# JSSE only Java API proposal

Some notes:

- This is a rough draft to concretize the idea. Of course it must be polished out.
- Point no 1. is **internal** (you can safely skip it). But I think it is better to include this in the API. We anyway will have to program sth like this when implementing the library. If we additionally don't hide it, the library will be more flexible and users will be able to change the logic **reusing our implementation**. Conversely to what we have in JCA.

```
1) interface CertificateChecker - performs low level checks of some certificate properties.
Intended to be used internally, as a part of certificate chain validation. May be stateful
implementations need not to be thread safe - one instance must not be used to check two
certificate chains simultaneously. Constructors may need complex arguments.
[
void check(X509Certificate[] cert, int position, Collection<String> unresolvedCritExts)
   Performs the check(s) on the specified certificate (possibly using its internal
   state and the rest of chain) and removes any critical extensions that it processes from the sp
   collection of OIDs).
Set<String>   getSupportedExtensions()
    Returns an immutable Set of X.509 certificate extensions that are supported.
void init()
    Resets this checker so it can be used to check a new certificate chain.
]

implementations:
TBD either more fine or coarse grained. For sure everything which is truststore-type independent
and dependent must be in separate classes.

1') ValidationListener - invoked when there is an error in processing returned by any checker. If
we may add also onSuccessfulValidation method and also if needed we can add information on the ac
checker class that returned error.
[
   void onValidationError(X509Certificate[] cert, int position, ValidationResult r)
]

1'') ValidationResult
Must contain *at least* error messages list and general result flag (valid or not).

2) interface X509CertChainValidator
[
   ValidationResult validate(CertPath | X509Certificate[])
   X509Certificate[] getTrustedIssuers()
   void addValidationListener(ValidationListener l);
   void removeValidationListener(ValidationListener l);
]

implementations: OpensslCertValidator, JKSCertValidator, ...
Constructor parameters are expected to be very different. Implementations will reuse as many of C
implementations as possible. Must be thread safe.

3) class CanlX509TrustManager implements X509TrustManager
[
   CanlX509TrustManager(X509CertChainValidator)
]

We provide one implementation of X509TrustManager, wrapping cert chain validator instance.

4) interface Credential
[
 KeyStore getAsKeyStore()
 KeyManager getAsKeymanager()
]
```

```
implementations:
 ->JKSCredential (mostly dummy -> impl. exists in JDK)
 ->PEMFileCredential

5) HostnameToCertificateChecker implements HandshakeCompletedListener

6) Utility class SocketFactoryCreator allows programmers to quickly create socket factories.
[
 SSLServerSocketFactory createSSF(Credential c, X509CertChainValidator v)
 SSLSocketFactory createSF(Credential c, X509CertChainValidator v)
]

TBD classes/interfaces:

ProxyCertificateUtils - provides code to produce Proxy certificate.
ProxyCertificate - extends X509Certificate class to hold proxy certificate.
CertificateUtils - other methods as DN comparator.



================== EXAMPLE ====================

OpensslStyleValidator v = new OpensslStyleValidator("/my/certs",
            OpensslStyleValidator.CRL.REQUIRE, OpensslStyleValidator.NAMESPACE.IGNORE);

//for verification
ValidationResult result = v.validate(toBeChecked);

//ssl connection creation
Credential c = new PEMFileCredential(/*params*/);
SSLServerSocketFactory sslSsf = SocketFactoryCreator.createSSF(c, v);

/*
//verbose version - i.e. what happens in SocketFactoryCreator.createSSF()?
KeyManager km = new PEMFileCredential().getAsKeyManager();
TrustManager tm = new CanlX509TrustManager(v);
SSLContext sslCtx = SSLContext.getInstance("TLS", "SunJSSE");
sslCtx.init(new KeyManager[] {km}, new TrustManager[] {tm},
      new SecureRandom());
SSLServerSocketFactory sslSsf = sslCtx.getServerSocketFactory();
*/

ServerSocket sslSS = sslSsf.createServerSocket();
```

-- KrzysztofBenedyczak - 25-Oct-2010

---

This topic: EMI > EmiJra1T4CaNlAPIKB2
Topic revision: r4 - 2010-11-08 - unknown

Copyright &© 2008-2020 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback