Signature found. The global Verification Status can be one of:

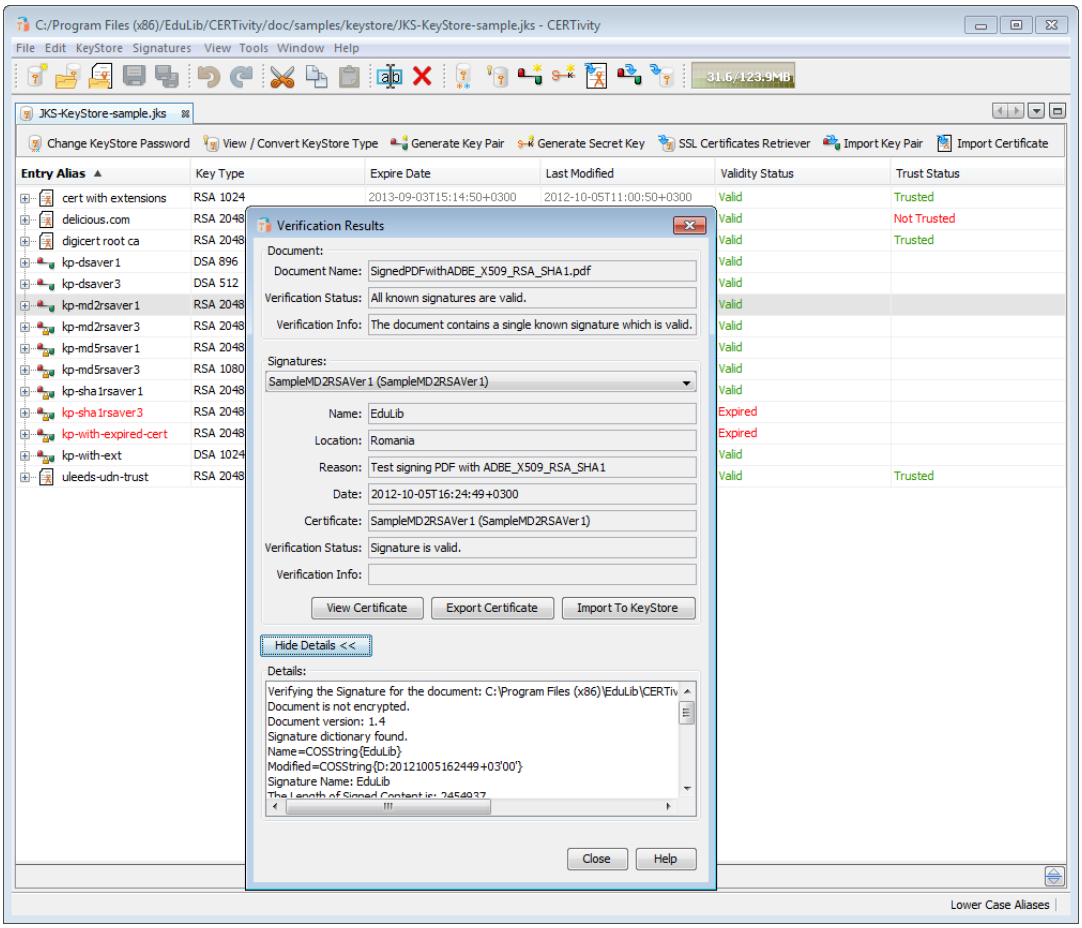
- "`File not signed.`" in case the PDF file was not signed;
- "`At least one known signature is invalid.`" in case at least one of the supported (known) PDF file signature is not valid;
- "`All known signatures are valid.`" in case all of the supported (known) PDF file signatures are valid according to the sub-filter values and algorithm (including the digest being recomputed and compared with the one stored in the signature). The trusted state of the embedded certificates is not checked.
- "`Unknown.`" in case the document is containing only unsupported SubFilters.

For each signature recognized in the document, you can see the signer details, such as name, location, reason, date, certificate, signature verification status and verification info. The embedded certificate of each signature can be viewed and even exported into an external file.

A Verbose text report can be analysed (`Show Details`) revealing the reason why, for example, some signatures are not valid, or revealing the value of the SubFilter/Filter. This is especially useful to observe the details for invalid cases as many information is logged. For example, according to the `adbe.pkcs7.sha1` SubFilter the signature process involves two digests - the SHA1 digest of the byte range which is encapsulated in the PKCS#7 signed-data field with ContentInfo of type Data, and then this signed-data field is digested and signed according to the PKCS#7 standard. So there are two digest verified, and if one of these fails the validation fails, and this could be visible by inspecting the Details section, for example:

`The calculated SHA1 Message Digest coincides with the encapsulated PKCS#7 signed-data field. Continuing the signature verification procedure. Digest Mismatch [message-digest attribute value does not match calculated value].`

Although a signature may not be valid, the option `View Certificate > Export Certificate > Import to KeyStore` can be available in many situations (usually if preliminary validation passes) as long as the certificate is embedded according to the PDF standards. For example, in the case above where the second message-digest mismatch the embedded certificate can still be viewed/exported/imported to KeyStore.



Note

You can use PDF examples provided in the distribution kit in `doc/samples/pdf` folder, to test the verify PDF features.

Note

Verifying the signature of a PDF which is encrypted is not supported.

