eRoaming platform

Secure Connection Guide

intercharge

everywhere

Contents

List of tabels

List of figures

1. Revisions overview

Date	Version	Notice/Reason of Change	Author
08.01.2013	0.1	Creation	Daniel van Maren
11.01.2013	0.8	Completion of the first Draft	Daniel van Maren
15.01.2013	1.0	Added usage of environment variables under windows, corrected minor error.	Daniel van Maren
27.02.2013	1.1	Updated document according changes of the 26.02.2013	Nikolaj Langner
01.03.2013	1.2	Updated the document according feedback of the 01.03.2013	Nikolaj Langner
21.03.2013	1.3	Updated the document according feedback of the 19.03.2013	Nikolaj Langner
25.10.2013	1.4.DRAFT	Added chapters Abbreviations and Security as well as comments	Christian Pestel Michael Thiel
13.11.2013	1.4.DRAFT2	Checked Connectivity Guide for technical Errors and corrected them.	Daniel van Maren
17.06.2014	1.5	Updated pictures and links	Nikolaj Langner
26.11.2015	2.0	Overall layout and CSR generation	Jaime Brodhag

Table 1: Revisions overview

2. Abbrevations

Abbrevation	Description
CA	Certificate authority
CSR	Certificate signing request
DER	Distinguished Encoding Rules
GNU	GNU's Not Unix!
GUI	Graphical User Interface
HBS	Hubject Brokerage System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JKS	Java KeyStore
PEM	Privacy-enhanced Electronic Mail
PGP	Pretty Good Privacy
PKCS10	Public-Key Cryptography Standard #10
PKCS12	Public-Key Cryptography Standard #12
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User interface

Table 2: Abbrevations

3. eRoaming Security Concept

3.1. Motivation

The eRoaming Platform is accessed by roaming partners. Besides the users who access the GUI of the platform, there are backend systems which communicate with the platform in automated ways via web service calls. In the same way the eRoaming Platform accesses the backend systems of the roaming partners.

The eRoaming Platform is operated by Bosch Software Innovations. Roaming partners are operating their systems separately. Communication between them uses standard internet infrastructure. Connections need to be secured to achieve the following general goals of information security:

- Confidentiality – messages can only be read by intended recipients.
- Integrity – altering of messages during transmission (deliberately or by technical errors) must be detected.
- Authenticity – messages must be attributable to a unique sender. The sender must not be able to repudiate the transmission of a message.

3.2. System architecture

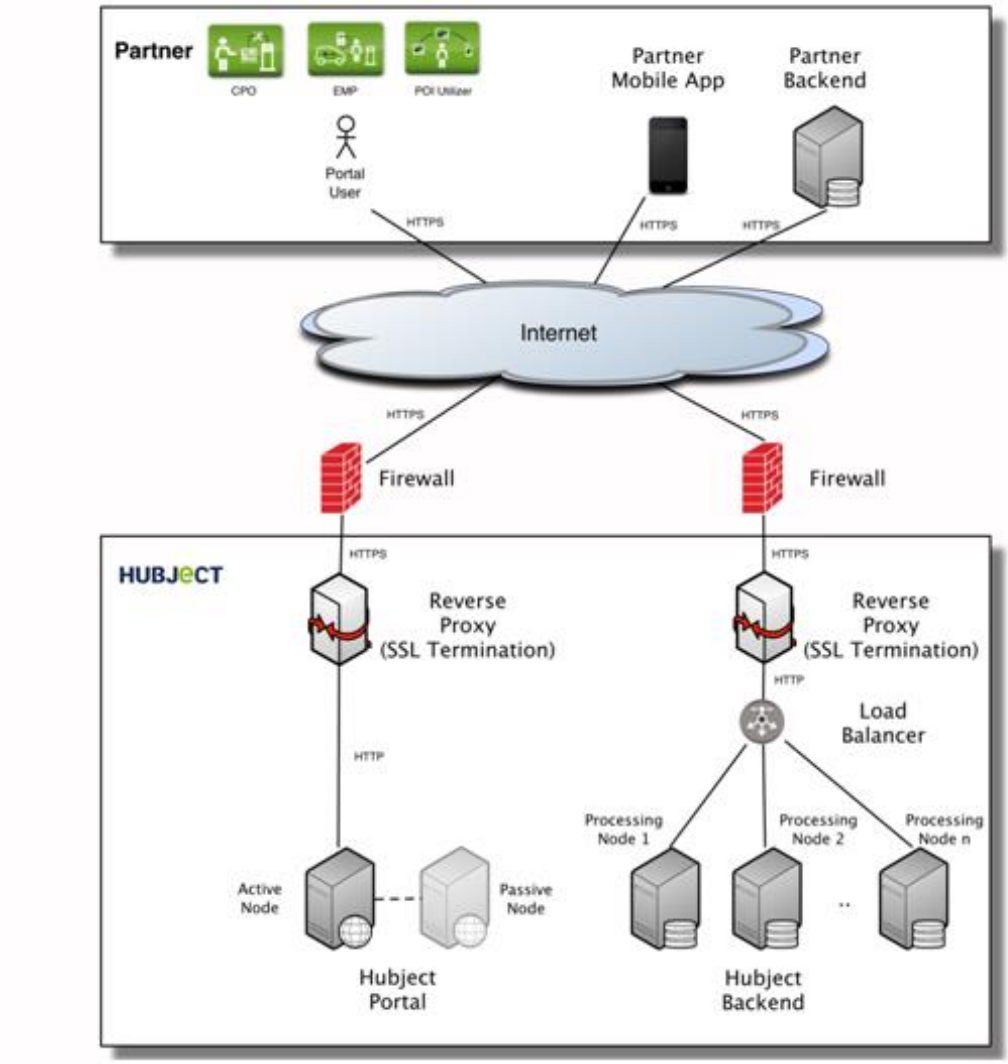

Figure 1: System Architecture

Incoming connections of roaming partner backends are sent via the internet to the load balancer node of which acts as a reverse proxy. The connection is distributed to a cluster of service processing nodes. The connection has to pass a 2-layered firewall when entering the network of the eRoaming Platform.

Both the reverse proxy and the firewall provide access control using white lists which only grant access to specific IP addresses.

The firewall at the network and transport layer is restricted to allowed source/destination IPs and ports.

The proxy at the application layer is restricted to certain URLs.

Outgoing HBS connections are sent directly to partner backend systems passing the firewall.

3.3. HTTPS

The web services are transmitted using HTTPS, the SSL/TLS secured HTTP variant. The SSL tunnel guarantees the above goals of information security: confidentiality via encryption, integrity via signed checksums and authenticity via authentication using digital signatures and certificates.

3.4. Certificates

With HTTPS strong server and client authentication using certificates will be used to authenticate the actual connection used for each service call.

The reverse proxy / load balancer handles the central HTTPS encryption and authentication of all incoming connections to the Hubject Platform. This includes the requests for the portal which is accessed by users with their web browsers.

No changes to the eRoaming Platform are required to add a new partner (i.e. no adding of certificates to local trust stores, no re-deployment of changed Web service connectors and no restarting of Tomcat servers).

4. Preconditions

4.1. OpenSSL

There are a lot other tools for tasks regarding SSL – however we recommend the use of OpenSSL, therefore this guide uses OpenSSL for various instructions.

- If you are using Windows, OpenSSL is available for download from various sources. One possible source is available here: http://www.heise.de/download/win32-openssl.html
- If you are using Linux/Unix, install OpenSSL from your package repository

4.2. Requirements for your PKCS10 CSR

The PKCS10 Certificate Signing Request (CSR) contains different Data: your Public Key and a set of Metadata. In order to receive a Certificate signed by the HubjectCA it is necessary to fulfill some requirements for your CSR:

- Key algorithm: RSA
- Key length: 2048 bit
- Common Name (CN): <DNS of the Partner Backend>
- Organization (O): ”Hubject”
- Organization Unit (OU): <Partner Name>

How to set these values will be covered in the chapter 6 “Creating your PKCS10 CSR”.

