How to configure SNMP4J for TLS usage?

Since version 3.0, SNMP4J and SNMP4J-Agent support TLS. This How-To describes how those SNMP4J APIs are configured to use TLS.

How to configure SNMP4J to use TLS?

Before you can start using TLS (or DTLS), you need a key pair. You can generate a self signed one for testing purposes as follows:

```
keytool -keystore dtls-cert.ks -alias dtls-snmp4j-test -storepass snmp4j -keypass snmp4j -genkeypair -keyalg
RSA -keysize 4096 -validity 5000 -dname "CN=www.snmp4j.org, OU=Unit-Test, O=AGENTPP, L=Stuttgart, S=Baden-
Wuerttemberg, C=DE" -ext "san=dns:localhost,ip:127.0.0.1"
```

The following steps then prepare the SNMP4J API for TLS usage:

- The SNMP TLS Transport Model (TLSTM) uses certificate based authentication, thus we need to configure a **trust store** for client authentication (SNMP command generator) and a **key store** (SNMP command responder):

```
-Djavax.net.ssl.trustStore=<trustStoreFilePath> -Djavax.net.ssl.trustStorePassword=<trustStorePassword> -Djavax.
net.ssl.keyStore=<keyStoreFilePath>
-Djavax.net.ssl.keyStorePassword=<keyStorePassword>
```

- Create the TLSTM TransportMapping (which may be used with TlsAddress classes only) and set its SecurityCallback for authentication of remote certificates and selecting the local certificate to be used by the TLSTM for client authentication:

```
// create the TLS transport mapping:
AbstractTransportMapping transport = new TLSTM();

// set the security callback (only required for command responder,
// but also recommended for command generators) -
// the callback will be configured later:
DefaultTlsTmSecurityCallback securityCallback = new DefaultTlsTmSecurityCallback();
((TLSTM)transport).setSecurityCallback(securityCallback);
MessageDispatcher md = new MessageDispatcherImpl();
// we need MPv3 for TLSTM:
md.addMessageProcessingModel(new MPv3());

Snmp snmp = new Snmp(md, transport);

// create and initialize the TransportSecurityModel TSM:
SecurityModels.getInstance().addSecurityModel(new TSM(new OctetString(mpv3.getLocalEngineID()), false));

// do not forget to listen for responses:
snmp.listen();
```

- Create a target and set its address if the SNMP instance is command generator:

```
String sn = "myTlsSecurityName";
CertifiedTarget ct = new CertifiedTarget(GenericAddress.parse("tls:127.0.0.1/161"), new OctetString(sn),
        // server fingerprint (replace with the fingerprint of the server's certificate):
        OctetString.fromHexString("4a:48:60:20:35:10:97:92:de:62:79:ae:85:b9:49:65:e9:03:6d:5a:f8:f3:70:41:9d:
db:50:5a:76:3c:de:b5"),
        // Client fingerprint could be empty string (no check)
        new OctetString());
ct.setSecurityModel(SecurityModel.SECURITY_MODEL_TSM);
```

- If the SNMP instance is a command responder or if one of the following applies then configure the TlsSecurityCallback for the TLSTM instance (see RFC 5953):
    1. The Java virtual machine of the SNMP instance has a key store configured with more than one certificate (then a certificate has to be selected by the http://www.snmp4j.org/doc/org/snmp4j/transport/tls/TlsTmSecurityCallback.html#getLocalCertificateAlias(org.snmp4j.smi.Address) method).

2. No trust key store has been configured or additional trusts (on top of the trust key store) should be established, for example through the mapping rules defined by RFC 5953.

```
// add the distinguished name (DN) of the certificates we want to accept as peer:
securityCallback.addAcceptedSubjectDN("EMAILADDRESS=info@company.com, C=US, CN=Foo Bar");
//
```