When importing a selected embedded certificate into the active KeyStore, the certificate trust will be verified in the same way it is verified when importing a trusted certificate into the active KeyStore. If a Trust Path can not be established using the provided TrustStores and the

Trust Path validation options (which can be set from `Tools > Options > Trust Path Options`), a message will be displayed informing about that and asking if the certificate should be displayed for user verification. If "No" is selected, or the dialog is closed, the import operation is aborted. If "Yes" is selected, the certificate details will be displayed, and the user will have the option to continue the import operation (by selecting the "Accept Import" button) or to abort it (by selecting the "Cancel Import" button).

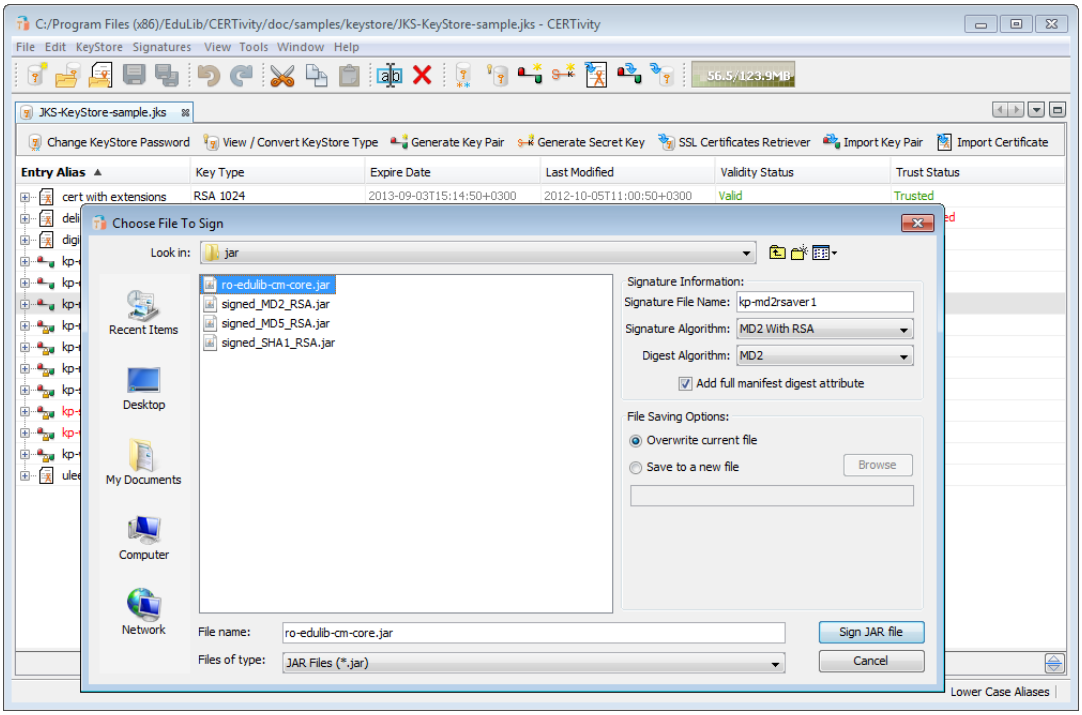
6.2 Sign

6.2.1 Signing JAR Files

In order to sign a JAR file, make the following steps:

- select a Key Pair from the KeyStore tree table;
- either choose the main menu **`Signatures > Sign > JAR file`** or the contextual menu **`Sign > Jar file`**;
- unlock the Key Pair if requested by providing its password;
- select the JAR file that will be signed;
- complete the signature information:
    1. signature file name;
    2. digest algorithm:
        - MD2 (reference can be found in RFC 1319);
        - MD5 (reference can be found in RFC 1321);
        - SHA1(reference can be found in FIPS 180--3);
    3. signature algorithm - SHA1 with DSA;
    4. check the "`Add full manifest digest attribute`" option in case you want this attribute to be added at the signature;

The signed JAR file can be overwritten or can be saved in an other location, according to the options selected in the "`File Saving Options`" area.



Note

You can use JAR examples provided in the distribution kit in `doc/samples/jar` folder, to test the sign JAR features.