In case there is no “real” backend which establishes the connection, but for example a person accessing the web services via SoapUI, please use the DNS Name of the Website of your Company as Common Name, or if only one specific person uses the certificate, use the name of that person.

4.3. Java Keytool

For some of the conversions described in the chapter 9 “Converting your certificate” the Java Keytool is needed. The Java Keytool is part of the JavaSE JRE, which can be downloaded here: http://java.com/en/download/index.jsp

Alternatively, you can install java from the package repository of your Operating System.

4.4. SoapUI

SoapUI can be downloaded from the following website http://sourceforge.net/projects/soapui/.

4.5. Accessing IP-address

At this moment, accesses to all Hubject web services, except the web service for Mobile Authorization are restricted to specific IPs. Before you are able to connect to one of the restricted web services, you will need to communicate your public IP to your onboarding manager which supports you in establishing the connection. This can be for instance the static public IP of your partner backend, or the public IP of your companys Internet Proxy.

For requests from the eRoaming Platform to your Partner Backend please check chapter 11 “Configure an outgoing partner communication”.

5. Establishing a connection

Hubject web services for Backend Communication are protected by using SSL/TLS Client Authentication. This implies that you will need a valid SSL/TLS configuration in order to be able to connect to Hubject web services.

The basic way of establishing a connection to these web services is as follows:

1. Private Key Generation
2. PKCS10 CSR Generation
3. Sending the CSR to your onboarding manager
4. Receiving the HubjectCA Signed Certificate + HubjectCA + BoschSI RootCA Certificates
5. Converting your Certificates & Private Key Pair into your desired format
6. Accessing the web services with Client Authentication

6. Creating your PKCS10 CSR

Please note, that all commands here are examples.
Below each command there is a list explaining the different parameters and what should be the value of the parameter - these values will most probably be different than the example.

During this process, you will create **two files: the private key and the PKCS10 Certificate Signing Request**. Please archive the private key in a secure way: in order to use your x509 Certificate which we will send you, you will need the private key. Without the private key it is impossible to decrypt messages which were encrypted to your public key (which is contained in your CSR and your Certificate).

Furthermore, this means that anyone with your private key may decrypt all messages which were sent to you (also responses on web service requests!), and are able to identify themselves with your client certificate. Therefore, please make sure that only authorized personal is able to access the private key.

Create a directory where the CSR and Private Key should be saved

1. Open your command line

On Windows 7 for instance you can do this by typing “cmd” into the search within the start menu, and click on the “cmd.exe” within the search results.

CAUTION:
Be careful with copying the listed commands below and pasting it into the command line window. It can cause execution problems.

2. Change environment variables

Windows

```
set RANDFILE=C:\Users\userABC\hubject_ssl\.rnd
set JAVA_HOME=%ProgramFiles(x86)%\Java\jre7
set PATH=%PATH%;C:\OpenSSL-Win32\bin
set PATH=%PATH%;%ProgramFiles(x86)%\java\jre7\bin\
```

Please adjust the yellow selection to your preconditions. This means typing in your username and adjusting whether you use Open SSL in Win32 or Win64 bit version.

Make sure that you have Read & Write permissions to the file path you have set in RANDFILE.

CAUTION:
Please note that setting the environment variables will only apply to your current session (Windows: Terminal Window). If the Terminal Window was closed during the process, you will need to repeat this step.
Also please note that the above values are examples which may differ from your environment.

Linux/Unix

Under Linux/Unix Environment variables should be already set correctly.

3. Change to the directory you created before

Windows

```
cd C:\Users\userABC\hubject_ssl
```

Linux/Unix

```
cd ~/hubject_ssl
```

4. Use openssl to create the private key and the csr.

Windows

```
openssl req -config C:\OpenSSL-Win32\bin\openssl.cfg -newkey rsa:2048 -keyout my_private.key -out my_pkcs10.csr -nodes
```

Please adjust the yellow selection to your preconditions. This means adjusting whether you use Open SSL in Win32 or Win64 bit version.

