Nations Unies ECE/TRANS/WP.30/GE.1/2021/5

# Conseil économique et social

Distr. générale
3 décembre 2020
Français
Original : anglais

## Commission économique pour l'Europe

Comité des transports intérieurs

### Groupe de travail des problèmes douaniers intéressant les transports

**Groupe d'experts des aspects théoriques et techniques de l'informatisation du régime TIR**

**Première session**
Genève, 27-29 janvier 2021
Point 6 a) de l'ordre du jour provisoire
**Système international eTIR : Rapport sur l'état d'avancement de l'élaboration du système international eTIR**

# Services Web eTIR − Glossaire et procédures techniques*

**Note du secrétariat**

## I. Introduction − Mandat

1. À sa quatre-vingt-deuxième session (23-28 février 2020), le Comité des transports intérieurs a approuvé la création du Groupe d'experts des aspects théoriques et techniques de l'informatisation du régime TIR (WP.30/GE.1) (ECE/TRANS/294, par. 84[1]) et a approuvé son mandat[2] (ECE/TRANS/WP.30/2019/9 et ECE/TRANS/WP.30/2019/9/Corr.1), sous réserve de l'accord du Comité exécutif de la CEE. Le Comité exécutif, à sa réunion informelle à distance du 20 mai 2020, a approuvé la mise en place du WP.30/GE.1 jusqu'en 2022, sur la base du mandat figurant dans le document ECE/TRANS/WP.30/2019/9 et Corr.1, tel que reproduit dans le document ECE/TRANS/294 (ECE/EX/2020/L.2, par. 5 b))[3].

2. Le mandat du Groupe dispose que celui-ci doit concentrer ses travaux sur l'élaboration d'une nouvelle version des spécifications eTIR, en attendant la mise en place officielle de l'Organe de mise en œuvre technique (TIB). Plus précisément, le Groupe est chargé : a) d'établir une nouvelle version des spécifications techniques de la procédure eTIR, avec les modifications à y apporter, en veillant à assurer leur conformité avec les

* Le présent document a été soumis tardivement aux services de traitement de la documentation en raison de contretemps liés à sa mise au point.

[1] Décision du Comité des transports intérieurs( ECE/TRANS/294, par. 84) − www.unece.org/fileadmin/DAM/trans/doc/2020/itc/ECE-TRANS-294f.pdf.

[2] Mandat du Groupe nouvellement créé, approuvé par le Comité des transports intérieurs et le Comité exécutif de la CEE − www.unece.org/fileadmin/DAM/trans/bcf/wp30/documents/2019/ECE-TRANS-WP30-2019-09f.pdf et rectificatif ; www.unece.org/DAM/trans/bcf/wp30/documents/2019/ECE-TRANS-WP30-2019-09c1f.pdf.

[3] Décision du Comité exécutif (ECE/EX/2020/L.2, par. 5 b)) − www.unece.org/DAM/commission/EXCOM/Agenda/2020/Remote_informal_mtg_20_05_2020/Item_4_ECE_EX_2020_L.2_Mandates_fr.pdf.

spécifications fonctionnelles de la procédure eTIR ; b) d'établir une nouvelle version des spécifications fonctionnelles de la procédure eTIR, avec les modifications à y apporter, en veillant à assurer leur conformité avec les spécifications conceptuelles de la procédure eTIR ; c) d'élaborer des amendements aux spécifications conceptuelles de la procédure eTIR, à la demande du Groupe de travail des problèmes douaniers intéressant les transports (WP.30).

3. Le présent document contient un glossaire et décrit les procédures techniques à suivre pour créer un certificat X.509 et tester la connexion aux services Web eTIR. Il est valable pour la version 1.0 du système international eTIR, basé sur la version 4.3a des spécifications eTIR.

# II. Procédures de remplacement

4. Les procédures de remplacement pratiques du système eTIR sont actuellement décrites dans les amendements approuvés aux documents théoriques, fonctionnels et techniques relatifs au système eTIR (version 4.2a), qui seront inclus dans la prochaine version des spécifications eTIR (4.3). En outre, chaque message peut être assorti d'une procédure de remplacement technique, qui sera décrite dans le guide d'utilisation du message eTIR correspondant.

# III. Assistance et coordonnées

5. Il convient de noter que, dans le cadre des projets d'interconnexion des services douaniers, le secrétariat TIR est prêt à aider les Parties contractantes à connecter leurs systèmes douaniers nationaux au système international eTIR. Pour toute question ou problème se rapportant au présent document ou au système international eTIR, les coordonnées ci-dessous peuvent être utilisées (les communications par courrier électronique sont préférables).

| | |
|---|---|
| *Entité* | *Secrétariat TIR de la Commission économique pour l'Europe,*<br>*ONU,*<br>*Palais des Nations,*<br>*1211 Genève 10, Suisse* |
| Coordonnées | Courriel : etir@un.org<br>Téléphone : +41 (0)22 917 55 06 |

# IV. Glossaire

**Algorithme de cryptographie asymétrique**

Système cryptographique reposant sur l'utilisation de deux clés : une clé publique, connue de tout le monde, et une clé privée (ou clé secrète), que seul le propriétaire de la paire de clés connaît. À titre d'exemple, lorsqu'Alice veut envoyer un message sécurisé à Bob, elle utilise la clé publique de celui-ci pour chiffrer le message. Bob utilise ensuite sa clé privée pour le déchiffrer. L'algorithme RSA est un exemple d'algorithme asymétrique.

**API**

Interface de programmation d'applications. Interface logicielle utilisée pour accéder à une application ou à un service à partir d'un programme.

**Authentification**

Processus consistant à vérifier ou à tester la validité d'une identité déclarée. Le sujet doit fournir des informations supplémentaires qui correspondent à l'identité qu'il revendique. Le système d'authentification le plus courant est l'utilisation d'un mot de passe (qui peut prendre des formes variables, comme le code secret (PIN) ou la phrase secrète). L'authentification consiste à vérifier l'identité du sujet en comparant un ou plusieurs

éléments à ceux enregistrés dans la base de données des identités valides (c'est-à-dire les comptes utilisateurs).

**Autorité de certification (AC)**

Autorité qui occupe une position de confiance : le certificat qu'elle émet lie l'identité d'une personne ou d'une entreprise à la paire de clés publique et privée (cryptographie asymétrique) qui est utilisée pour sécuriser la plupart des transactions sur Internet. Par exemple, lorsqu'une entreprise ou une personne souhaite utiliser ces technologies, elle demande à une AC de lui délivrer un certificat. L'AC recueille des informations concernant la personne ou l'entreprise, qu'elle va vérifier avant de délivrer le certificat.

**B2B**

Relation d'entreprise à entreprise (le terme « entreprise » désignant une entité du secteur privé).

**B2C**

Relation d'entreprise à service douanier (le terme « entreprise » désignant une entité du secteur privé).

**C2B**

Relation de service douanier à entreprise (le terme « entreprise » désignant une entité du secteur privé).

**C2C**

Relation de service douanier à service douanier.

**Certificat numérique X.509**

En général, un certificat est un document délivré par une autorité pour attester une vérité ou donner une preuve. Un certificat numérique est couramment utilisé pour fournir sous forme électronique des preuves concernant le détenteur du certificat. Dans les ICP, il émane d'une tierce partie de confiance, appelée autorité de certification (AC), dont il porte la signature numérique.

**Confidentialité**

Principe appliqué en recourant à des mesures destinées à protéger le caractère secret des données, des objets ou des ressources. La protection de la confidentialité a pour but d'empêcher ou de réduire au minimum l'accès non autorisé aux données. Elle repose sur des mesures de sécurité qui visent à garantir que personne d'autre que le destinataire d'un message ne le reçoive ou ne puisse le lire. Il s'agit de donner aux utilisateurs autorisés un moyen d'accéder à des ressources et d'interagir avec elles, tout en empêchant activement les utilisateurs non autorisés de le faire.

**Émetteur**

Dans le présent document et tous autres documents relatifs aux paires de messages, système d'information de la partie prenante du système eTIR qui établit et envoie un message à une autre partie prenante.