6.2.2 Signing XML Files

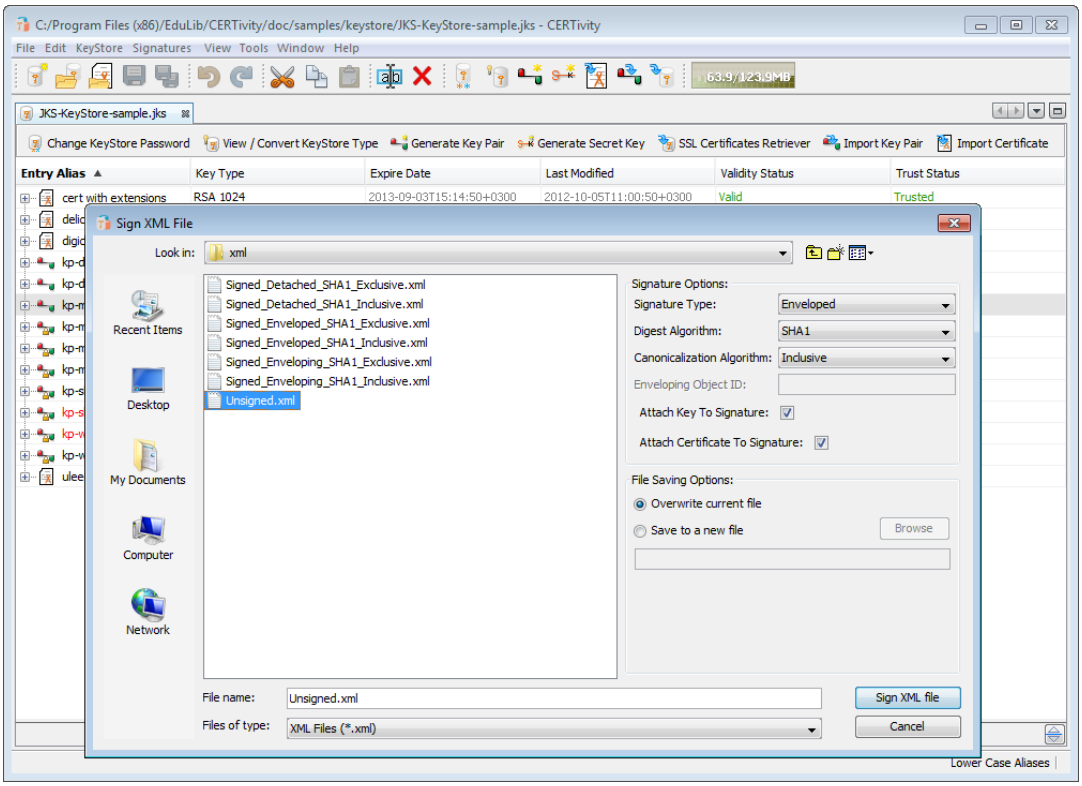
In order to sign an XML file, make the following steps:

- select a Key Pair from the KeyStore tree table;
- either choose the main menu **`Signatures > Sign > XML file`** or the contextual menu **`Sign > XML file`**;
- unlock the Key Pair if requested by providing its password;
- select the XML file that will be sign;
- complete the signature options:
  1. signature type:
     - enveloped - the signature applied over the XML content that contains the signature as an element.
     - enveloping - the signature applied over the content found within an Object element of the signature itself.
     - detached - the signature applied over the content external to the Signature element, and it can be identified by way of a URI or a transform.
  2. digest algorithm:
     - SHA1;
     - SHA256;

- SHA512;

3. check "Attach key information to signature" and "Attach certificate information to signature" in case you want to attach those information to the signature.

The signed XML file can be overwritten or can be saved in an other location, according to the options selected in the "File Saving Options" area.



Note

You can use XML examples provided in the distribution kit in `doc/samples/xml` folder, to test the sign XML features.

An example of using the XML signature is for signing PAD files. PAD is the Portable Application Description file in which an author provides product descriptions and specifications for online sources in a standard way. PAD signing provides a mechanism by which PAD file consumers can ensure that PAD files are authentic.

The steps for signing a PAD file using CERTivity are:

- create an XML PAD file according to the standards;
- create a new PKCS12 keystore;
- generate a key pair for which the organization name of the certificate to match exactly with the company name or the first and last name defined in your PAD file. Add Extended Key Usage extension (Code Signing) and Key usage extension (Digital Signature) to the certificate;
- sign your XML PAD file using this keystore.

