Using a Credential Store for Jobs

Introduction

Sensitive information that is required in job scripts can be stored in a Credential Store and retrieved at run-time.

This feature is similar to the method used by the YADE file transfer job (and command line utility) to store information such as credentials in the YADE Credential Store.

FEATURE AVAILABILITY STARTING FROM RELEASE 1.12.6

Overview

- Starting Point
  - Users frequently operate jobs that require credentials, e.g. to access a database, a file transfer SFTP server etc.
  - Such jobs are implemented as simple shell jobs or by use of the API Interface.
- Security Considerations
  - Sensitive information in jobs should not be hard-coded, not be used from parameters and should not be disclosed, e.g. written to log files, therefore the solution does not store sensitive information in parameters.
  - Instead a run-time interface is offered that allows to retrieve sensitive information from a credential store. References to credential store entries can safely be stored with parameter values.
- Credential Store
  - A credential store allows the secure storage and retrieval of credentials for authentication, as well as connection and other parameters, for a detailed features and supported products see YADE Credential Store.
- Solution Outline
  - Access to the credential store is provided by a Java class that can be loaded from shell jobs and from API jobs implemented e.g. for JavaScript, PowerShell etc.
  - The Java class is parameterized with the path that identifies the requested entries from the credential store.
  - This solution can be operated with JobScheduler Master and with Agents.

Usage

- The `SOSKeePassDatabase` Java class can be invoked
  - in a Shell Job by calling the `java` command line utility with the class name:
    - If the class is executed successfully:
      - return code = *0*, output is sent to *stdout*
    - If execution of the class ends in error:
      - return code = *99*, exception output is sent to *stderr*
  - in a JavaScript Job
    - by directly instantiating the Java class from JavaScript.
    - if execution of the class ends in error then an exception is raised.
  - in Powershell Jobs (for use with Agents only) by calling the *java* command line utility with the class name:
    - A return code is provided similar to Shell jobs.
- The class can be invoked
  - from the command line like this:
    - `java com.sos.keepass.SOSKeePassDatabase "cs://server/SFTP/homer.sos@password?file=credential_store.kdbx"`
    - When invoking the class then the path to the entry in the credential store is specified.
  - by use of a script provided with JobScheduler e.g. for Unix environments:
    - `$SCHEDULER_HOME/bin/jobscheduler_credential_value.sh "cs://server/SFTP/homer.sos@password?file=credential_store.kdbx"`
    - The script hides the call to the `java` command line utility.

Syntax

A call to the `SOSKeePassDatabase` class syntactically uses a single parameter string that holds a URI and a number of query parameters:

URI

- `cs://<entry_path>@<property_name>` - required
  - The URI based syntax includes the protocol `cs://`
  - followed by the *<entry_path>* that specifies the directory structure and entry name in the credentials store file.
  - followed by the @ character
  - followed by the *<property_name>* that should be retrieved:
    - frequently used properties include credential store field names such as `title, user, password`. Custom field names are supported.
    - for detailed explanations of available properties see the Using Credential Store to securely store authentication, connection and other parameters article.

Query Parameters

- `file` - required
  - the path to the credential store database file. This file can be stored anywhere in the file system.
  - The path can be specified either relatively or absolutely. As a path separator both forward slashes and backslashes can be used. For example:
    - *cs://databases/mysql_localhost@password?file=config/credential_store.kdbx*
    - *cs://databases/mysql_localhost@password?file=C:/jobscheduler/data/config/credential_store.kdbx*
  - Relative values with a Master are with respect to the Master's *SCHEDULER_DATA* directory.
  - Relative values with an Agent are with respect to the *SCHEDULER_HOME* (install) directory.
- `password` - optional
  - the password for the credential store database file.
  - It is recommended not to use this parameter and instead to use a `key_file` to access the credential store.
- `key_file` - optional, default: *<credential_store_database_filename_without_extension>.key*
  - a key file for the credential store database file.
  - If this parameter is set:
    - this path can be specified either relatively or absolutely. See the explanations for the `file` parameter.
    - An exception will be thrown if the *.key* file is not found.
  - If this parameter is not set:
    - a *<credential_store_database_filename_without_extension>.key* file such as *credential_store.kdbx -> credential_store.key* will be sought in the directory where the credential store database file is located.
      - The *.key* file will be used if it is found.
      - An exception will be thrown if a *.key* file is not found and the `password` parameter is not used.
- `ignore_expired` - optional, default: *0*
  - *ignore_expired=0* - an exception is thrown when the entry has expired.
  - *ignore_expired=1* - expiring of an entry is ignored.
- `attachment` - optional
  - *attachment=1* - a *attachment* field is read.