Linux/Unix

```
openssl req -newkey rsa:2048 -keyout my_private.key -out my_pkcs10.csr -nodes
```

See a listing of all parameters in the examples here:

Parameter	Example Value	Description
req	-	Openssl command to manipulate Certificate Signing Requests
-newkey	RSA:2048	Create a new Key with RSA Algorithm and 2048 bits
-keyout	my_private.key	Filename of the new key to generate
-out	My_pkcs10.csr	Filename of the PKCS10 CSR to generate
-nodes	-	Private key will be saved without encryption
-config	C:\OpenSSL-Win32\bin\openssl.cfg	-

Table 3 Creating your PKCS10 CSR

OpenSSL will now require some user input - there are some requirements which your CSR needs to match - in order to fill these fields correctly please refer to the table of parameters below.

Example:

```
Country Name (2 letter code) [AU]:      DE

State or Province Name (full name) [Some-State]:      Baden-Wuerttemberg
```

Locality Name (eg, city) []:Waiblingen

Organization Name (eg, company) [Internet Widgits Pty Ltd]:**Hubject**

Organizational Unit Name (eg, section) []:Bosch Software Innovations GmbH

Common Name (e.g. server FQDN or YOUR name) []:example.bosch-si.com

Email Address []:max.muster@bosch-si.com

You will be asked to fill in now ‘extra’ attributes to be sent with your certificate request. For this Hubject related request it should be left empty (just press enter)

A challenge password [ ]:

An optional company name [ ]:

Parameter	Example Value	Description
Country Name	DE	2 letter code of your country name
State or Province Name	Baden-Wuerttemberg	Your Province or State Name
Locality Name	Waiblingen	city where your company is located
Organization Name (O)	Hubject	Organization name, this value has to be Hubject for Hubject related CSRs as shown in the example
Organizational Unit Name (OU)	Bosch Software Innovations	Your Organizational Unit name, for Hubject related requests you should use your company name in here
Common Name (CN)	example.bosch-si.com	The DNS of your Partner Backend, or your Company’s Website DNS Name in case there is no "real" backend. Alternatively the name of the person using the certificate can be used.
Email-Address	daniel.muster@bosch-si.com	The Email Address of the Person responsible for your CSR
Challenge Password		This Password is used by some CAs to check for revocation. For Hubject related requests it should be left empty (just press enter)
An optional company name		As the parameter describes, optional

Table 4 Requirements of CSR

Now you should have two files within your directory – the example commands create a *my_private.key* file, and a *my_pkcs10.csr* file.

If you have set your RANDFILE environment variable you will additionally have generated a .rnd file.

7. Checking and sending the CSR

Please check the *.csr file before you upload it to your partner wiki space using the following command in your command line. Please note that you maybe need to set your environment variables again because they don't persist unless you set them permanently in your system variables.

```
openssl req -noout -text -in my_pkcs10.csr
```

Your command line will show you the complete *.csr. Please check if the values are set correctly for O and OU as shown above.

Now you should upload only the *my_pkcs10.csr* file to your partner onboarding wiki space. Related to our security concept please create a password secured zip file which incl. the CSR. The password associated with the zip file must be sent separately to your onboarding manager eg. via phone or e-mail.

8. Receiving the certificate

After you uploaded your PKCS10 CSR to your partner wiki space, your CSR will be signed by our IT service provider. After no more than 3 working days you will receive an e-mail that the following files are uploaded to your partner wiki space:

- Your x509 Certificate, signed by the HubjectCA, in the PEM format
- The trusted CA certificates (HubjectCA and BoschSIRootC) -
  to be able to verify the Backend certificate, in the PEM format

The CA certificates should be used for establishing the TLS Trust between the HBS and the partner backend, as the certificate for the service will be renewed in a regular interval (every two years), which would cause the connection to be untrusted if only the service certificate is directly trusted. The HubjectCA certificate will be renewed only every 10 years. The decision if only the sub-certificate or also the root certificate (or even only the chain of both certificates) need to be trusted needs to be done by each partner, as it largely depends on the security obligations and used technologies for TLS.