**Environnements**

Un logiciel est développé et maintenu dans plusieurs environnements : l'environnement de développement est utilisé pour mettre au point le logiciel, l'environnement de test d'acceptation sert à faire tester et valider le logiciel par d'autres parties prenantes et l'environnement d'exploitation est celui dans lequel le logiciel est exploité une fois lancé et mis à la disposition des utilisateurs finals.

**Erreur**

Grave défaut de validation qui entraîne le rejet du message.

**ICP**

Infrastructure à clés publiques. Infrastructure nécessaire à la cryptographie asymétrique.

**Intégrité**

Principe consistant à protéger la fiabilité et l'exactitude des données. La protection de l'intégrité empêche toute altération non autorisée des données et garantit que celles-ci demeurent correctes, inchangées et préservées. Convenablement assurée, la protection de l'intégrité permet d'apporter des modifications autorisées tout en protégeant les données des activités intentionnellement malveillantes (telles que les attaques de virus et les intrusions) ainsi que des erreurs commises par les utilisateurs autorisés (telles que les erreurs et les omissions).

**ITDB**

Banque de données internationale TIR. Application de la CEE dans laquelle sont consignées toutes les données relatives aux titulaires de carnets TIR et aux bureaux de douane. Les données qui figurent dans ce système d'information sont gérées par les associations nationales et les autorités douanières du système TIR.

**Jeton**

Parfois appelé jeton de sécurité. Objet contrôlant l'accès à un bien numérique. Traditionnellement, ce terme désigne un dispositif matériel d'authentification, c'est-à-dire un petit appareil utilisé dans un environnement en réseau pour générer un mot de passe à usage unique, que le propriétaire saisit dans un écran de connexion en plus d'un identifiant et d'un code secret. Dans le contexte des services Web, au vu du besoin croissant de dispositifs et de processus multiples d'authentification sur des réseaux ouverts, l'acception du terme « jeton » a été élargie pour englober également les dispositifs logiciels. Un jeton peut être un certificat X.509 qui associe par exemple une identité à une clé publique.

**Jeton X.509**

Certificat X.509 dans lequel figurent les moyens d'identification électronique de l'utilisateur final qui peuvent être transmis par message SOAP. Le jeton X.509 peut être utilisé pour authentifier l'utilisateur dans le contexte d'une relation de confiance (ICP), ainsi que pour signer, chiffrer et déchiffrer un message SOAP.

**Keystore**

Base de données de clés de chiffrement qui sert à stocker des certificats, notamment ceux de toutes les parties de confiance, en vue de leur utilisation par un programme. Grâce à son fichier keystore, une entité peut authentifier d'autres parties et s'authentifier auprès d'autres parties.

**Non-répudiation**

Principe qui garantit que le sujet d'une activité ou la personne qui a causé un événement ne peut pas nier que l'événement s'est produit. En vertu de ce principe, un sujet ne peut pas prétendre qu'il n'a pas envoyé un message, accompli une action ou été la cause d'un événement. Il est rendu possible par l'identification, l'authentification, l'autorisation, l'obligation de rendre compte et l'audit. La non-répudiation peut être appliquée à l'aide de certificats numériques, d'identifiants de session, de relevés des transactions et de nombreux autres dispositifs de contrôle des transactions et des accès.

**OASIS**

Organization for the Advancement of Structured Information Standards. Consortium international à but non lucratif dont l'objectif est de promouvoir l'adoption de normes indépendantes des produits.

**Partie prenante du système eTIR**

Entité faisant partie du système eTIR et utilisant la procédure eTIR telle que décrite dans l'annexe 11 de la Convention TIR. Une partie prenante utilise ses systèmes d'information pour faire partie du système eTIR. Il peut s'agir de l'une des entités suivantes :

- La CEE, avec le système international eTIR ;
- L'Union internationale des transports routiers (IRU), qui représente la chaîne de garantie, avec ses systèmes d'information ;
- Les autorités douanières, avec leurs systèmes douaniers nationaux ;
- Les titulaires, avec leurs systèmes d'information.

**Récepteur**

Dans le présent document et tous autres documents relatifs aux paires de messages, système d'information de la partie prenante du système eTIR qui reçoit un message envoyé par une autre partie prenante et le traite.

**RSA**

Algorithme inventé en 1977 par Ronald L. Rivest, Adi Shamir et Leonard Adleman. Il s'agit d'un algorithme asymétrique, c'est-à-dire qu'il repose sur l'utilisation de deux clés différentes ayant une relation mathématique. La clé publique et les clés privées sont générées à l'aide de l'algorithme RSA et peuvent être utilisées pour chiffrer des informations ou pour signer.

**Schéma XML**

Recommandation du W3C précisant comment décrire formellement les éléments d'un document au format XML.

**Sécurité des services web (WS-Security)**

Spécification décrivant les améliorations apportées au protocole SOAP version 1.1 pour renforcer la protection (l'intégrité) et la confidentialité des messages. Ces améliorations passent par des fonctionnalités permettant de sécuriser les messages SOAP grâce à une signature numérique XML, d'assurer la confidentialité à l'aide du chiffrement XML et de propager les moyens d'identification électronique grâce aux jetons de sécurité (par exemple, le jeton X.509).

**Service Web**

Communication sur un réseau entre deux applications (et non entre une personne et une application, par exemple). Type de communication également appelé « communication de machine à machine ».

**SI eTIR**

Acronyme parfois utilisé dans les diagrammes pour désigner le système international eTIR.

**Signature numérique**

Code numérique qui peut être joint à un message transmis par voie électronique, dans deux buts distincts : 1) garantir au destinataire que le message provient réellement de l'expéditeur déclaré, en appliquant le principe de non-répudiation (c'est-à-dire que l'expéditeur ne peut pas prétendre ultérieurement que le message était falsifié) ; 2) garantir au destinataire que le message n'a pas été altéré pendant son transit de l'expéditeur au destinataire. La signature

numérique protège à la fois contre la modification malveillante (une tierce partie altère délibérément le sens du message) et contre la modification involontaire (due à des failles dans le processus de communication, comme des interférences électriques).

**Signature XML**

Spécification établie conjointement par le World Wide Web Consortium (W3C) et le Groupe d'étude sur l'ingénierie Internet (IETF). La signature XML fournit des services de protection de l'intégrité et d'authentification du message ou du signataire pour des données de tous types, que ce soit dans le fichier XML comprenant la signature ou ailleurs.

**SOAP**

Simple Object Access Protocol. Protocole de messagerie défini pour l'échange d'informations dans le cadre de l'exécution de services Web. Il s'agit d'un protocole basé sur XML qui se compose de trois parties :

- Une enveloppe, qui définit la structure du message (en-tête et corps) et la façon dont il doit être traité ;
- Un ensemble de règles de codage permettant de décrire les instances des types de données liées à l'application ;
- Une convention pour la représentation des appels de procédure et des réponses.

**Truststore**

Fichier keystore qui contient des certificats d'autres parties avec lesquelles on prévoit de communiquer ou d'autorités de certification auxquelles on fait confiance pour identifier d'autres parties.

**Valeur de hachage**

Également appelée « hash » ou « résumé de message ». Valeur générée à partir d'un texte, sensiblement plus réduite que celui-ci. Elle est générée par une formule conçue de telle sorte qu'il est extrêmement improbable qu'un autre texte produise la même valeur de hachage.

**WSDL**

Langage de description des services Web. Langage de description d'interface basé sur XML, utilisé pour décrire la fonctionnalité offerte par un service Web.

**XML**

Langage de balisage extensible. Langage définissant un ensemble de règles pour le codage des documents sous une forme lisible à la fois par les personnes et par les machines. Il est utilisé dans le cadre du protocole SOAP pour encoder les messages envoyés par les services Web.