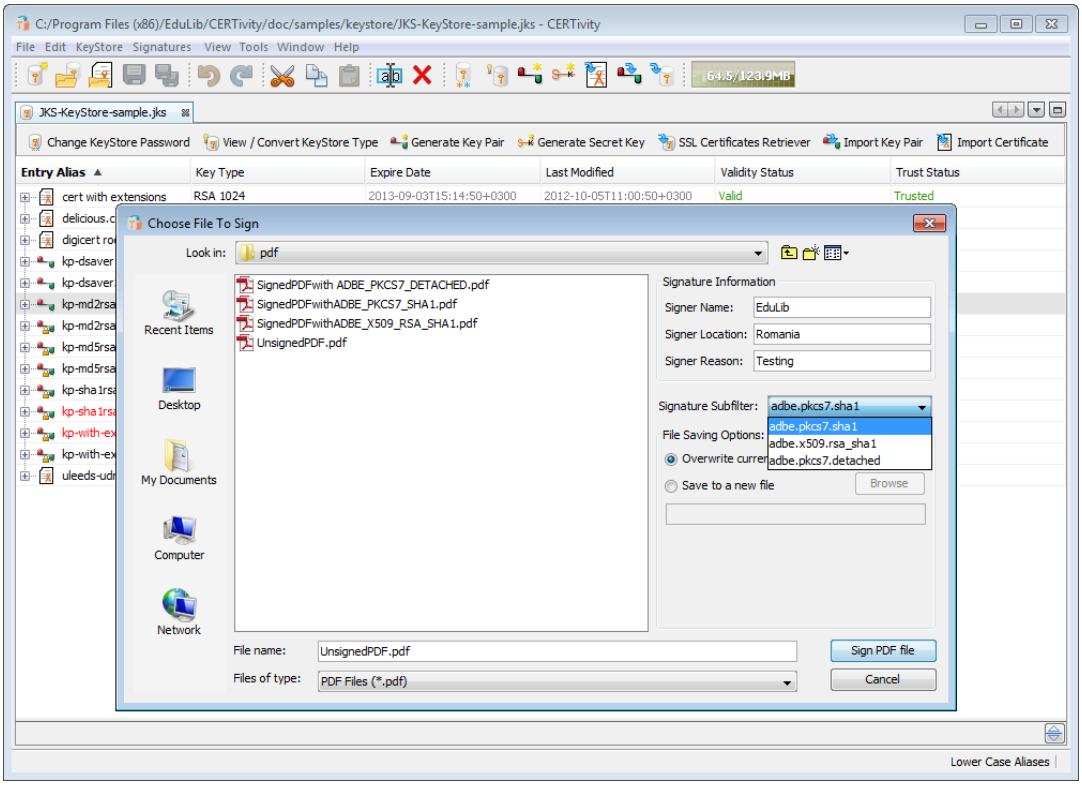
6.2.3 Signing PDF Files

CERTivity can digitally sign by public/private-key encrypted byte range digest a PDF document, supporting the standard SubFilter values `adbe.x509.rsa_sha1`, `adbe.pkcs7.detached`, and `adbe.pkcs7.sha1`. The signature supported by CERTivity is of *document* (or *ordinary*) type (according to the PDF Reference, version 1.7) and without a visual representation. The name of the signature handler (`Filter`) is `Adobe.PPKLite`. Multiple signatures can be applied incrementally. The signature process is currently not conditioned by the existence of other signature types or by any post-signing changes (`DocMDP`).

In order to sign a PDF file, make the following steps:

- select a Key Pair from the KeyStore tree table;
- either choose the main menu **`Signatures > Sign > PDF file`** or the contextual menu **`Sign > PDF file`**;
- unlock the Key Pair if requested by providing its password;
- select the PDF file that will be signed;
- complete the signature information:
    1. Signer Name;
    2. Signer Location;
    3. Signer Reason;
    4. select signature SubFilter - standard value that represents the encoding to use when signing the PDF file:
        - *adbe.pkcs7.sha1* - The adbe.pkcs7.sha1 digest of the byte range is encapsulated in the PKCS#7-signed data field;
        - *adbe.pkcs7.detached* - No data is encapsulated in the PKCS#7-signed data field;
        - *adbe.x509.rsa_sha1* - The adbe.x509.rsa.sha1 digest uses the RSA encryption algorithm and SHA-1 digest method. This SubFilter is available only for RSA Key Pairs.

The signed PDF file can be overwritten or can be saved in another location, according to the options selected in the "`File Saving Options`" area.



Note

You can use PDF examples provided in the distribution kit in `doc/samples/pdf` folder, to test the sign PDF features.

Note

Signing a PDF which is encrypted is not currently supported. Signing a PDF containing xref-streams is not fully supported and for example the size of the generated signed PDF could become much too large and the time for processing is pretty expensive. A warning message is presented if xref-streams are detected, with the option to continue the signing procedure.

6.2.4 Signing CSR Files

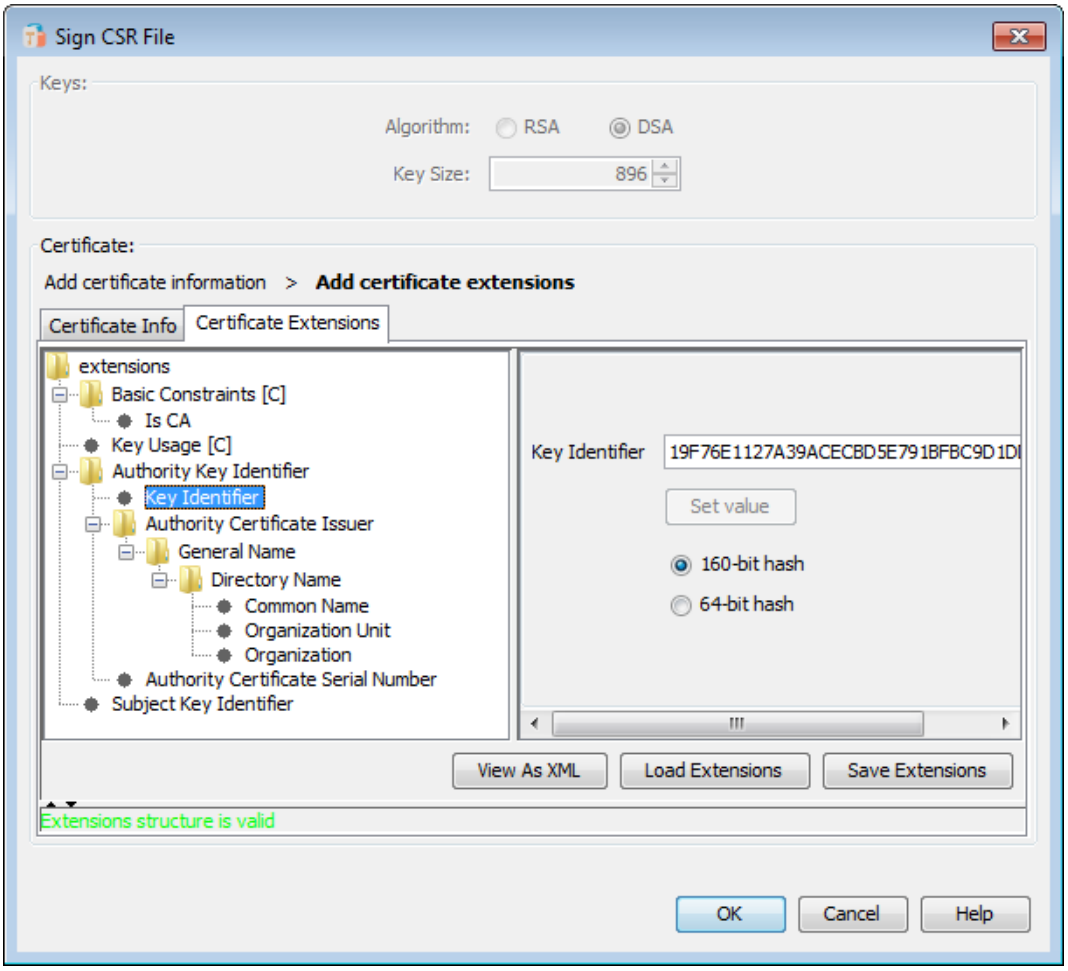
In order to sign a CSR file, make the following steps:

- select a Key Pair from the KeyStore tree table;
- either choose the main menu **`Signatures > Sign > CSR file`** or the contextual menu **`Sign > CSR file`**;
- unlock the Key Pair if requested by providing its password;
- select the CSR file that will be signed;
- select a file where to save the CA Reply.

The certificate details from the CSR will be shown in a new opened dialog requiring to provide a Serial Number and double checking the validity period. Additionally, when signing the CSR

file, certificate extensions can be added to the certificate that will result in the CA Reply. Adding the extensions can be done in the same way as it is done when creating a self signed certificate when generating a new key pair (please see "**`Generate Key Pair`**" and "**`Add Extensions To Certificate`**" chapters for more details).

The dialog that allows adding extensions when signing a CSR can be seen in the screenshot below:



Using the information mentioned above (Serial Number, Extensions, and the information collected from the CSR), the CSR file will be signed generating a CA Reply.

Note

You can use CSR examples provided in the distribution kit in `doc/samples/csr` folder, to test the sign CSR features.

7. FAQ

7.1 How to Install the Unlimited JCE Jurisdiction Policy?

If you exported your PKCS#12/Uber KeyStore file from your browser and used a password that is greater than 7 characters, you may need to download and install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files so you can read the file.

This is a matter of U.S. policy and U.S. export controls (not due to technical reasons).

You can download the required files from:

- http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html [http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html].

In order to install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files, follow the steps:

1. Download the unlimited strength JCE policy files;
2. Uncompress and extract the downloaded file - this will create a subdirectory called jce;
3. Install the unlimited strength policy JAR files:
   - the standard place for JCE jurisdiction policy JAR files is: `<java-home>/lib/security` [Unix] or `<java-home>\lib\security` [Win32]

7.2 Which Are the Available KeyStores Types in CERTivity Application?

The available KeyStore types are:

- *jks* - Java KeyStore (Oracle's KeyStore format);
- *pkcs12* - Public-Key Cryptography Standards #12 KeyStore (RSA's Personal Information Exchange Syntax Standard);
- *jceks* - Java Cryptography Extension KeyStore (More secure version of JKS);
- *bks* - Bouncy Castle KeyStore (Bouncy Castle's version of JKS);
- *uber* - Bouncy Castle UBER KeyStore (More secure version of BKS);
- *Windows Root CA KeyStore*;
- *Windows User KeyStore*.

7.3 Sometimes the Entry Name (Alias) Changes its Case

For UBER KeyStore (Bouncy Castle UBER KeyStore) as well as in PKCS12 KeyStore, the alias name is case sensitive. For other types of KeyStore, the alias name is not case sensitive.

So, when converting from UBER or PKCS12 to other type of KeyStores, or when moving an entry from a UBER or PKCS12 type KeyStore to an other, the alias name will be changed to lower case.

7.4 Fonts too large

If the system font is too big, on some platforms the application is not rendering well out of the box - especially text is getting out of the editing fields.

If you encounter such a case and you are not using Gnome as a Desktop Environment you can edit the file `${certivity_home}/etc/config/certificates.conf` and under the option `default_options` modify the parameter `--fontsize <size>`.

The default out of the box value for `size` is `11` points at a `72` DPI .

If you are using Linux and Gnome as a Desktop Environment, the default GTK Look and Feel is ignoring the font size specified via the Java command line arguments, as well as any possibility of specifying the font from Java. If this is your case, the options you have are either to set the Gnome system font size to a smaller value or to use the Metal Look and Feel.

To change the font settings in Gnome, use the appropriate menu item in the Gnome menus. The menus differ depending on the OS distribution and version. In recent Gnome releases, it is usually: *System > Preferences > Appearance, tab Fonts, row Application font*, for the font size continuing with the **Details** section from the same **Fonts** tab, then **Resolution** box for the DPI if needed.

You can switch to the Metal or Motif Look and Feels by editing the file `${certivity_home}/etc/config/certificates.conf` and adding under the option `default_options` the parameter `--laf javax.swing.plaf.metal.MetalLookAndFeel`.

7.5 Where is the Help Window on MAC OS?

On MAC OS platforms, the help window might be positioned behind some of the application windows (e.g. dialog windows). In this case, you can move the application window out of the help window in order to be able to read and use the Java Help, or minimize the help window and reopen it.

7.6 Having rendering issues?

In case CERTivity KeyStores Manager has graphical or rendering issues such as:

- incorrect repaints;
- non-viewable text and buttons;
- incorrect rendering windows;
- black pop-up windows;
- even system crashes for certain system configurations.