Now you are able to convert your certificates in addition to your private key into the format you need.

9. Converting your certificate

This section describes how to convert your certificate for various uses. At the end of this chapter you should be able to configure SOAP UI with a my_keystore.jks file for SSL/TLS Client Authentication enabling you to access Hubject's web services via SOAP UI. This guide uses the OpenSSL utility and Java Keytool for conversion issues – refer to chapter 4 Preconditions.

All certificates sent to you by Hubject will be plain x.509 Certificates, formatted in the PEM Format (Base64 encoded DER)

All conversion examples have the precondition, that you have opened your command line, and changed to the directory where the certificates and your private key are located. Instructions how to do this are contained in chapter 5 "Establishing a connection".

Please make sure to obtain the Hubject Root certificate (HubjectCA.crt) from the partner wiki space before you proceed with this manual. The Root certificate needs to be placed in the same folder you are using to generate your own certificate request.

9.1. PEM to PKCS12 conversion

Please note, that all commands here are examples.
Below each command there is a list explaining the different parameters and what should be the value of the parameter - these values will probably be different than the example.

A PKCS12 file contains your **private key**, **your certificate** and the **certificate chain** which belongs to your certificate.

As at the moment the Hubject Root CA directly signs Partner Backend Certificates, the certificate chain only contains this (Hubject CA) certificate. Later when there will be a structured CA, check an updated version of this document to see how you assemble a file containing your certificate chain with multiple certificates.

CAUTION: Sometimes it can happen that the execution of the command below generates the following error "unable to write "random state". Please make sure you have set your RANDFILE Environment variable as described in chapter 6 "Creating your PKCS10 CSR", and you have read & write permissions to the file.

Windows

```
openssl pkcs12 -export -in my_x509.crt -inkey my_private.key -out My_pkcs12.p12
```

Linux/Unix

```
openssl pkcs12 -export -in my_x509.crt -inkey my_private.key -out My_pkcs12.p12
```

In order to be able to convert the PKCS12 into a JKS file later, you will need to enter an export password when executing one of the commands above. This needs to be a 4 number code.

Parameter	Example Value	Description
pkcs12		OpenSSll L utility for manipulating PKCS12 files
-export		You want to export a PKCS12 file
-in	my_x509.crt	Your x.509, PEM formatted certificate you want to convert
-key	my_private.key	The private key you created when creating the CSR for your certificate
-out	My_pkcs12.p12	The file where the new PKCS12 file will be written
-config	C:\OpenSSL-Win32\bin\openssl.cfg (Windows only)	Choose the openssl config file, for troubleshooting with windows installations of openssl, may be optional

Table 5: Parameters for converting to a pkcs12 file

9.2. PKCS12 to JKS conversion

Please note, that all commands here are examples.
Below each command there is a list explaining the different parameters and what should be the value of the parameter - these values will probably be different than the example.

A Java Keystore is able to contain a complete SSL/TLS configuration – it can consist of **your certificate**, **your private key**, **your certificate chain** and can also contain trusted CA Certificates. Java Keystores are manipulated with the Java Keytool – refer to chapter Java Keytool.

The JKS format is needed to configure **SoapUI** with SSL/TLS Client Authentication.

Convert the PKCS12 file to the JKS format. For the following command you need to choose a jks_password. This need to be at least 6 characters long.

Windows

```
keytool -importkeystore -deststorepass <jks_password> -destkeypass <jks_password> -destkeystore my_keystore.jks -srckeystore My_pkcs12.p12 -srcstoretype PKCS12 -srcstorepass <pkcs12_export_password> -alias 1
```

Please adjust the yellow selection to your decided input. Type in your password without brackets.

