Using Credential Store to securely store authentication, connection and other parameters

Introduction

- The Credential Store (CS) allows sensitive data to be encrypted and stored securely and independently of the application(s) such as YADE and the JobScheduler YADE JITL Jobs that use this data.
- The advantage of using a CS is that the CS stores sensitive information such as credentials in a standardized, secure and fully encrypted database and sensitive authentication information is not exposed in use. Applications access the CS database by using password, encryption-key file or a combination of both.
- The CS requires the use of a standard open database format (`.kdb` or `.kdbx`), which allows the use of graphical and API interfaces across the most relevant operating systems.

Scope

This article is in two parts:

- The first part describes the use of the Credential Store in a relatively simple example file transfer configuration. The configuration described can be used to transfer files from a live server and a download containing the configuration file is available so that users can get a working Credential Store up, running and tested as easily as possible.
- The second part of the article describes Credential Store configuration elements not covered by the example configuration. In addition, the use of the Credential Store in with the JobScheduler YADE JITL jobs is described

This article does not attempt to provide a step-by-step description of file transfer configuration, which is available elsewhere in this article, for example, in the tutorials for YADE and the JobScheduler.

Example Description

The example presented in this article illustrates the configuration and use of the Credential Store as part of a simple file transfer operation - downloading files from an online server to the user's local file system.

The configuration is stored in an XML settings file and includes the elements specifying the Credential store and the file transfer as a whole. This settings file can be used by both the YADE Client and by the YADE JITL jobs that are provided with the JobScheduler.

The example described in this article is based on the simple file transfer example that is described in detail in The YADE Client Command Line Interface - Tutorial 1 - Getting Started article. This tutorial describes the configuration required to download a number of files from an online server provided by the SOS GmbH and save these files on the user's local file system. Using this server together with the downloaded configuration file means that users can get a working example up and running with a minimum of effort.

In the current example, the Credential Store is to store configuration information for the online server - i.e. for the file transfer source. The principle described can be equally well used for the configuration of multiple file transfer source, target, proxy and jump-host servers and for the other file transfer protocols that can be used by the YADE Client.

Note that a YADE Client or JobScheduler Master is required to carry out the example file transfer. Instructions for installing and configuring the YADE Client can be found in the YADE - Tutorials article. Instructions for installing and configuring the JobScheduler can be found in the JobScheduler Master - Installation Guide series of articles.

The example configuration file can be downloaded here:

- sos-berlin_demo_2_local.xml.

Configuration Procedure for the Example

Installing the Credential Store and configuring the KeePass database

KeePass 2, which is just one of the applications available for creating and configuring `.kdb` or `.kdbx` databases, has been used in the current article to implement the Credential Store database and is used in the screenshots. The installation and use of KeePass 2 is described on the KeePass Web Site.

Feature Availability

FEATURE AVAILABILITY STARTING FROM RELEASE 1.12.2

The full range of Credential Store features such as secure, compliant and password-free use of the Credential Store as well as compatibility with KeePass `.kdb` databases requires the YADE Client in version 1.12.2 or newer.

Database Configuration

Credential Store databases are stored as either `.kdb` or `.kdbx` files on the file system.

The database included with the download files was configured as follows:

- Path: `.\jade_demo\keepass\demo_credential_store.kdbx`
- Master Password: `sos`

Note that a Master Key file can be used to provide further protection for the database, either instead of or in addition to the Master Password. This is described in the Advanced Configuration section of this article below but has not been configured for the download database.

Adding connection information to the Credential Store

Connection configuration information is stored in the Credential Store as an *Entry* and Entries can be organized into Groups.

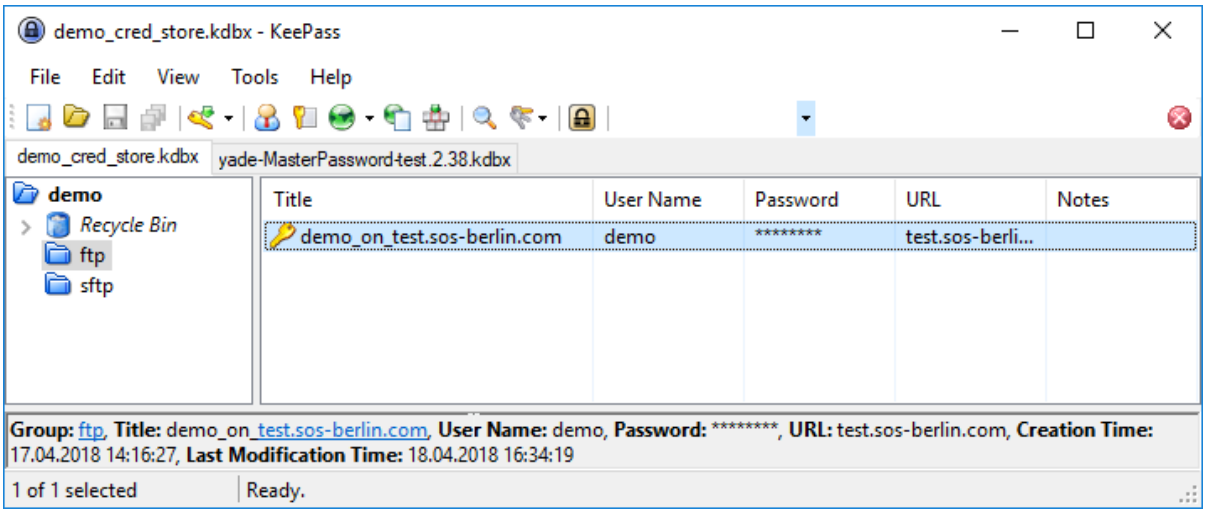
The following fields are available for storing information in a CS *Entry*:

- **Title**: The identifier for the Entry, this could be a string containing, for example, the host name/server name.
- **User name**: The user identification of a user account who is authenticated for the operation.
- **Password**: Assigned password for a user account or passphrase for a private key.
- **URL**: The host name/server name or IP address of the server.
- **Notes**: This block can be used to specify additional parameters for the file transfer.
- **File Attachment** & **String Fields**: Files such as PGP or SSH private keys can be stored as attachments.
    - A first file is specified as an *attachment* .
    - Further files are specified using *String field* parameter / value pairs.
      YADE will retrieve the contents of an attached file at run-time - intermediate or temporary files are not created when reading attachments.
      Note that *Attachments* and *String Fields* are specified in the KeePass GUI via the *AdvancedEdit Entry* tab.

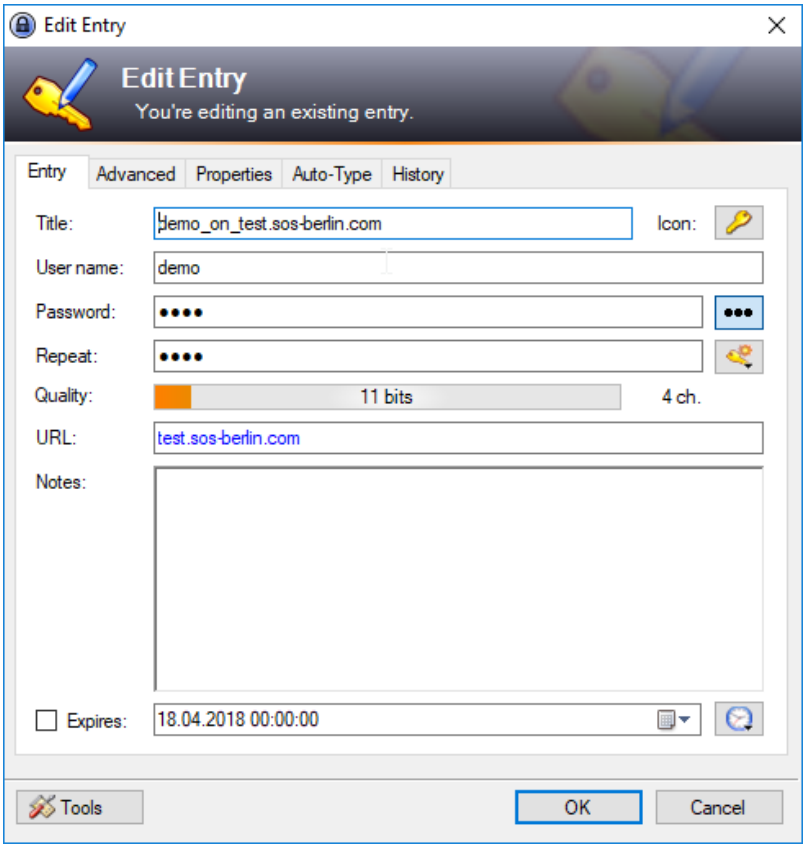
The following information needs to be specified for the current example:

- Groups: `demo,ftp` (optional)
- Title: `demo_on_test.sos-berlin.com`
- User name: `demo`
- Password: `demo`
- URL: `test.sos-berlin.com` (Alternatively, the IP address could have been specified here.)

The following screenshot shows that two *Groups* have been configured for the current example, named "demo" and "ftp", along with the *Entry* "demo_on_test.sos-berlin.com".



The next screenshot shows the configuration of the parameters in the "demo_on_test.sos-berlin.com" *Entry*.



Integrating the Credential Store in a File Transfer Configuration

The use of the Credential Store is specified in YADE Client file transfer configuration files, which are written in XML. We recommend using the SOS XML Editor to edit these files. Instructions for downloading, installing and using the XML Editor are linked from this page.

In the remainder of the current article, it is assumed that readers have made themselves familiar with the organization of the YADE Client file transfer configurations into *Profiles* and *Fragments*. This is described in The YADE Client Command Line Interface - Tutorial 1 - Getting Started article mentioned above.

The current example uses the XML configuration from the Getting Started tutorial article above and describes the necessary configuration elements required to move the sensitive information such as user name and password from the XML file to the Credential Store. Users wishing implement the current example should download the tutorial file transfer configuration file linked above and open it in their XML Editor, where they can then add the necessary configuration information.