Then you need to customize how the 2D painting system operates. For example some of the rendering issues are due to an incompatibility between Java, Windows DirectDraw and the Video card and are generally affecting any Java Desktop application on that machine.

Oracle (Sun) has some parameters available for fixing the incorrect rendering, for the Windows and DirectDraw case they are :

- **ddoffscreen** used to turn off the Java 2D system's use of DirectDraw and Direct3D for offscreen surfaces;
- **noddraw** used to turn off the Java 2D system's use of DirectDraw and Direct3D completely;
- **d3d** used to turn off the Java 2D system's use of Direct3D.

For the whole list of switches there are more details on http://docs.oracle.com/javase/7/docs/technotes/guides/2d/flags.html [http://docs.oracle.com/javase/7/docs/technotes/guides/2d/flags.html].

When experiencing rendering problems on Microsoft Windows systems, a good solution is to deactivate DirectDraw. Try to set the following parameters (properties) **-J-Dsun.java2d.ddoffscreen=false -J-Dsun.java2d.d3d=false -J-Dsun.java2d.noddraw=true**

in the file ${certivity.home}/etc/certivity.conf(.sh) from the CERTivity installation directory (default for the JRE installer case is C:\Program Files (x86)\EduLib\CERTivity 1.0\etc \certivity.conf) under the default_options directive. For safety reasons do a backup of that file before editing.

Edit the file so that the default_options line becomes:

default_options="--branding certivity -J-Xms128m -J-Xmx256m -J-Dsun.java2d.ddoffscreen=false -J-Dsun.java2d.d3d=false -J-Dsun.java2d.noddraw=true --fontsize 11"

Make sure it remains a single line (i.e. no line terminator inside) and save the file. Close CERTivity and then restart it.

7.7 Why do I get an "Access Denied" error when trying to save a KeyStore to a file located in Program Files?

In Windows 7, saving or modifying files located in the Windows special folders (such as Program Files, Program Files (x86), etc.) may require your permission due to the User Account Control (UAC) function. The UAC function should ask for your permission any time a program wants to make a major change to your computer, and does not allow modifying or writing the files into the specified directory if the permission is not granted.

In many applications, saving to or modifying files from the special directories leads to an "Access Denied" error, due to the restriction imposed by the UAC. That is why, when trying to save a KeyStore or export a Certificate or Private / Public Key to a file located in one of the Windows special folders you might get an "Access Denied" type error.

To avoid this from happening, you can try setting the UAC function to a lower level or turning it off. A detailed description about the UAC settings and the potential impact of each setting on the security of your computer can be found at http://windows.microsoft.com/is-IS/windows7/What-are-User-Account-Control-settings. To find out how to turn off the UAC function you can find more information at http://windows.microsoft.com/en-US/windows-vista/Turn-User-Account-Control-on-or-off. Also, if you don't want to modify the UAC function settings or to turn it off, you can try to save the files to a different folder where writing to files is not restricted.

8. License Agreement

This EULA (End User License Agreement) is a legal agreement between you (either an individual or an entity), the end user, and EduLib S.R.L. for the use of CERTivity KeyStores Manager Software, its enhancements/derivatives and all supporting documentation (the "Software"). PLEASE READ THIS EULA CAREFULLY BEFORE COMPLETING THE INSTALLATION PROCESS AND USING THE SOFTWARE. By obtaining the Software, you agree to comply with the terms and conditions of this license. Installing and using the Software will signify your agreement to be bound by these terms and conditions. If you do not agree to these terms and conditions, do not continue installing the Software or do not continue using the software and destroy all its copies.

This is a license agreement and not an agreement for sale.

8.1 Definition

- "EduLib" means EduLib S.R.L.;
- "Software" means the executable code of CERTivity KeyStores Manager Software, its enhancements/derivatives and all supporting documentation;
- "Agreement" means this End User License Agreement;
- "License" means the permission granted by EduLib to use the licensed Software product.

8.2 Grant of License

The Software product is owned by EduLib. It is licensed, not sold.

The Software product is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. EduLib reserves all intellectual property rights, including copyrights and trademark rights.

This License permits you to use a single copy, or multiples copies if you are the only user of the Software product identified above. According to your order, the Software is licensed as a single product, to an individual user for Single User License, or group of users for Multi-User License. This Agreement requires that each user of the Software be licensed, either individually, or as part of a group. A Multi-User License provides for a specified number of users to use this Software at any time. This does not provide for concurrent user Licensing. Each user of this Software must be covered either individually, or as part of a group Multi-User License. The Software is in use on a computer when it is loaded into the temporary memory (i.e. RAM) or installed into the permanent memory (e.g. hard disk) of that computer. This Software may be installed on a network provided that appropriate restrictions are in place limiting the use to licensed users only.

Backup Copy: You may make copies of the Software product and the Software product License as reasonably necessary for the use authorized above, including as needed for backup and/or archival purposes. No other copies may be made. Each copy must reproduce all copyright and other proprietary rights notices on or in the Software product.

The above apply for: Trial License, Standard License and Professional License of the Software.