**Résumé technique**

Pour référence, le tableau ci-dessous récapitule tous les renseignements techniques en une seule page.

| *Élément* | *Valeur* | *Observations* |
|---|---|---|
| Spécifications eTIR | Version 4.3 | |
| Adresse URL de l'environnement de test d'acceptation nº 1 | https://etir-uat-01.unece.org/ | |
| Point de terminaison SOAP pour les messages internes | {URL de l'environnement}/etir/v4.3/customs | |
| Point de terminaison SOAP pour les messages externes | {URL de l'environnement}/etir/v4.3/guaranteeChain | |
| Version du protocole SOAP | v1.2 | |
| Protocole utilisé pour le HTTPS | Protocole TLS v1.2 et v1.3 | Il est recommandé d'avoir un système d'information qui supporte le protocole TLS v1.3, car TLS v1.2 sera certainement mis hors service en 2021 pour des raisons de sécurité. |
| Version du certificat X.509 | 3 | Il s'agit des certificats X.509 utilisés comme signatures électroniques pour les messages envoyés entre les différentes parties prenantes du système eTIR. |
| Taille de la clé utilisée pour la génération du certificat X.509 | 2048 bits | |
| Algorithme de signature utilisé pour la génération du certificat X.509 | SHA-256 couplé à RSA | |
| Version de l'UUID à utiliser | 4 | Les UUID sont utilisés dans les champs « Identificateur de message » et « Référence fonctionnelle » |

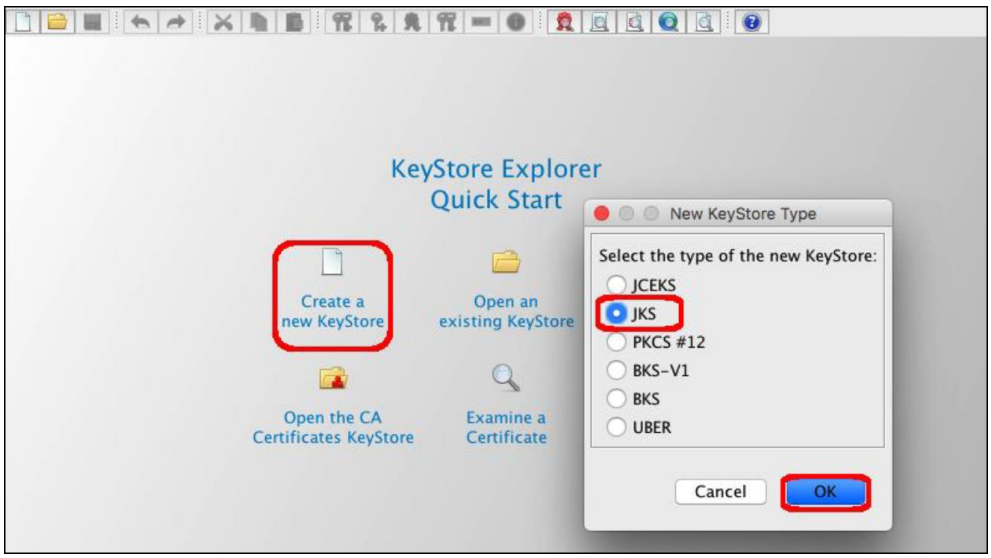
# V. Keystore : explication pas à pas pour la génération de paires de clés

## A. À l'aide de l'application KeyStore Explorer

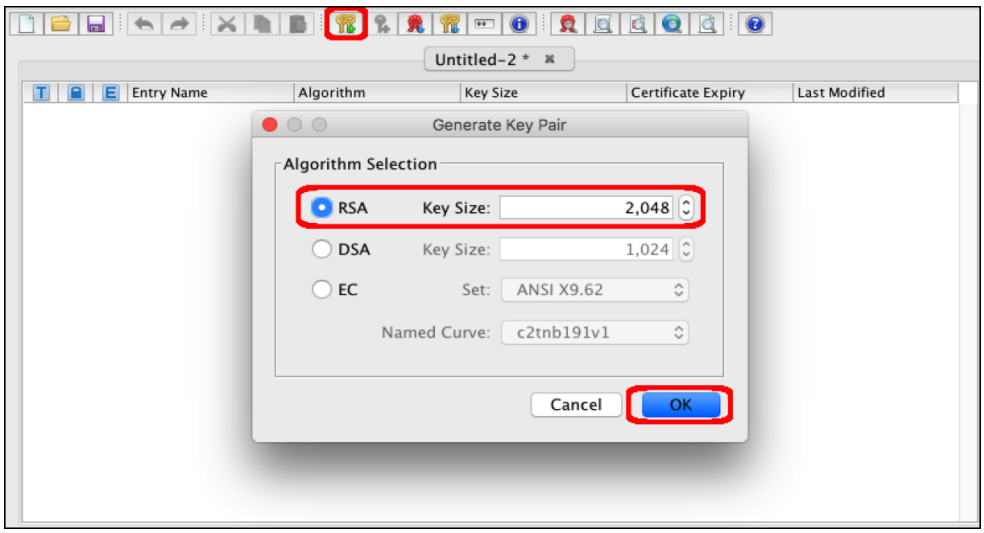
La procédure ci-dessous décrit les manipulations à faire à l'aide de l'interface utilisateur graphique en libre accès KeyStore Explorer. La dernière version de l'outil peut être téléchargée à l'adresse suivante : http://www.keystore-explorer.org/.

**Générer une paire de clés**

1. Ouvrir KeyStore Explorer. Cliquer sur « Create a new KeyStore » (Créer un nouveau magasin de certificats) puis sélectionner le type JKS.

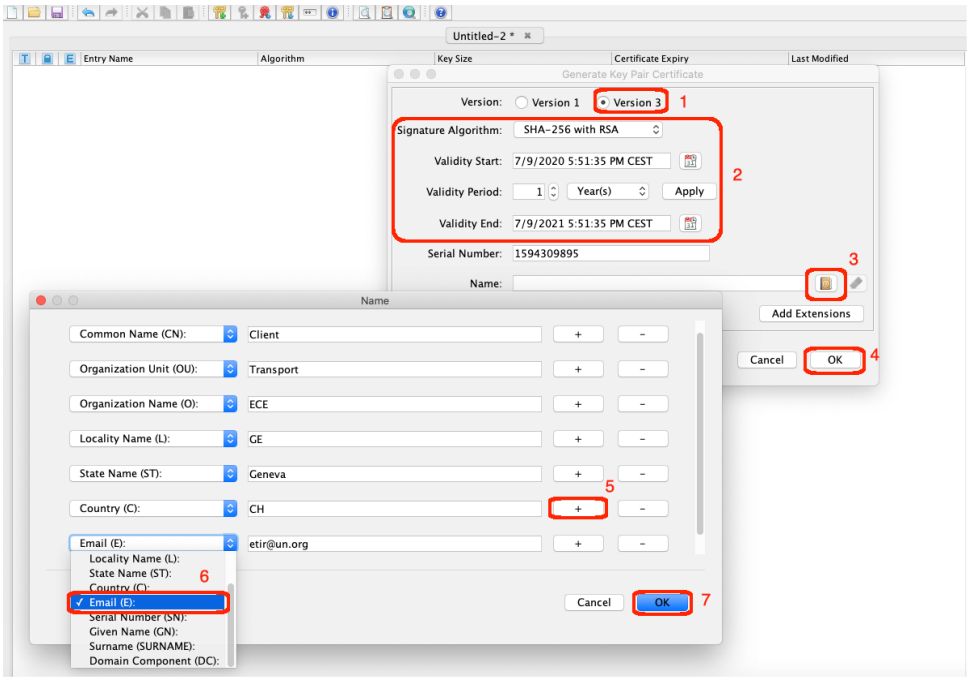



2. Cliquer sur l'icône « Generate Key Pair » (Générer une paire de clefs) puis, dans la fenêtre, choisir l'algorithme RSA et saisir 2048 dans le champ « Key Size » (Taille de clef).

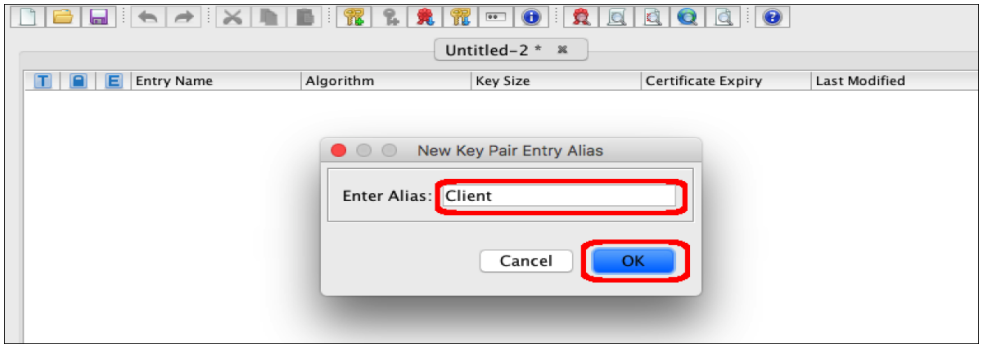


3. Dans la fenêtre « Generate Key Pair Certificate » (Générer un certificat pour la paire de clefs), suivre l'ordre des étapes indiqué dans la capture d'écran ci-dessous, en sélectionnant « Version 3 » et « SHA-256 with RSA » (SHA-256 avec RSA) comme algorithme et en indiquant la période de validité souhaitée.