If your keytool.exe is not accessible from your hubject_ssl directory, you can convert the PKCS12 file to the JKS format by navigating to your java/bin directory (example: C:\Program Files\Java\jre7\bin\). Type the following command and change the highlighted parts to your values:

keytool -importkeystore -deststorepass <jks_passwort> -destkeypass <jks_passwort> -destkeystore C:\Users\YourUserName\hubject_ssl\my_keystore.jks -srckeystore C:\Users\YourUserName\hubject_ssl\My_pkcs12.p12 -srcstoretype PKCS12 -srcstorepass <pkcs12_export_password> -alias 1

Linux/Unix

```
keytool -importkeystore -deststorepass <jks_password> -destkeypass <jks_password> -destkeystore my_keystore.jks -srckeystore My_pkcs12.p12 -srcstoretype PKCS12 -srcstorepass <pkcs12_password> -alias
```

Please adjust the yellow selection to your decided input. Type in your password without brackets.

Parameter	Example Value	Description
-importkeystore		Keytool Command to import from a keystore
-deststorepass	<jks_password>	Password for the new JKS Keystore to be generated
-destkeypass	<jks_password>	Password for the private key within the java key store – for maximum compability this should be the same as -deststorepass
-destkeystore	my_keystore.jks	The filename where the new jks keystore should be written
-srckeystore	My_pkcs12.p12	The filename of the PKCS12 file which should be imported
-srcstoretype	PKCS12	The Certificate store type – the value of this parameter needs to be PKCS12 for the use case of converting a PKCS12 file to the Java Keystore format.
- srcstorepass	<pkcs12_password>	The password which was used previously when creating the PKCS12 file
-alias	1	The alias which will be used for importing – this should be “1” for compability reasons

Table 6: Parameters for converting to a jks file

Importing the trusted CA for backend certificate validation:

Windows

```
keytool -import -alias HubjectCA -keystore my_keystore.jks -trustcacerts -file
HubjectCA.crt
```

Linux/Unix

```
keytool -import -alias HubjectCA -keystore my_keystore.jks -trustcacerts -file
HubjectCA.crt
```

After this command you will be asked, if you want to trust the certificate. For a success certification process please type in “yes”.

Parameter	Example Value	Description
-import		Keytool command to import a certificate
-alias	HubjectCA	Alias of the certificate within the keystore, for this use case the value should be the name of the CA
-keystore	my_keystore.jks	The filename of your previously created Java Keystore.
-trustcacerts		Adds the certificate as “trusted” within the java key store
-file	HubjectCA.crt	The filename of the Trusted CA, this will contain the x.509 PEM formatted certificate which signed the Hubject backend certificate. This file will be sent to you in answer on your PKCS10 CSR.

Table 7: Parameters to trust a CA certificate

Now you should be able to configure for example SoapUI with the *my_keystore.jks* file for SSL/TLS Client Authentication. This enables you to use SoapUI for accessing the Hubject web services.

10. Configure SoapUI for SSL usage

Please follow the instructions below how to configure SoapUI to use the created Java keystore *my_keystore.jks.*

Start SoapUI and click on *"File" -> "Preferences".* Choose the *"SSL Settings"* in the left menu.