8.3 Restricted Use for Evaluation

This Software product can be used in conjunction with a free evaluation Software License. During the evaluation period, EduLib grants you a limited, non-exclusive, non-transferable,

non-renewable License to copy and use the Software for 30 days evaluation purposes only and not for any commercial use. At EduLib discretion, EduLib may provide limited support through email or discussion forums at EduLib web site. Evaluation Software has been limited in some way either through time outs, restricted use or evaluation warnings. EduLib bears no liability for any damages resulting from use (or attempted use after expiration) of the Software product, and has no duty to provide any support before or after the expiration date of an evaluation License.

On or before expiry of the evaluation period you may pay a License fee to obtain the right to use the Software for extended use. If you do not pay such a fee you must destroy all copies of the Software.

8.4 Support Services

EduLib may provide you with support services related to the Software product according to your order.

Any supplemental software code or related materials that EduLib provides to you as part of the support services, in periodic updates to the Software product or otherwise, is to be considered part of the Software product and is subject to the terms and conditions of this Agreement.

With respect to any technical information you provide to EduLib as part of the support services, EduLib may use such information for its business purposes without restriction, including for product support and development. EduLib will not use such technical information in a form that personally identifies you without first obtaining your permission.

Technical support is provided via electronic mail at the following address: support@edulib.ro [mailto:support@edulib.ro] . EduLib will use its best efforts to provide you with technical support within 2 working days of your request. Please check our web site to find our latest contact information.

8.5 Refund

Even if the Software is provided free of charge during the Trial Period to allow potential customers to evaluate and test it before paying the License fee, EduLib allows for 30 days money refund.

8.6 Restrictions

You may not use, copy, or distribute the Software product, except as granted by this Agreement, without written authorization from EduLib.

You may not tamper with, alter, or use the Software product in a way that disables, circumvents, or otherwise defeats its built-in licensing verification and enforcement capabilities.

You may not remove or alter any trademark, logo, copyright or other proprietary notice, legend, symbol or label in the Software product.

You may not modify or create derivative copies of the Software License.

You may not reverse engineer, decompile, defeat License encryption mechanisms, or disassemble the Software product or the Software License Key (File).

You may not modify the Software or create derivative works based upon the Software.

You may not rent, lease, lend, or in any way distribute or transfer any rights in this Agreement or the Software product to third parties without EduLib's written approval, and subject to written agreement by the recipient of the terms of this Agreement.

8.7 High Risk Activities

The Software product is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, in which the failure of the Software product, or any software, tool, process, or service that was developed using the Software product, could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). Accordingly, EduLib and its suppliers and licensors specifically disclaim any express or implied warranty of fitness for High Risk Activities. You agree that EduLib and its suppliers and licensors will not be liable for any claims or damages arising from the use of the Software product, or any software, tool, process, or service that was developed using the Software product, in such applications.

8.8 Third Party Rights

Any software provided along with the Software that is associated with a separate license agreement is licensed to you under the terms of that license agreement. This license does not apply to those portions of the Software. Copies of these third party licenses are included in all copies of the Software.

8.9 Laws and Regulations

According to the EU Export regulations, the Software is not classified as a dual use 5D002.c product as all its functionalities qualifies it for the exceptions from category 5A002.a.1, which means no export control because actually the product is not a dual use one according to the EU COUNCIL REGULATION (EC) No 428/2009 of 5 May 2009. This conclusion was reached through a consultancy with the Romanian National Authority for Export Controls (ANCEX [http://www.ancex.ro]).

Nevertheless users are advised and agree that they must come to fully understand, respect and comprehend the cryptographic functionality found in the licensed Software and must agree and undertake to comply with any laws of their forum pertaining to said cryptographic functionality.

You hereby also acknowledge that:

- you have obtained the Software directly from EduLib, through its Romanian-based web site;
- the Software developed by EduLib is of Romanian origin and therefore subject to Romania's and European Union's laws, rules and regulations.

You hereby irrevocably declare and agree that the Software you obtained from http://www.edulib.com:

- is publicly available, without restriction except the normal copyright ones by means of electronic transactions and a web download;
- has cryptographic functionality that cannot be easily changed by you;

- is installable by you without substantial support from EduLib; and
- details of the product is available in the documentation included in the product, as well as available without restriction on EduLib site.

You hereby warrant that:

- you are allowed to obtain a copy of this Software at your location or in your forum or territory;
- your installation and usage of this Software conforms to your country's applicable laws and regulations;
- you will ensure that users having access to your copy of this Software will abide by the applicable laws and regulations; and
- you undertake to defend and indemnify EduLib (including assuming EduLib attorneys fees) if you violate any laws or regulations pertaining to cryptographic software or functionality;
- you have the obligation to obtain at your own expense any license or authorizations required by any legal authority for acquisition, delivery or use of the products, and if necessary produce evidence to EduLib. You should be liable for all expenses or charges incurred by EduLib from your failure to obtain such license or authorizations.

8.10 Limited Warranty

You have ensured with the above mentioned evaluation version that the Software works according to your requirements and the advertised features. EduLib disclaims all warranties for deficiencies that are reasonably discoverable with the evaluation version of the Software. You acknowledge that the Software cannot be completely error-free. EduLib disclaims all warranties regarding non-severe deviations of the advertised features of the Software. EduLib and its third party suppliers and licensors disclaim all other representations, warranties, and conditions, expressed, implied, statutory, or otherwise, including, but not limited to, implied warranties or conditions of merchantability, satisfactory quality, fitness for a particular purpose, title, and non-infringement. The entire risk arising out of use or performance of the Software product remains with you.