Availability starting from release 1.12.10.

- `create_entry` - optional, default: `0`
  - `create_entry=0` - an exception is thrown when the entry not found.
  - `create_entry=1` - creates an entry if it does not exist.
    - an exception is thrown when the root group of the `cs://<entry_path>` URI does not match with the database root group.
    - creates the full path to the entry if it does not exist.
- `set_property` - optional
  - set value of a string property:
    - `cs://<entry_path>@<property_name>?...&set_property=<value>`
      - *<property_name>* property exists:
        - property value will be updated.
      - *<property_name>* property does not exist:
        - *<property_name>* property will be created with the given value.
  - set value of a binary property:
    - `cs://<entry_path>@<property_name>?...&set_property=<file path>&attachment=1`
      - *<property_name>* property exists:
        - property value will be updated with the `<file path>` file content.
      - *<property_name>* property does not exist:
        - *<property_name>* binary property will be created with the `<file path>` file content.
    - `cs://<entry_path>@attachment?...&set_property=<file path>`
      - creates a new binary property with the given file content if it does not exist (property name is a file name), or updates the binary property value with the given file content.
- `stdout_on_set_binary_property` - optional, default: `0`
  - `stdout_on_set_binary_property=0`
    - does not create the output of the binary property set with the `set_property` to the STDOUT.
  - `stdout_on_set_binary_property=1`
    - creates the output of the binary property set with the `set_property` to the STDOUT.

Examples

Shell Job Example for Master/Agent (Windows)

Shell Job Example (Windows)

```
<job  order="no" stop_on_error="no" stderr_log_level="error">
    <script  language="shell">
        <![CDATA[
@echo off
rem Sample 1 %SCHEDULER_CREDENTIAL_VALUE%
call "%SCHEDULER_HOME%/bin/jobscheduler_credential_value.cmd" "cs://server/SFTP/homer.sos@password?file=%
SCHEDULER_DATA%/config/credential_store.kdbx"
if ERRORLEVEL 1 exit /b %ERRORLEVEL%
echo %SCHEDULER_CREDENTIAL_VALUE%

rem Sample 2 stdout
call "%SCHEDULER_HOME%/bin/jobscheduler_credential_value.cmd" "cs://server/SFTP/homer.sos@password?file=%
SCHEDULER_DATA%/config/credential_store.kdbx" stdout
if ERRORLEVEL 1 exit /b %ERRORLEVEL%

 ]]>
    </script>
    <run_time />
</job>
```

Explanations

- The <job> element makes use of the `stderr_log_level` attribute to cause the job to fail in case of errors being reported to stderr, e.g. from the Java class `SOSKeePassDatabase`.
- The script `jobscheduler_credential_value.cmd` is available from the Master's or Agent's `./bin` directory. The environment variable `%SCHEDULER_HOME%` is automatically provided.
- The credential store file is located in the `./config` directory of a Master or Agent. The environment variable `%SCHEDULER_DATA%` is automatically provided. However, the credential store file can be located anywhere in the file system.
- The credential value to be retrieved is returned by the built-in environment variable `%SCHEDULER_CREDENTIAL_VALUE%` by the script `jobscheduler_credential_value.cmd` (see `Sample 1 %SCHEDULER_CREDENTIAL_VALUE%`).
    - `%SCHEDULER_CREDENTIAL_VALUE%` contains the last row of the possible multiple rows value.
- The additional parameter `stdout` (see `Sample 2 stdout`) controls that all outputs are forwarded to stdout.

Shell Job Example for Master/Agent (Unix)

Shell Job Example (Unix)

```
<job  order="no" stop_on_error="no" stderr_log_level="error">
    <script  language="shell">
        <![CDATA[
SCHEDULER_CREDENTIAL_VALUE=`"$SCHEDULER_HOME/bin/jobscheduler_credential_value.sh" "cs://server/SFTP/homer.
sos@password?file=$SCHEDULER_DATA/config/credential_store.kdbx"`
RETURNCODE=$?
if [ $RETURNCODE -ne 0 ]
  then
        exit $RETURNCODE
fi
echo $SCHEDULER_CREDENTIAL_VALUE
       ]]>
    </script>

    <run_time />
</job>
```