The information required to use the Credential Store falls into two "areas":

- Information about the Credential Store itself (location, how to access its contents, etc.).
  This information is configured in the XML file in a *CredentialStoreFragment*.
- Information about how the information in the Credential Store (server address, password, etc.) is to be accessed for the file transfer.
  This information is stored as a string in each of the relevant XML *ProtocolFragment* child elements (Hostname, Account. etc.).

In addition, the *ProtocolFragment* element has a reference specifying that the Credential Store is to be used.

Specifying the Credential Store

The following list shows the organization of the XML elements required to specify the Credential Store. These elements and their attributes are shown in full in the XML Editor screenshot below.

- `Fragments`
  - `ProtocolFragments`
    - `FTPFragment name=`*"ftp_demo_sos-berlin_cs"*
      - ....
      - `CredentialStoreFragmentRef` ref =*"ftp_demo"*
  - `CredentialStoreFragments`
    - `CredentialStoreFragment` name =*"ftp_demo"*
      - `CSFile file path` *%USERPROFILE%\jade_demo*....
      - `CSAuthentication`
        - `PasswordAuthentication`
          - etc.
      - `CSEntryPath`
- `Profiles`
  - etc.

Addressing the information in the Credential Store

Parameters stored in a Credential Store database *Entry* can be addressed in the *CredentialStoreFragment* XML element as follows:

- The *CSEntryPath* element is used to specify the base path in the Credential Store database to the *Entry*.
  In the current example this would be set to:
  - `demo/ftp/demo_on_test.sos-berlin.com`
    where `demo` and `ftp` are (optional) *Group* names, as already mentioned, and `demo_on_test.sos-berlin.com` is the *Title* of the KeePass database *Entry*.

The Credential Store *Entry* parameters are addressed using one of the following syntaxes:

- relative:
  - `cs://@parameter_name`, where the *parameter_name* is the name of the relevant parameter specified for the *Entry* - for example, url and the *CSEntryPath* element is filled as shown above
- fully specified:
  - `cs://demo/ftp/demo_on_test.sos-berlin.com@parameter_name`, and where the *CSEntryPath* element, which is a required element, is left blank

The following parameters are fully specified in the Credential Store in the current example:

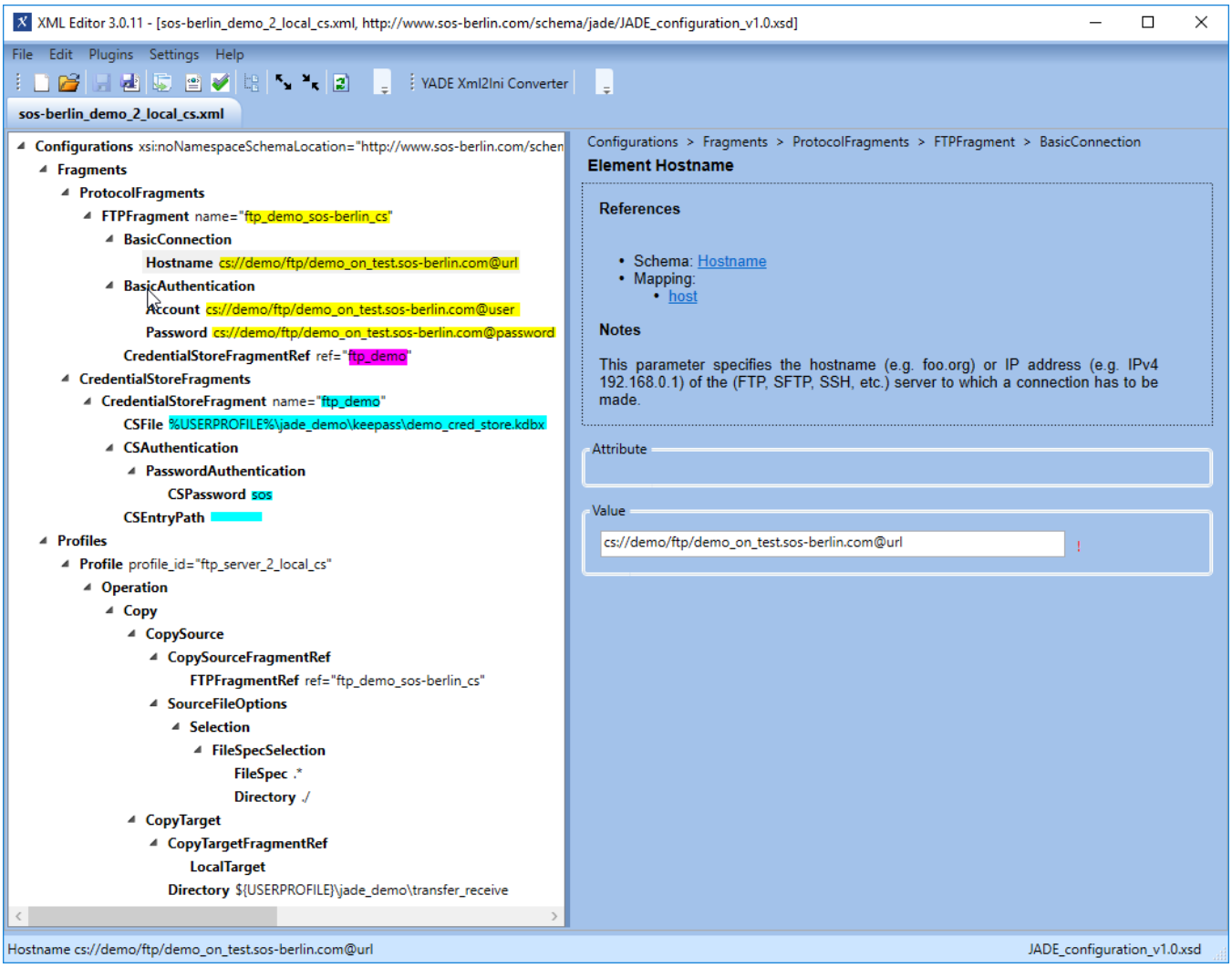
- Hostname: `cs://demo/ftp/demo_on_test.sos-berlin.com@url` (where `@url` specifies the *URL* element stored in the database)
- Account: `cs://demo/ftp/demo_on_test.sos-berlin.com@user` (where `@user` specifies the *User name* element stored in the database)
- Password: `cs://demo/ftp/demo_on_test.sos-berlin.com@password` (where `@password` specifies the *Password* element stored in the database)

Note that a full list of parameters is described in the Adding an *Entry* to the Credential Store section above.

Configuration in the XML Editor

> **Tip**
>
> The XML Editor includes up-to-date documentation for elements as can be seen in the screenshot below, which shows the documentation for the Hostname element.

The parts of the XML configuration relevant to the use of the Credential Store are shown in the following screenshot of the configuration for the current example, with parameter values highlighted according to their function:



The Transfer Target Directory

The screenshot above shows a *CopyTarget.Directory* parameter for a Windows environment:

- `${USERPROFILE}\jade_demo\transfer_receive`

Depending on their operating system, users may find it necessary to modify this attribute before running the example.

XML Listing

The following code block can be opened to show the full XML configuration for the example:

Simple Example Listing

```
<?xml version="1.0" encoding="utf-8"?>
<Configurations xsi:noNamespaceSchemaLocation="http://www.sos-berlin.com/schema/jade/JADE_configuration_v1.0.
xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Fragments>
    <ProtocolFragments>
      <FTPFragment name="ftp_demo_sos-berlin_cs">
        <BasicConnection>
          <Hostname><![CDATA[cs://demo/ftp/demo_on_test.sos-berlin.com@url]]></Hostname>
        </BasicConnection>
        <BasicAuthentication>
          <Account><![CDATA[cs://demo/ftp/demo_on_test.sos-berlin.com@user]]></Account>
          <Password><![CDATA[cs://demo/ftp/demo_on_test.sos-berlin.com@password]]></Password>
        </BasicAuthentication>
        <CredentialStoreFragmentRef ref="ftp_demo" />
      </FTPFragment>
    </ProtocolFragments>
    <CredentialStoreFragments>
      <CredentialStoreFragment name="ftp_demo">
        <CSFile><![CDATA[%USERPROFILE%\jade_demo\keepass\demo_cred_store.kdbx]]></CSFile>
        <CSAuthentication>
          <PasswordAuthentication>
            <CSPassword><![CDATA[sos]]></CSPassword>
          </PasswordAuthentication>
        </CSAuthentication>
        <CSEntryPath />
      </CredentialStoreFragment>
    </CredentialStoreFragments>
  </Fragments>
  <Profiles>
    <Profile profile_id="ftp_server_2_local_cs">
      <Operation>
        <Copy>
          <CopySource>
            <CopySourceFragmentRef>
              <FTPFragmentRef ref="ftp_demo_sos-berlin_cs" />
            </CopySourceFragmentRef>
            <SourceFileOptions>
              <Selection>
                <FileSpecSelection>
                  <FileSpec><![CDATA[.*]]></FileSpec>
                  <Directory><![CDATA[./]]></Directory>
                </FileSpecSelection>
              </Selection>
            </SourceFileOptions>
          </CopySource>
          <CopyTarget>
            <CopyTargetFragmentRef>
              <LocalTarget />
            </CopyTargetFragmentRef>
            <Directory><![CDATA[${USERPROFILE}\jade_demo\transfer_receive]]></Directory>
          </CopyTarget>
        </Copy>
      </Operation>
    </Profile>
  </Profiles>
</Configurations>
```

Running the YADE Client with the Credential Store

The use of the Credential Store is contained within the settings file and is not exposed when calling the YADE Client. For example, on Windows systems, the YADE Client is called for the current example using:

Call on Windows systems

```
C:\Program Files\sos-berlin.com\jade\client\bin>jade.cmd -settings="%USERPROFILE%\jade_demo\sos-
berlin_demo_2_local_cs.xml" -profile="ftp_server_2_local_cs"
```