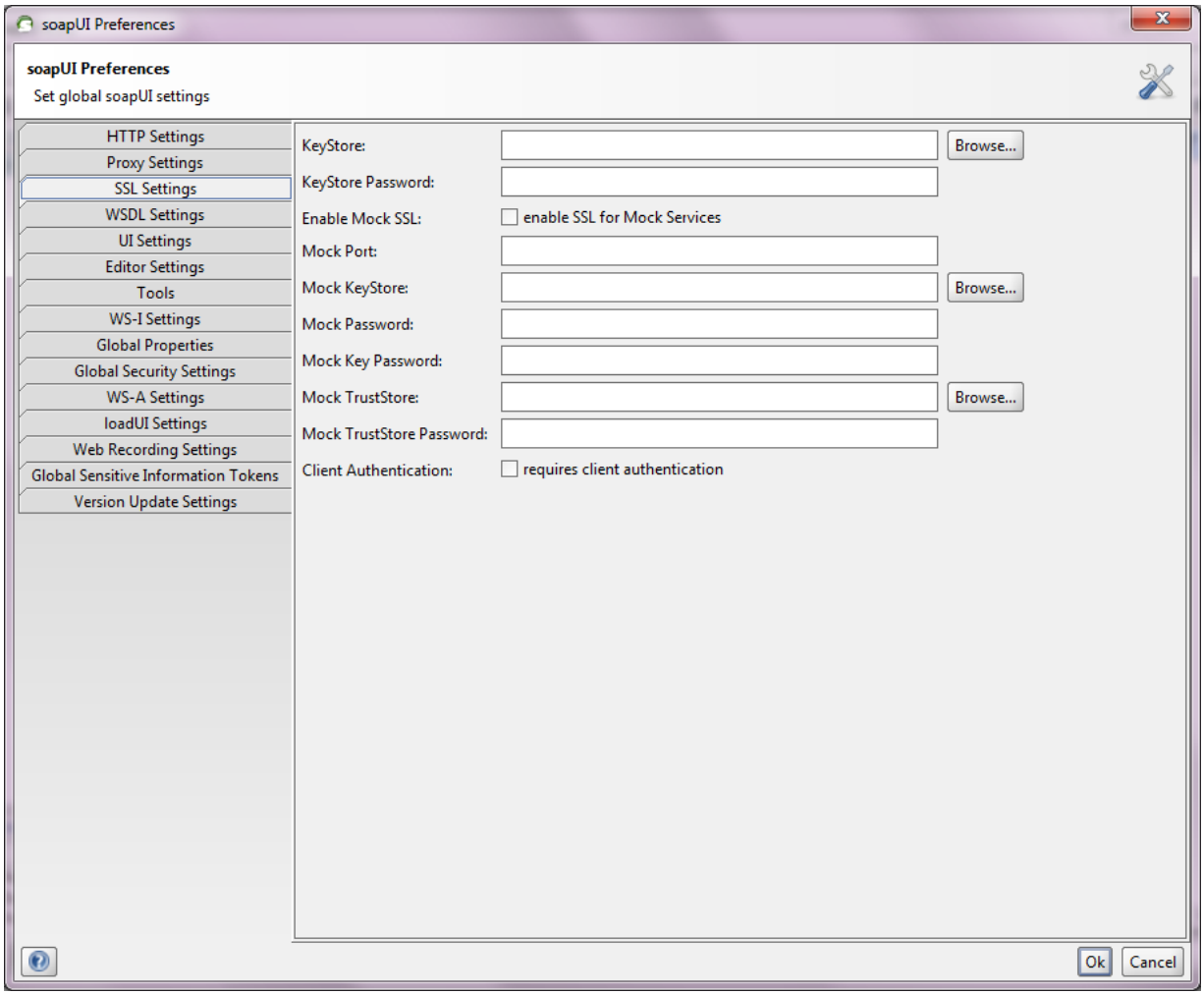

Figure 2: Configure SSL settings in SaopUI

Click on the *"Browse..."* button next to the *"Keystore:"* text field and choose the *my_keystore.jks*.

Enter your keystore Password into the "Keystore Password" field.
Enable the *"Client Authentication:"* option for using the client certifcate stored in your generated JKS.