Explanations

- The <job> element makes use of the `stderr_log_level` attribute to cause the job to fail in case of errors being reported to stderr, e.g. from the Java class `SOSKeePassDatabase`.
- The script `jobscheduler_credential_value.sh` is available from the Master's or Agent's `./bin` directory. The environment variable `$SCHEDULER_HOME` is automatically provided.
- The credential store file is located in the `./config` directory of a Master or Agent. The environment variable `$SCHEDULER_DATA` is automatically provided. However, the credential store file can be located anywhere in the file system.
- The credential value to be retrieved is returned to stdout by the script `jobscheduler_credential_value.sh`. The above example makes use of the environment variable `$SCHEDULER_CREDENTIAL_VALUE` that holds output to stdout of the script, i.e. receives the credential value.

JavaScript Job Example for Master/Agent (all platforms)

JavaScript Job Example

```
<job  order="no" stop_on_error="no">
  <script  language="java:javascript"><![CDATA[
                function getCredentialStoreProperty( uri ) {
                        try {
                                return Packages.com.sos.keepass.SOSKeePassDatabase.getProperty( uri );
                        }
                        catch (e) {
                                throw new Error( "can't get property: " + e.message );
                        }
                }

                function exportCredentialStoreAttachment2File( uri, targetFile ) {
                        var fos                         = null;
                        try {
                                var data        = Packages.com.sos.keepass.SOSKeePassDatabase.getBinaryProperty
( uri );
                                fos                 = new Packages.java.io.FileOutputStream( targetFile );
                                fos.write( data );
                        } catch (e) {
                                throw new Error( "[" + targetFile + "] can't write attachment to file: " + e.
message );
                        }
                        finally {
                                if ( fos !== null ) {
                                        fos.close();
                                }
                        }
                }

                function spooler_process() {
            // find credential store from a Master's ./config directory
                        var file                = "config/credential_store.kdbx";

                        spooler_log.info( "--- get string property ---" );
                        var property         = "server/SFTP/homer.sos@password";
                        var uri                 = "cs://" + property + "?file=" + file;
                        var val                 = getCredentialStoreProperty( uri );
                    spooler_log.info( "[" + property + "]=" + val );

                        spooler_log.info( "--- get binary property as string ---" );
                        property                = "server/SFTP/homer.sos@homer.privat.dsa";
                        uri                        = "cs://" + property + "?file=" + file + "&attachment=1";
                        val                                = getCredentialStoreProperty( uri );
                    spooler_log.info( "[" + property + "]=" + val );

                        spooler_log.info( "--- get binary property as byte array and write to file ---" );
                        property                = "server/SFTP/homer.sos@homer.privat.dsa";
                        uri                        = "cs://" + property + "?file=" + file;
                        var targetFile         = "D:/my_homer.privat.dsa";
                        exportCredentialStoreAttachment2File( uri, targetFile );
                        spooler_log.info( "[" + property + "] written to " + targetFile );

                    return false;
                }
        ]]></script>
    <run_time />
</job>
```

Explanations

- Two methods can be used:
  - `com.sos.keepass.SOSKeePassDatabase.getProperty( uri )`
    - is used to read a value from a credential store entry, e.g. an account, password etc.
    - returns a string that holds the requested value
  - `com.sos.keepass.SOSKeePassDatabase.getBinaryProperty( uri )`

- is used to read a file attachment from a credential store entry, e.g. if the entry is attached a private key file.
- returns a byte array from a credential store entry that specifies an attachment

PowerShell Job Example (Agent for Windows)

The recommended way is to call the same script `scheduler_credential_value.cmd` as explained above for shell jobs:

PowerShell Job Example (call to script)

```
<job  order="no" stop_on_error="no" stderr_log_level="error" process_class="/Agent">
    <script  language="powershell"><![CDATA[
               $file                = "$env:SCHEDULER_DATA/config/credential_store.kdbx";
               $property         = "server/SFTP/homer.sos@password";
               $uri                 = "cs://" + $property + "?file=" + $file;

               $val                 = Invoke-Expression "&`"$env:
SCHEDULER_HOME\bin\jobscheduler_credential_value.cmd`" '`"$uri`"' stdout"
               $spooler_log.info( "[" + $property + "]=" + $val);
    ]]></script>
    <run_time />
</job>
```

Alternatively the Java class `SOSKeePassDatabase` can be invoked directly.