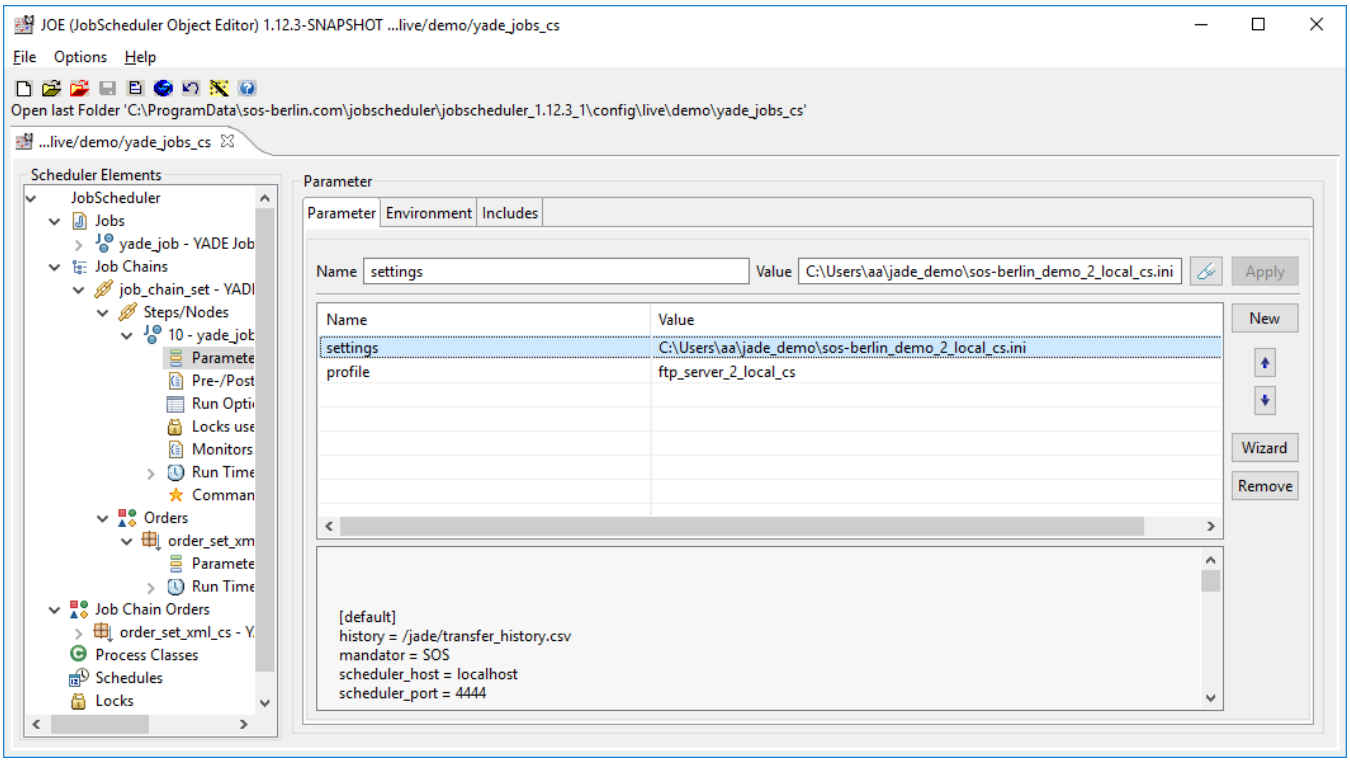
After the YADE command has finished execution the number of files transferred can be read from the log file.

Note that the log files neither indicate that a credential store has been use for the transfer nor reveal any passwords.

Using the Credential Store with the JobScheduler JITL YADE jobs

Settings XML files such as the `sos-berlin_demo_2_local_cs.xml` file which was used to configure the example described above can be used for JobScheduler JITL jobs. Here, only two parameters are needed to run the YADE JITL job (*settings* and *profile*) as can be seen in the next screenshot.

Note that we recommend that these parameters are set manually using the *New* button shown in the screenshot below and not using the *Wizard* button.



Note also that while the YADE Client runs under the current user account, the JobScheduler generally runs under a predefined account. This means that while paths in the configuration file can use parameters such as *%USERPROFILE%* when the configuration file is being used with the YADE Client, it is generally necessary to use absolute paths when the configuration file is to be used for JITL jobs.

Advanced Configuration

Key File Authentication

Key file authentication can be used for the Credential Store either alone or together with the password authentication described in the example above.

This option allows the the Credential Store to be used completely securely, yet without passwords, if required.