8.11 Limitation of Liability

To the maximum extent permitted by applicable law, in no event shall EduLib or its suppliers be liable for any special, incidental, indirect, or consequential damages whatsoever (including, but not limited to, damages for loss of profits or confidential or other information, for business interruption, for personal injury, for loss of privacy, for failure to meet any duty including of good faith or of reasonable care, for negligence, and for any other pecuniary or other loss whatsoever) arising out of or in any way related to the use or inability to use the software, the provision of or failure to provide support services, or otherwise under or in connection with any provision of this Agreement, even in event of fault, tort (including negligence), strict liability, breach of contract or breach of warranty of EduLib or any supplier, and even if EduLib or any supplier has been advised of the possibility of such damages.

Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, EduLib's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Agreement between EduLib and you.

8.12 General

EduLib reserves the right at any time to cease the support of the Software and to alter prices, features, specifications, capabilities, functions, licensing terms, release dates, general availability or other characteristics of the Software.

This Agreement embraces the full and complete understanding of the parties as to the subject matter hereof and may not be diluted or modified except by written amendment which expressly refers to this Agreement and which is duly executed by both parties.

This Agreement is to be governed by and construed in accordance with Romanian laws.

The United Nations Convention on Contracts for the International Sale of Goods shall not apply to this Agreement.

8.13 Contact Information

If you have any questions about this Agreement, or if you want to contact EduLib for any reason, please email to support@edulib.ro [mailto:support@edulib.ro].

8.14 Changes to our License Agreement

If we decide to change our License Agreement, we will post those changes on http://www.edulib.com/products/keystores-manager/license/, and update the License Agreement modification date below.

This agreement was last modified on 19th March 2012.

9. Sales and Support

Technical support is provided via electronic mail at the following address: mailto://support@edulib.ro. EduLib will use its best efforts to provide you with technical support within 2 working days of your request. Please check our web site to find our latest contact information.

EduLib software can obtained from http://www.edulib.com.

Terms and Conditions of Sale are available at http://www.edulib.com/terms-of-sale/.

Appendix A. CERTivity®'s Features Matrix

The existence and capabilities of the CERTivity features are controlled by the category of your license - Standard, Professional or Trial versions.

Feature	Trial License	Standard License	Professional License
KeyStore Management			
Create a New KeyStore	Limited to 5 New Actions per instance	+	+
Open an Existent KeyStore	Limited to 5 Open Actions per instance	+	+
Open Windows Root CA KeyStore	+	-	+
Open Windows User KeyStore	+	-	+
Open JREs CA TrustStores	+	+	+
Save a KeyStore	+	+	+
Convert KeyStore Type	+	+	+
Change the KeyStore Password	+	+	+
Delete KeyStore Entry	+	+	+
Change KeyStore Entry Alias	+	+	+
Import Certificate into KeyStore	+	+	+
Generate Key Pair	+	+	+
Generate Secret Key	+	+	+
SSL Certificates Retriever	+	-	+
Import Certificate from Signed JAR	+	+	+
Import Certificate from XML and PDF	+	-	+
Import Key Pair	+	+	+
Add Certificate Extensions	+	+	+
Save Certificate Extensions as XML	+	+	+
View Private Key Details	+	+	+
Extend Validity of Self Signed Key Pairs	+	+	+
Regenerate Key Pair	+	+	+
Copy and paste entries from one KeyStore to another	+	+	+
Change a Key Pair’s password	+	+	+
Password manager to avoid entering Key passwords each time	+	+	+

Feature	Trial License	Standard License	Professional License
Emphasizing expired and about to expire Certificates or Key Pairs	+	+	+
Generate self signed Key Pairs (Private Key with corresponding Certificate)	+	+	+
Copy to clipboard Certificates from Key Pair's Certificate Chain	+	+	+
View Public Key, Certificate Chain Details	+	+	+
Certificates Operations			
Open a standalone Certificate	+	+	+
View Certificate Details	+	+	+
Display multiple certificates including certificate chains	+	+	+
View Public Key Details	+	+	+
View PEM Representation for a Certificate	+	+	+
View ASN.1 for a Certificate	+	+	+
View ASN.1 for a Certificate Extension	+	+	+
View Certificate Extensions	+	+	+
Obtain the Revocation Status	+	+	+
Test Certificates on Given Protocol	+	-	+
View the CRL associated to a certificate	+	+	+
Sign and Verify			
Verify Signatures for JAR/APK Files	+	+	+
Verify Signatures for XML Files	+	-	+
Verify Signatures for PDF Files	+	-	+
Sign JAR/APK Files	+	+	+
Sign XML Files	+	-	+
Sign PDF Files	+	-	+
Export Options			
Export Certificate	+	+	+
Export Certificate Chain	+	+	+
Export Key Pair	+	+	+
Export Private Key	+	+	+
Export Public Key	+	+	+

Feature	Trial License	Standard License	Professional License
TrustStores Management			
Set / Remove TrustStores at runtime	+	+	+
Configure Trust Path validation options at runtime	+	+	+
Certificate Authority functions			
Certificate Signing made easier	+	+	+
Sign CSR Files	+	+	+
Generate CSR Files	+	+	+
Open CSR Files	+	+	+
Open CRL Files	+	+	+
Import CA Reply	+	+	+
Trust verification when Importing CA Reply	+	+	+
Certificate chain management: append and remove certificate	+	+	+