PowerShell Job Example (java.exe from the Windows PATH variable)

```
 <job  order="no" stop_on_error="no" stderr_log_level="error" process_class="/Agent">
    <script  language="powershell"><![CDATA[
               $file                = "$env:SCHEDULER_DATA/config/credential_store.kdbx";
               $property         = "server/SFTP/homer.sos@password";
               $uri                 = "cs://" + $property + "?file=" + $file;

               $val                 = java.exe com.sos.keepass.SOSKeePassDatabase $uri
               $spooler_log.info( "[" + $property + "]=" + $val);
    ]]></script>
    <run_time />
</job>
```

Explanations

- The `<job>` element makes use of the `stderr_log_level` attribute to cause the job to fail in case of errors being reported to stderr, e.g. from the Java class `SOSKeePassDatabase`.

PowerShell Job Example (java.exe from the built-in environment variable %JAVABIN%)

```
 <job  order="no" stop_on_error="no" stderr_log_level="error" process_class="/Agent">
    <script  language="powershell"><![CDATA[
               $file                = "$env:SCHEDULER_DATA/config/credential_store.kdbx";
               $property         = "server/SFTP/homer.sos@password";
               $uri                 = "cs://" + $property + "?file=" + $file;

               $val                 = Invoke-Expression "&`"${env:JAVABIN}`" com.sos.keepass.
SOSKeePassDatabase '`"$uri`"'"
               $spooler_log.info( "[" + $property + "]=" + $val);
    ]]></script>
    <run_time />
</job>
```

Explanations

- The `<job>` element makes use of the `stderr_log_level` attribute to cause the job to fail in case of errors being reported to stderr, e.g. from the Java class `SOSKeePassDatabase`.

- The environment variable `%JAVABIN%` is automatically provided on an Agent.

PowerShell Job Example (java.exe from an user variable)

```
<job  order="no" stop_on_error="no" stderr_log_level="error" process_class="/Agent">
   <script  language="powershell"><![CDATA[
               $file                = "$env:SCHEDULER_DATA/config/credential_store.kdbx";
               $property         = "server/SFTP/homer.sos@password";
               $uri                 = "cs://" + $property + "?file=" + $file;
               $javaExe    = "C:\Program Files\Java\jre1.8.0_171\bin\java.exe"

       $val        = Invoke-Expression "&`"$javaExe`" com.sos.keepass.SOSKeePassDatabase '`"$uri`"'"
               $spooler_log.info( "[" + $property + "]=" + $val);
   ]]></script>
   <run_time />
</job>
```

Explanations

- The `<job>` element makes use of the `stderr_log_level` attribute to cause the job to fail in case of errors being reported to stderr, e.g. from the Java class `SOSKeePassDatabase`.

Finally an individual process can be started to invoke the Java class:

Powershell Job Example (invoke process)

```
 <job  order="no" stop_on_error="no" process_class="/Agent">
    <script  language="powershell"><![CDATA[
                function Get-CredentialStoreProperty( [string] $uri ) {
                        $arguments                                  = @("com.sos.keepass.SOSKeePassDatabase",
$uri)

                        $startInfo                                   = New-Object System.Diagnostics.
ProcessStartInfo
                        $startInfo.FileName                           = "${env:JAVABIN}"
                        $startInfo.RedirectStandardError            = $true
                        $startInfo.RedirectStandardOutput            = $true
                        $startInfo.UseShellExecute                    = $false
                        $startInfo.WindowStyle                           = 'Hidden'
                        $startInfo.CreateNoWindow                    = $true
                        $startInfo.Arguments                            = $arguments

                        try {
                                $process                                      = New-Object System.Diagnostics.Process
                                $process.StartInfo                    = $startInfo
                                $process.Start() | Out-Null
                                $stdout                                          = $process.StandardOutput.ReadToEnd()
                                $stderr                                          = $process.StandardError.ReadToEnd()
                                $process.WaitForExit()
                        }
                        catch {
                                throw "Failed $($startInfo.FileName): $error"
                        }

                        if ( $process.exitCode -ne 0 ) {
                                throw "Failed with exit code $($process.exitCode): $stderr"
                        }

                        $stdout
                }

                $file                 = "$env:SCHEDULER_DATA/config/credential_store.kdbx";
                $property        = "server/SFTP/homer.sos@user";
                $uri                 = "cs://" + $property + "?file=" + $file";

        $val                = Get-CredentialStoreProperty( $uri );
                $spooler_log.info( "[" + $property + "]=" + $val);
    ]]></script>
    <run_time />
</job>
```