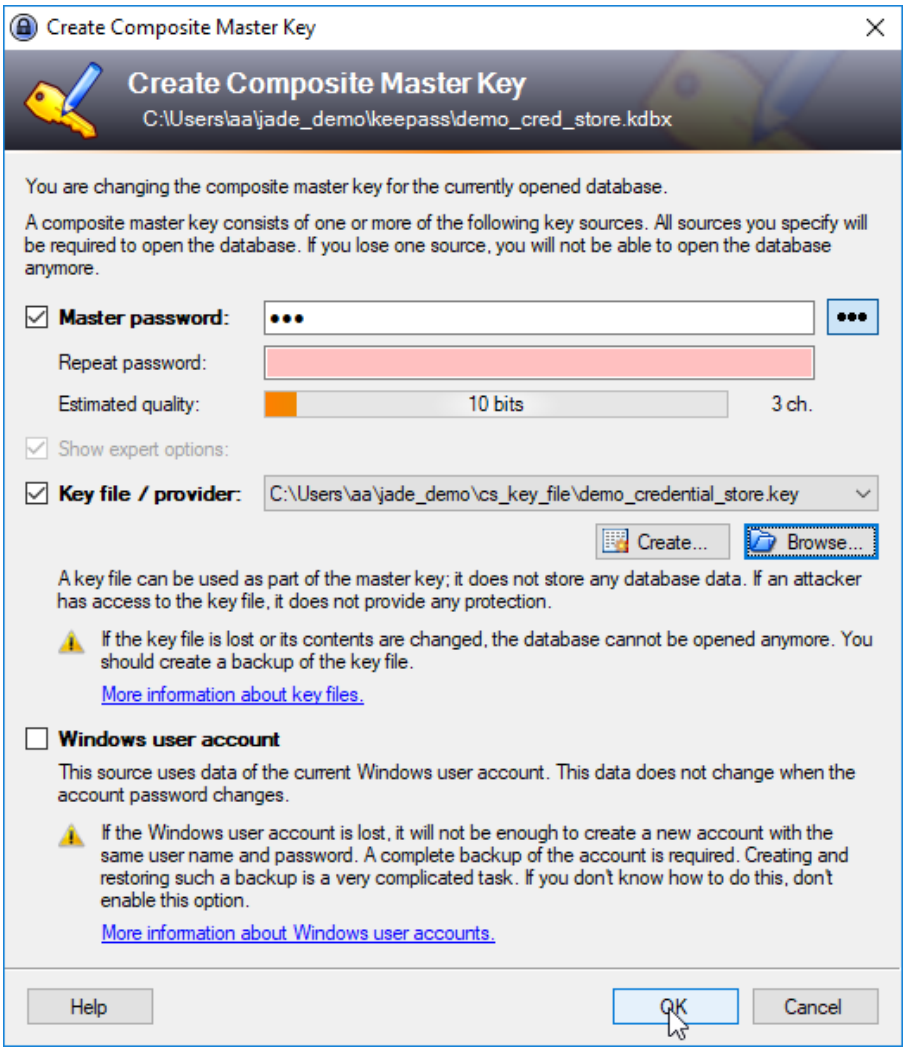
Key file authentication has to be configured for the Credential Store and in the XML settings file.

Configuring key file authentication in the Credential Store

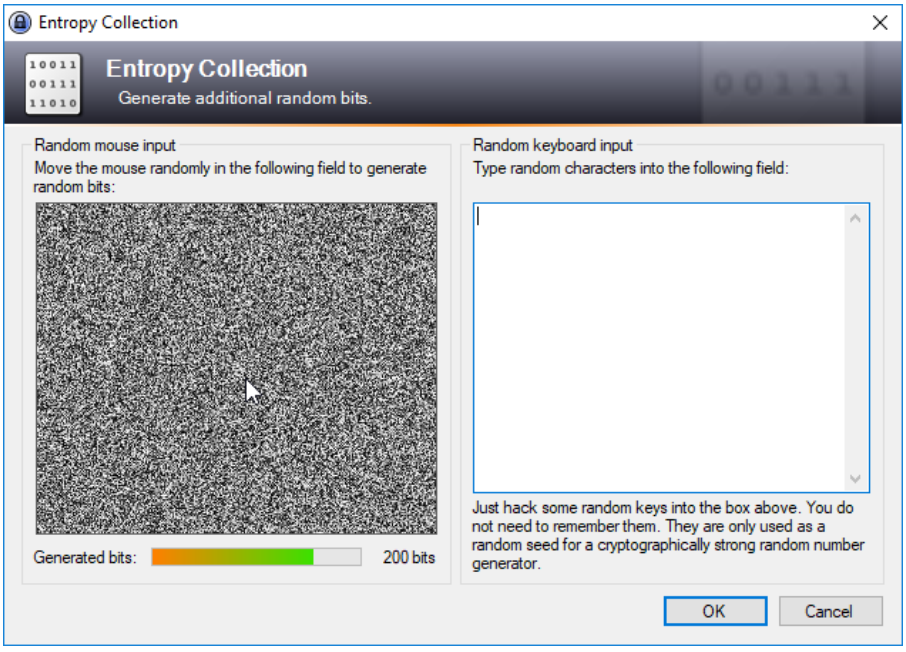
KeePass provides a Create Composite Master Key function that is reached with the *Files / Master Key...* menu item. screen

The Create Composite Master Key function is shown in the following screenshot (Note that the Show expert options checkbox has to be selected first.):

Note also that the *Master Password* checkbox should not be selected if key file authentication is to be used without a master password.



The entropy of the key generated can be increased as part of the key creation procedure. This is done as part of the key generation procedure in the interface shown in the next screenshot.