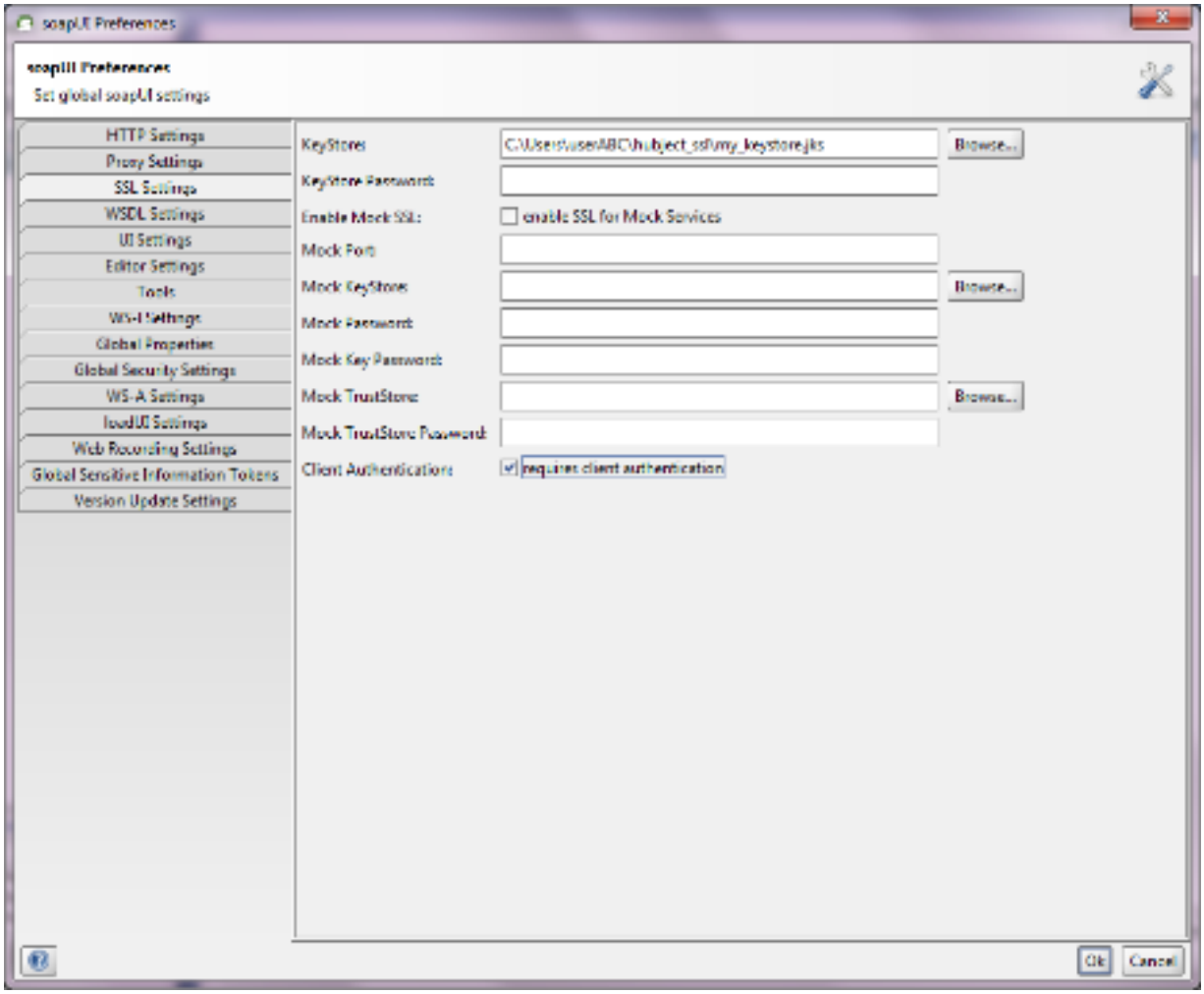

Figure 3: Import of jks file to SoapUI

Click on the *"Ok"* button to proceed configuring your web services.

Several SoapUI versions (for example 4.0.1) are buggy - one often experienced bug is, that Soap-UI does not properly apply your keystore configuration.
In order to lower the risk please consider restarting Soap-UI after configuration changes and double-check the SSL and Proxy configuration after the restart.

11. Configure an outgoing partner communication

For each partner an outgoing communication can be established to an external server. The connection needs to be SSL encrypted. Currently only an outgoing SSL connection with server certificate verification can be supported. To establish an outgoing connection to your desired system Hubject needs to import the Root certificate of the CA signing your server certificate (e.g. Verisign). The Root certificate can be downloaded from the website of the company providing the signed certificate. Please send the certificate to support@hubject.com.
For the latest information regarding outgoing partner connections please contact support@hubject.com.

If your Partner Backend is located behind a Firewall you also should allow incoming connections from the Hubject Platform IP Addresses to your Hubject web services. These IP Addresses differ depending on the environment you want to use or you want to be reached by.