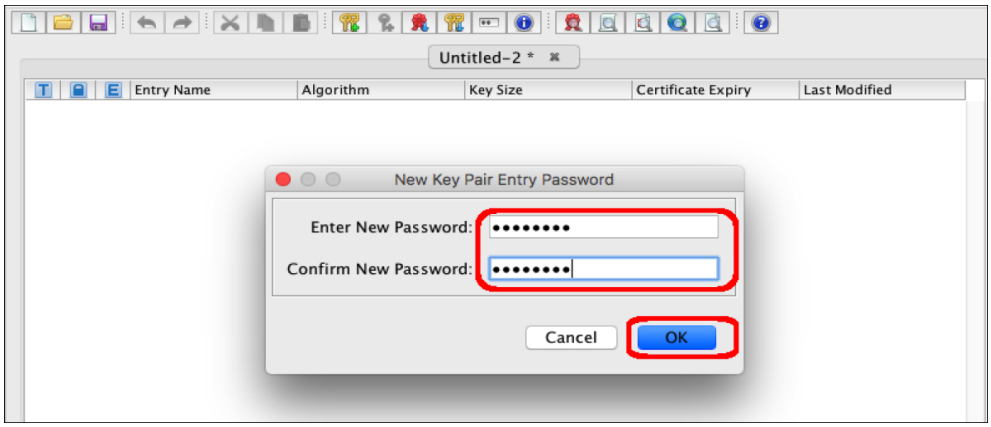
Il importe de saisir les informations relatives au bureau de douane visé et non les valeurs types affichées dans la capture d'écran.



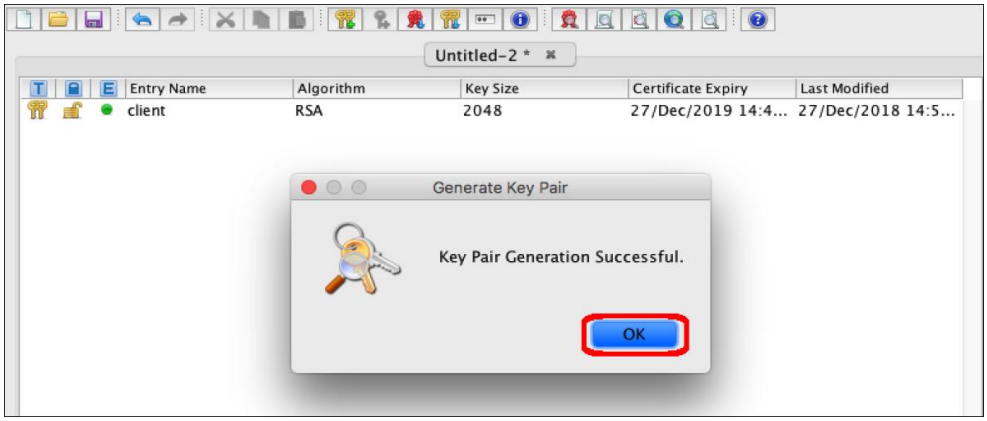

4. Dans la fenêtre qui s'ouvre, choisir un nom d'alias pour la paire de clés.



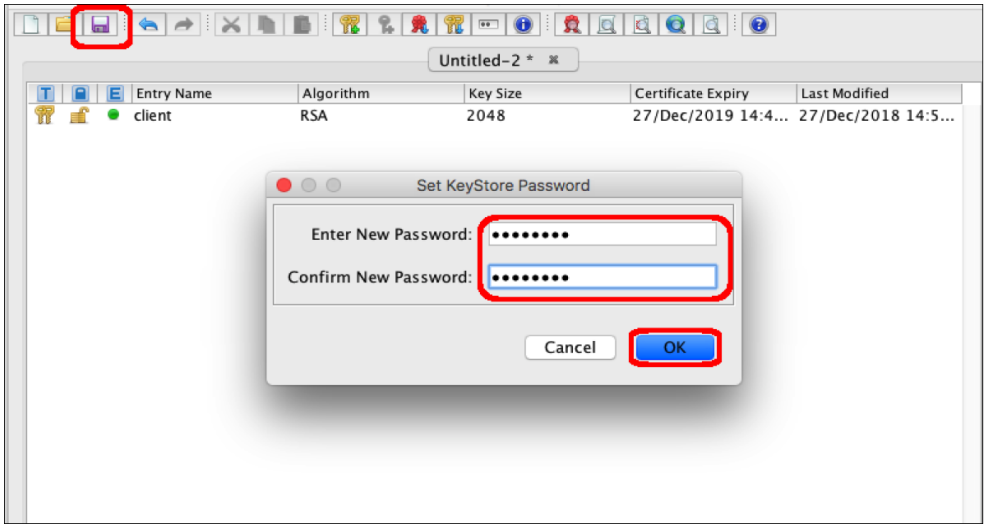

5. Dans la fenêtre qui s'affiche ensuite, définir un mot de passe pour la paire de clés.



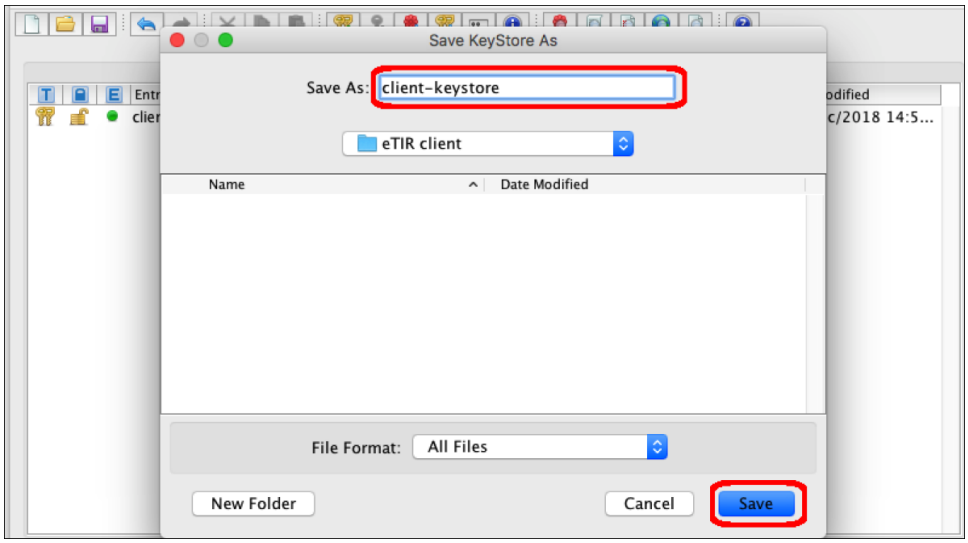
6. Une fenêtre confirme que la paire de clés a été générée.




7. Une fois la paire de clés générée, cliquer sur l'icône « Save » (Enregistrer). Dans la fenêtre qui apparaît, définir le mot de passe du keystore (il est conseillé d'utiliser un mot de passe différent de celui choisi pour la paire de clés).

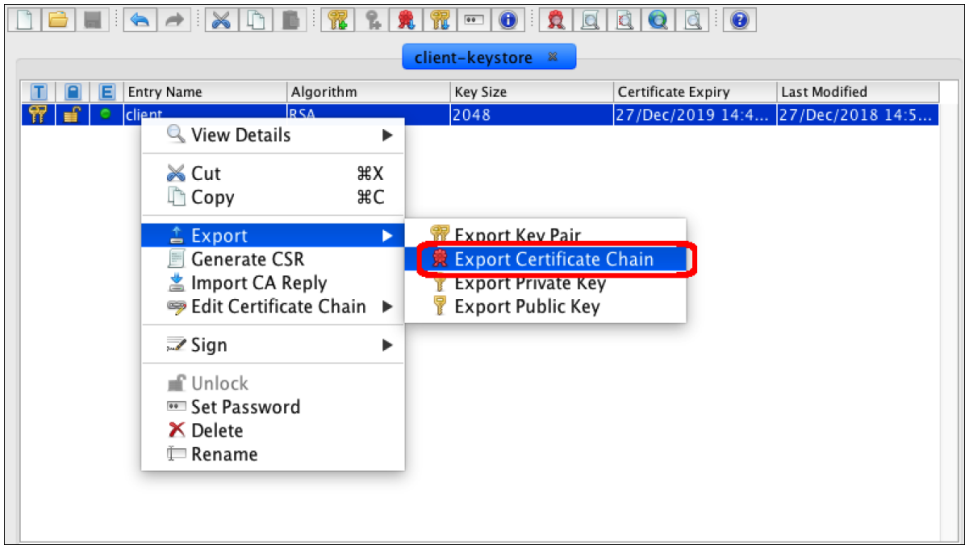



8. Choisir le nom et l'emplacement du keystore (le fichier devrait avoir une extension « .jks »), puis cliquer sur « Save » (Enregistrer).

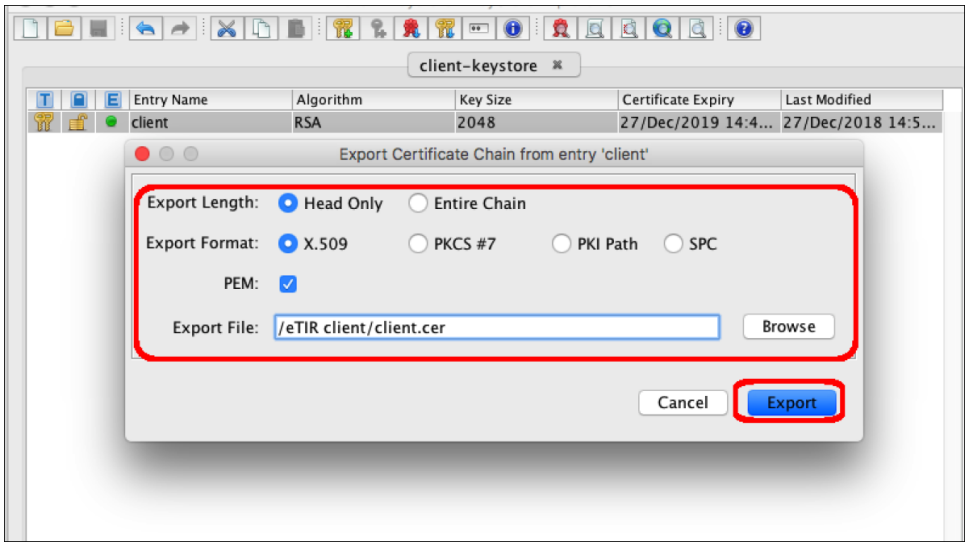


**Exporter le certificat client**

1. Dans KeyStore Explorer, ouvrir le keystore que vous avez créé. Faire un clic droit sur la paire de clés générée, sélectionner Export (Exporter) puis Export Certificate Chain (Exporter la chaîne de certificats).

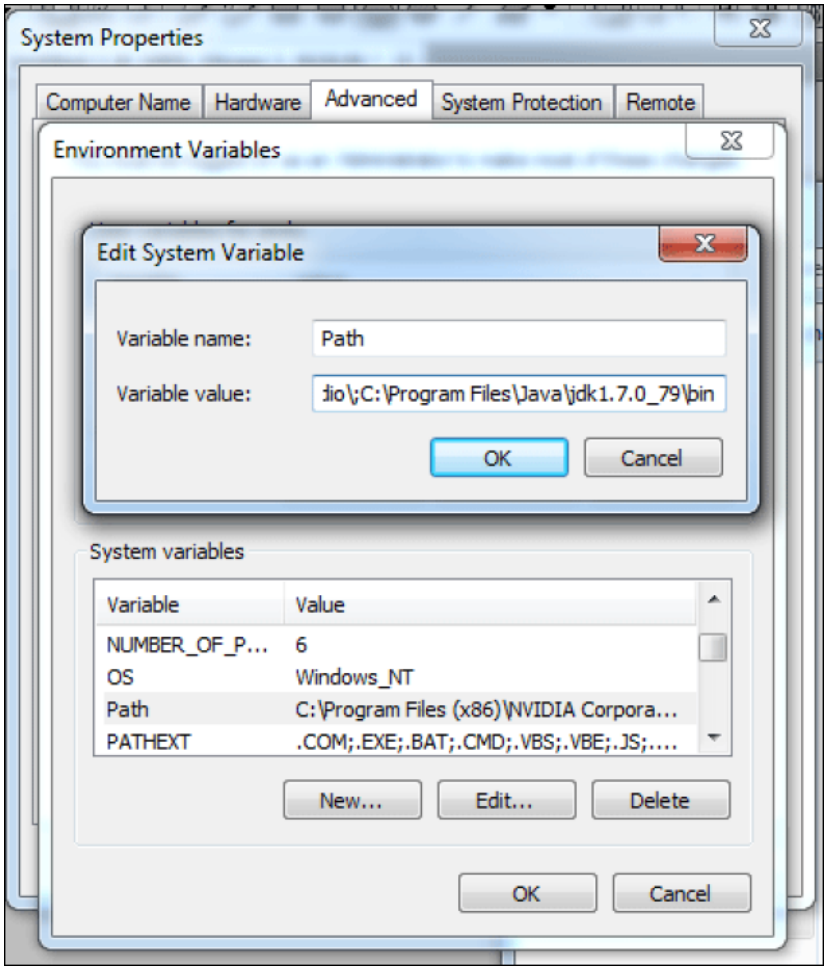



2. Conserver les valeurs par défaut, choisir l'emplacement du certificat à exporter et cliquer sur « Export » (Exporter).



## B. À l'aide de l'interface de ligne de commande Keytool

Java Keytool est un outil de ligne de commande permettant de générer des paires de clés publique-privée et de les stocker dans un fichier Java KeyStore. Avant de commencer à utiliser la ligne de commande keytool, le chemin d'accès au répertoire bin de Java doit être ajouté à la variable d'environnement « Path », comme suit :




Le fichier exécutable de Keytool s'appelle « keytool ». Pour l'exécuter, ouvrir une interface de ligne de commande (cmd, console, shell, etc.) et changer de répertoire : choisir le répertoire bin du kit de développement (SDK) de Java que vous avez installé. Taper « keytool » ; ce qui s'affiche devrait ressembler à la capture suivante :

```
C:\Program Files\Java\jdk1.8.0_111\bin>keytool
Key and Certificate Management Tool

Commands:
 -certreq                 Generates a certificate request
 -changealias             Changes an entry''s alias
 -delete                  Deletes an entry
 -exportcert              Exports certificate
 -genkeypair              Generates a key pair
 -genseckey               Generates a secret key
 -gencert                 Generates certificate from a certificate request
 -importcert              Imports a certificate or a certificate chain
 -importpass              Imports a password
 -importkeystore          Imports one or all entries from another keystore
 -keypasswd               Changes the key password of an entry
 -list                    Lists entries in a keystore
```

```
 -printcert              Prints the content of a certificate -printcertreq
Prints the content of a certificate request
 -printcrl               Prints the content of a CRL file
 -storepasswd            Changes the store password of a keystore

Use "keytool -command_name -help" for usage of command_name
```

La commande keytool supporte de nombreux paramètres, dont la configuration peut être difficile à retenir. Il sera donc judicieux de créer des scripts pour simplifier l'exécution des commandes keytool, ce qui simplifiera également la maintenance.