For the purpose of this article the key has been saved in the `jade_demo` folder used for the download example.

The next section describes the configuration of the XML settings file to include a reference to this file.

Configuring key file authentication in the XML settings file

Key file authentication is configured in the XML settings file by specifying a *KeyFileAuthentication* element as a child of the *CSAuthentication* element in the Credential Store fragment.

The key file element can be added either instead of or alongside a password authentication element as required.

This is shown in the following list:

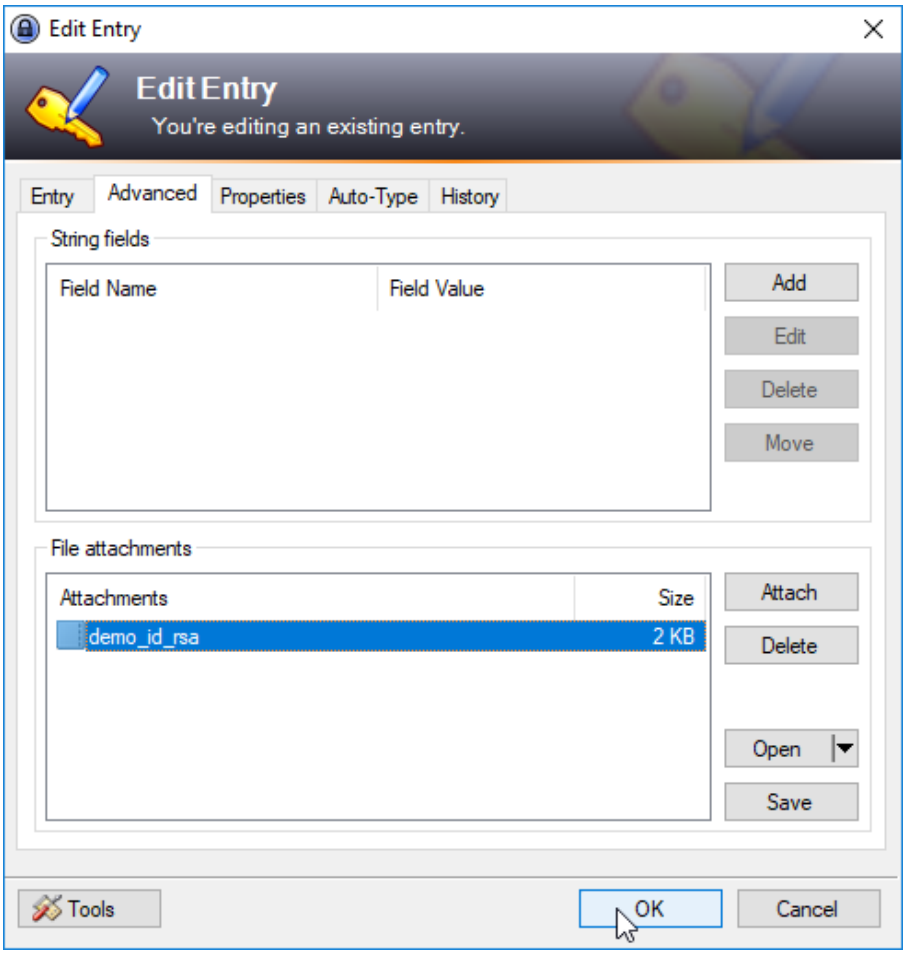
- `CredentialStoreFragments`
  - `CredentialStoreFragment` name = *"ftp_demo"*
    - `CSFile file path` *%USERPROFILE% \jade_demo....*
    - `CSAuthentication`
      - `PasswordAuthentication`
        - `.CSPassword` *myPassword*
      - `KeyFileAuthentication`
        - `CSKeyFile` *%USERPROFILE%\jade_demo\cs_key_file\demo_credential_store.key*
    - `CSEntryPath`

Connection authentication key files

The Credential Store can be used to store RSA and similar connection authentication key files. These are stored in the Credential Store database as *attachments*.

Configuring authentication key files in the Credential Store

Attachments are added to the Credential Store in KeePass in the *File Attachments* section of the Advanced tab as shown in the screenshot below. Note that only one attachment can be added for each Credential Store *Entry* :



Configuring authentication key files in the XML settings file

A first attachment for, for example, SSH would be configured in the XML settings file by specifying an *AuthenticationFile* element in the *SSHAuthentication* element.

The key file element can be added either instead of or alongside a password authentication element as required.

This following list shows the configured of the SFTP Fragment required to carry out the download from the *test.sos-berlin.com* SFTP/FTP server that was used for the simple example described above: The *AuthenticationFile* element that specifies the Attachment in the Credential Store *Entry* is specified in the same way as the *Hostname* and other elements described in the example above.