**Générer une paire de clés**

Générer une paire de clés publique-privée est l'une des tâches les plus courantes accomplies dans Keytool. La paire de clés générée est enregistrée dans un fichier Java KeyStore en tant que paire de clés autosignée. Le format usuel de la ligne de commande utilisé pour générer une paire de clés avec Keytool est le suivant :

```
-genkeypair {-alias alias} {-keyalg keyalg} {-keysize keysize} {-sigalg sigalg} [-dname dname] [-keypass keypass] {-
validity valDays} {-storetype storetype} {-keystore keystore} [-storepass storepass] {-providerClass
provider_class_name {-providerArg provider_arg}} {-v} {-protected} {-Jjavaoption}
```

Le keystore peut être généré à l'aide des instructions ci-dessous :

```
keytool -genkeypair -alias client -keystore ClientKeyStore.jks -keyalg RSA \
-dname "CN=Client, OU= Transport, O=ECE, L=GE, ST=GENEVA, C=CH, EMAILADDRESS=etir@un.org" \
-sigalg SHA256withRSA -validity 1000

Enter keystore password: keyStorePassword
Re-enter new password: keyStorePassword

Enter key password for <client>
        (RETURN if same as keystore password): certificatePassword
Re-enter new password: certificatePassword
```

**Exporter le certificat client**

La ligne de commande keytool permet également d'exporter des certificats stockés dans un keystore. On trouvera ci-dessous les modèles de commande de Keytool pour l'exportation de certificats :

```
-exportcert {-alias alias} {-file cert_file} {-storetype storetype} {-keystore keystore} [-storepass storepass] {-
providerName provider_name} {-providerClass provider_class_name {-providerArg provider_arg}} {-rfc} {-v} {-
protected} {-Jjavaoption}
```

Dans le dossier où se trouve le keystore que vous avez généré, extraire le certificat de clé publique à l'aide de la ligne de commande ci-après et l'envoyer aux adresses électroniques de la CEE (afin qu'il soit stocké dans le truststore du serveur de l'application) :

```
keytool -exportcert -keystore ClientKeyStore.jks -alias client -file client.cer -storepass certificatePassword
```

Appuyer sur la touche « Entrée » et le certificat sera stocké dans le même dossier que le keystore :

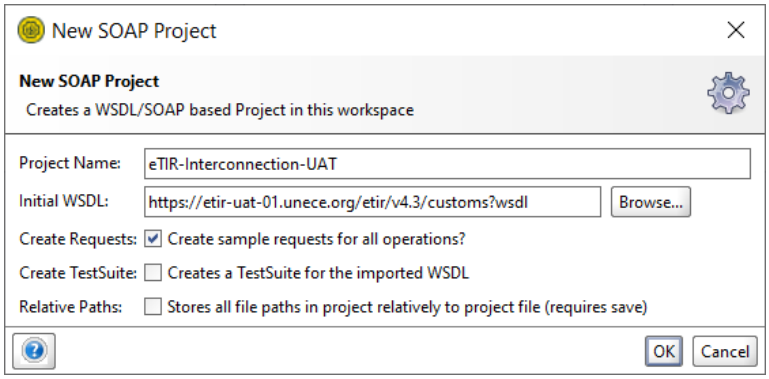
```
Certificate stored in file <client.cer>
```

# VI. Mini-guide SoapUI

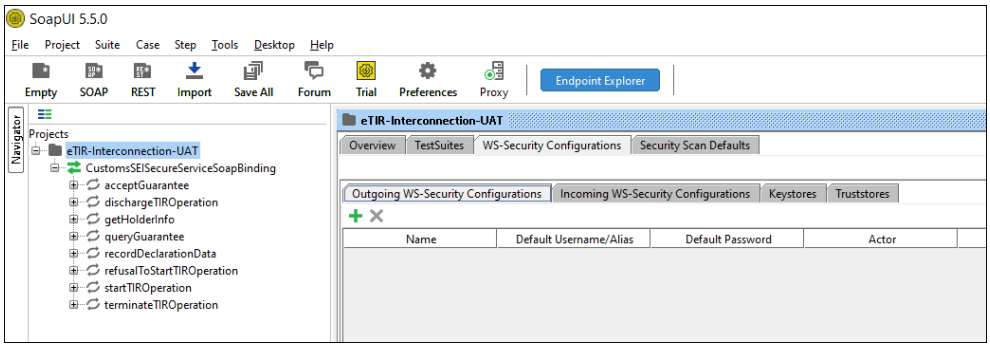
On trouvera ci-dessous les étapes de base à suivre pour installer et configurer l'outil de test SoapUI et pour envoyer et chiffrer une requête SOAP à l'aide de cet outil.

1. Télécharger et installer la version libre de l'application SoapUI depuis la page de téléchargement du site Web de SoapUI.

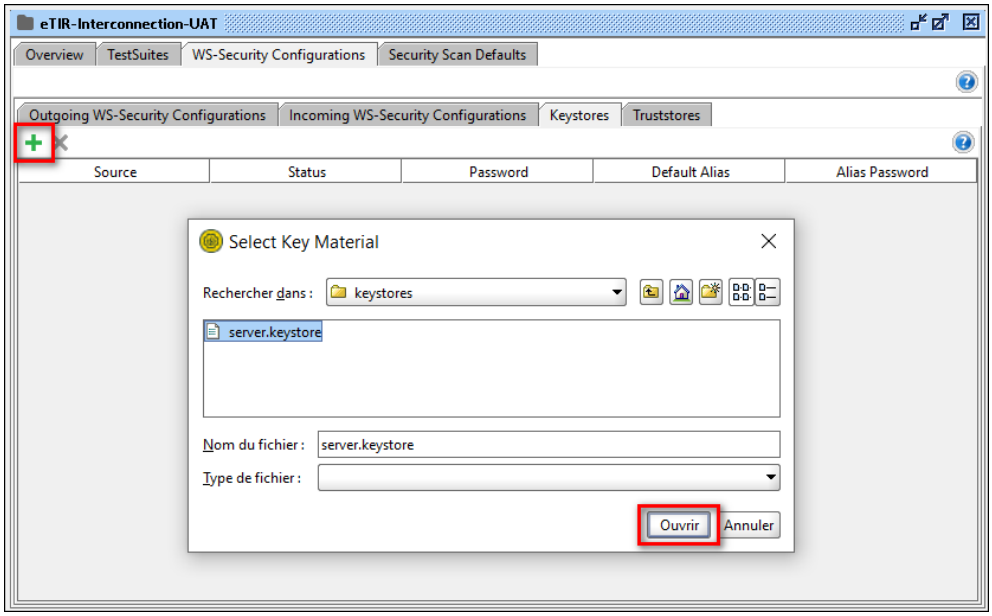
2. Lancer l'application SoapUI et créer un nouveau projet SOAP en cliquant sur « File », puis « New SOAP Project ». Saisir un nom de projet et définir l'URL de test pour le fichier WSDL : https://etir-uat-01.unece.org/etir/v4.3?wsdl.



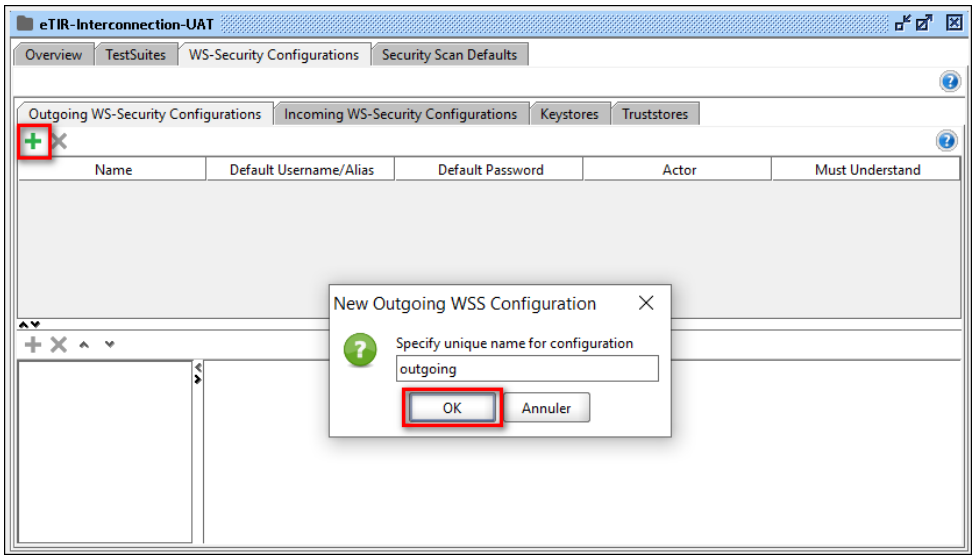

3. Une fois le projet créé, aller dans l'onglet « WS-Security Configurations » dans le menu « Show Project View » en faisant un clic droit sur le dossier du projet.



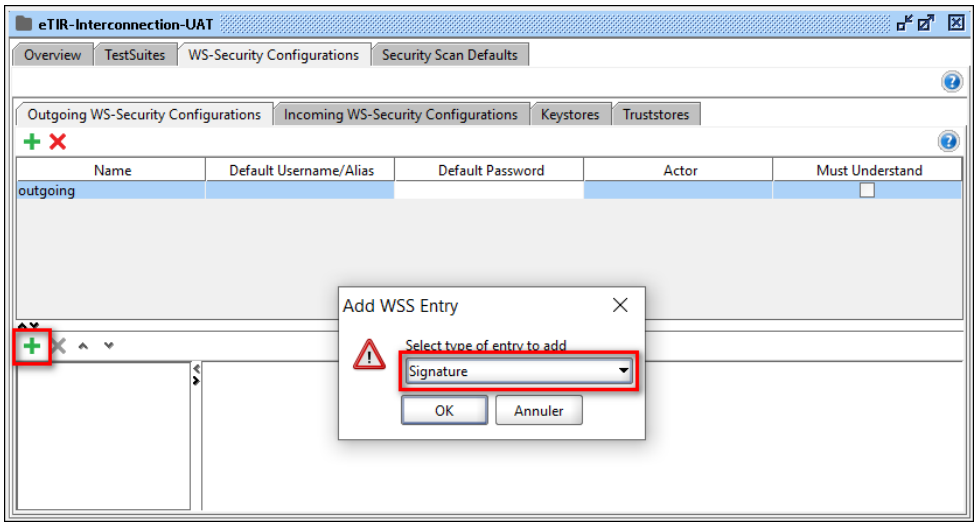

4. Cliquer sur le bouton « plus » pour ajouter un keystore en sélectionnant votre fichier keystore. Saisir le mot de passe puis cliquer sur « Ouvrir ».



5. Cliquer sur l'onglet « Outgoing WS-Security Configurations » pour afficher les configurations à appliquer aux messages sortants, dont les requêtes. Ajouter une nouvelle configuration de sortie.

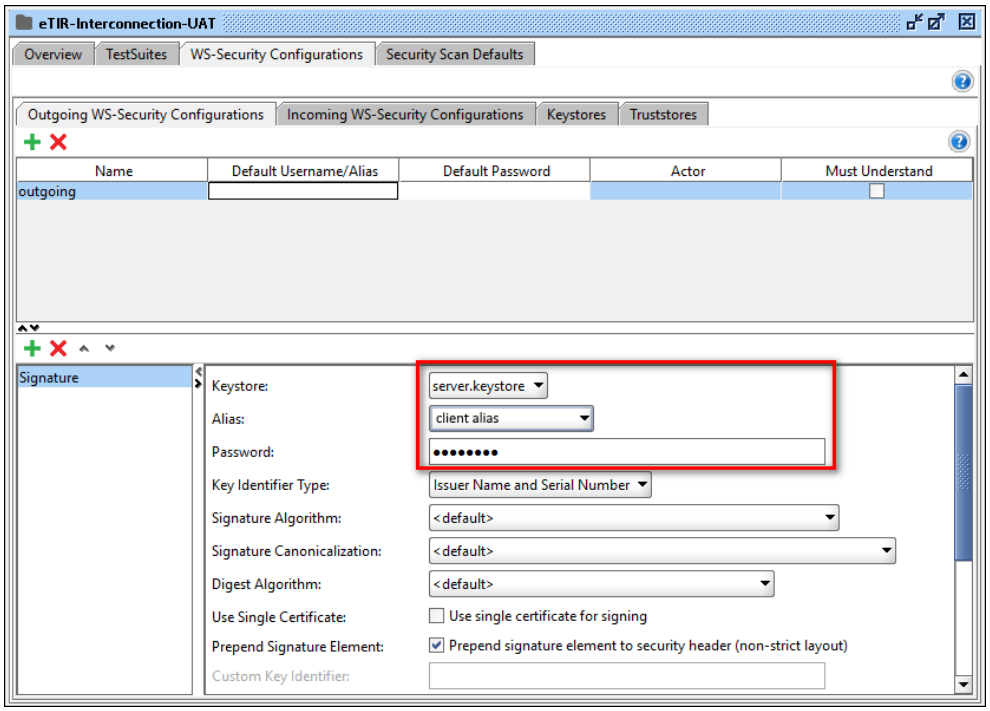



6. Ce type de configuration est utilisé pour le chiffrement, la signature et l'ajout d'une assertion SAML, d'un horodatage et d'un jeton de nom d'utilisateur à l'en-tête. On n'utilisera ici que la fonction signature. Ajouter une signature.

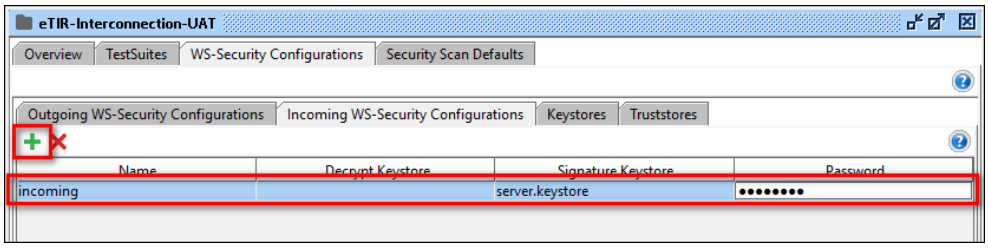


7. Choisir le keystore et l'alias de clé qui seront utilisés, ainsi que le mot de passe correspondant à l'alias, en configurant les champs suivants :

- **Keystore** : keystore à utiliser lors de la signature du message.
- **Alias** : alias (paire de clés) à utiliser lors de la signature du message.
- **Password** : mot de passe correspondant à l'alias.

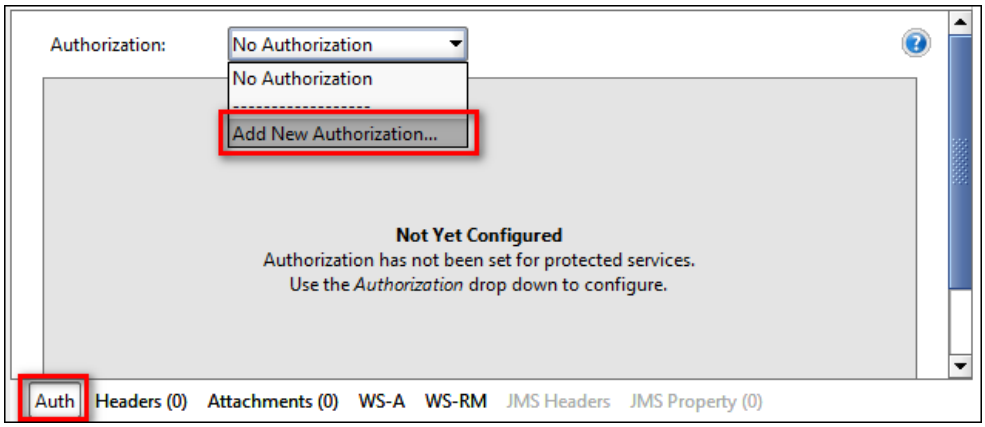



8. Cliquer sur l'onglet « Incoming WS-Security Configurations » pour afficher les configurations à appliquer aux messages entrants, dont les réponses. Saisir le nom du keystore et le mot de passe. Cette configuration permettra de valider la signature des réponses arrivant du serveur eTIR.