- `SFTPFragment` name =*"sftp_demo_sos-berlin_cs"*
  - `BasicConnection`
    - `Hostname` *cs://demo/sftp/demo_on_test.sos-berlin.com@attachment*
  - `SSHAuthentication`
    - `Account`
    - `AuthenticationMethodPublicKey`
      - `AuthenticationFile` *cs://demo/sftp/demo_on_test.sos-berlin.com@attachment*
      - `Passphrase` *myPassPhrase*
    - `CredentialStoreFragmentRef`*ref="ftp_demo"*

Note that this list also shows the use of a *Passphrase* element for the AuthenticationFile element. This is not required for authentication with the test.sos-berlin.com SFTP server but is provided as an illustration.

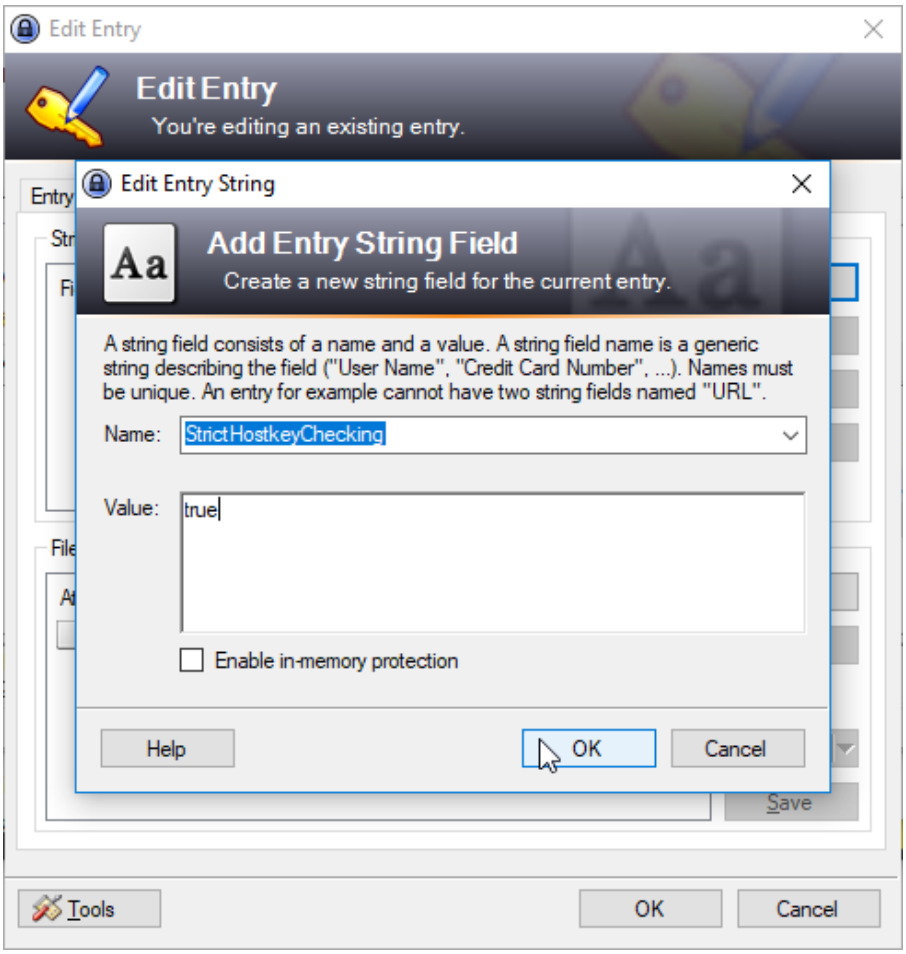
Passphrase elements are stored in the Credential Store as Notes.

Custom Parameters

Custom parameters allow XML *ProtocolFragment* elements such as *StrictHostkeyChecking* that do not belong to the standard keepass.GUI fields such as URL to be specified.

Configuring custom parameters in the Credential Store

Custom parameters are set as string name / value pairs. In the KeePass 2 GUI these are set by opening the *Edit* window for an *Entry* using the *Add* function in the *String Fields* section of the *Advanced* tab. .This is shown in the next screenshot:



Configuring custom parameters in the XML settings file

Custom parameters are configured in the XML settings file by ..

- `SFTPFragment` name = *"sftp_tokyo_japan.sos-berlin.com"*
    - `BasicConnection`
    - `SSHAuthentication`
    - `CredentialStoreFragmentRef` ref= *"demo_credential_store"*
    - `StrictHostkeyChecking` *"false"*

Note that custom parameters can only be used to set parameters in *ProtocolFragments* - they cannot be used to set *ProfileFragments* parameters such as *Recursive*.

See Also

- YADE User Manual - Credential Store

References

Support of the features described in this article is subject to the following issues:

Issues implemented with Release 1.12.1:

- **YADE-462** - Getting issue details... STATUS
- **YADE-464** - Getting issue details... STATUS
- **YADE-481** - Getting issue details... STATUS
- **YADE-482** - Getting issue details... STATUS
- **YADE-485** - Getting issue details... STATUS
- **YADE-487** - Getting issue details... STATUS
- **YADE-491** - Getting issue details... STATUS

Issues implemented with Release 1.12.2:

- **YADE-493** - Getting issue details... STATUS
- **YADE-498** - Getting issue details... STATUS
- **YADE-499** - Getting issue details... STATUS