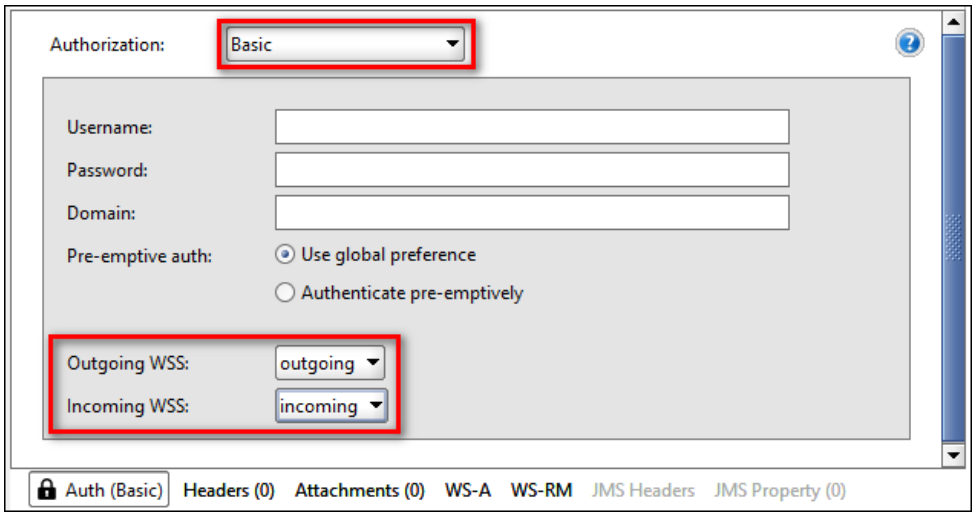



9. Ouvrir la requête SOAP en faisant un clic droit sur « queryGuarantee » dans l'arborescence située sur le côté gauche de l'application, puis sélectionner « New request » et saisir un nom.

10. Cliquer sur l'onglet « Auth » dans le coin inférieur gauche. Dans la liste déroulante du champ « Authorization », sélectionner « Add New Authorization… », puis sélectionner « Basic » pour une autorisation de base.




11. Dans la fenêtre de configuration qui apparaît, conserver les valeurs présélectionnées dans les champs « Outgoing WSS » et « Incoming WSS ».

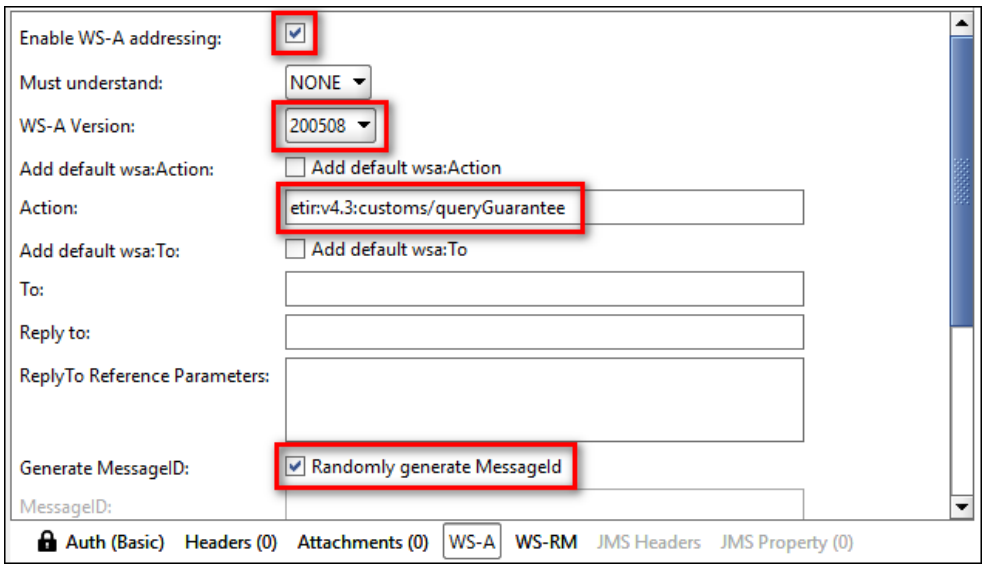


12. Aller sur l'onglet WS-A. Vérifier que les paramètres suivants sont correctement configurés :

a) Cocher la case "Enable WS-A addressing" ;

b) Cocher la case « Randomly generate MessageId » ;

c) Vérifier que le champ « Action » est correctement rempli ; son contenu doit ressembler à ce qui suit : https://etir-uat-01.unece.org/etir/v4.3.




13. Exécuter SoapUI sur le service Web en transmettant les messages SOAP au serveur. On pourra suivre l'exemple ci-dessous :

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:etir="etir:v4.3:customs" xmlns:etir1=
"etir:I5:v4.3">
   <soap:Header/>
   <soap:Body>
      <etir:queryGuarantee>
         <etir1:InterGov>
            <etir1:FunctionCode>9</etir1:FunctionCode>
            <etir1:ID>e393d591-963a-4cd1-9d4a-08467afcf524</etir1:ID>
            <etir1:TypeCode>I5</etir1:TypeCode>
            <etir1:ReplyTypeCode>00</etir1:ReplyTypeCode>
            <etir1:ObligationGuarantee>
               <etir1:ReferenceID>XC95000003</etir1:ReferenceID>
            </etir1:ObligationGuarantee>
         </etir1:InterGov>
      </etir:queryGuarantee>
   </soap:Body>
</soap:Envelope>
```

# VII. Exemple de requête SOAP utilisant WSS4J

L'exemple ci-après montre comment utiliser l'outil Web Services Security for Java (WSS4J) d'Apache pour créer un client SOAP. Le projet WSS4J est une implémentation Java des principales normes de sécurité pour les services Web, à savoir les spécifications OASIS WS-Security du comité technique d'OASIS sur la sécurité des services Web. WSS4J permet une implémentation du profil de jeton X.509. Dans l'exemple présenté, comme expliqué dans la section suivante, la signature des réponses reçues du système international eTIR peut être vérifiée et validée par des intercepteurs.

Il faut tout d'abord créer un certificat. Pour cela, on pourra suivre l'une des deux procédures présentées dans la partie V (Keystore : explication pas à pas pour la génération de paires de clés), à savoir celle qui repose sur l'application KeyStore Explorer ou celle qui repose sur l'interface de ligne de commande Keytool.

### Création du fichier de propriétés cryptographiques

WSS4J nécessite certaines propriétés cryptographiques, qui sont regroupées en trois sections, comme on le voit ci-dessous :

```
#Crypto properties
#General properties:
#WSS4J specific provider used to create Crypto instances. Defaults to
"org.apache.ws.security.components.crypto.Merlin".
#org.apache.ws.security.crypto.provider=
#Keystore properties:
#The location of the keystore
org.apache.ws.security.crypto.merlin.keystore.file=keystoreFile
#The password used to load the keystore. Default value is "security".
org.apache.ws.security.crypto.merlin.keystore.password=password
#Type of keystore. Defaults to: java.security.KeyStore.getDefaultType()), normally it is JKS
#org.apache.ws.security.crypto.merlin.keystore.type=JKS
#The default keystore alias to use, if none is specified.
org.apache.ws.security.crypto.merlin.keystore.alias=aliasName
```

### Gestionnaire de rappel de mot de passe

Comme indiqué dans la section précédente, le mot de passe du keystore figure dans le fichier de propriétés cryptographiques. Cependant, le mot de passe de la clé privée n'a pas encore été donné. Une propriété figure à cet effet dans le fichier de propriétés cryptographiques :

```
#The default password used to load the private key. Normally it is provided through a password-callback class
#org.apache.ws.security.crypto.merlin.keystore.private.password=
```

Pour plus de sécurité, une bonne pratique consiste à fournir le mot de passe par l'intermédiaire d'un gestionnaire de rappel de mot de passe. Le gestionnaire peut récupérer le mot de passe dans une base de données, dans un annuaire LDAP ou d'autres emplacements sécurisés. Dans le présent exemple, seul le mot de passe de la clé privée est affiché à partir de la classe du gestionnaire de rappel de mot de passe. On trouvera ci-dessous une implémentation de base du gestionnaire de rappel de mot de passe :

```
import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import org.apache.ws.security.WSPasswordCallback;

public class ServerPasswordCallback implements CallbackHandler {

  public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {

     for (int i = 0; i < callbacks.length; i++) {
      WSPasswordCallback pc = (WSPasswordCallback) callbacks[i];
      pc.setPassword("key-pass");
    }
  }
}
```

### Intercepteurs WSS4J

La dernière étape consiste à définir les intercepteurs WSS4J entrants et sortants du côté du client. La requête SOAP nécessite deux intercepteurs. Les beans des intercepteurs configurent le reste des paramètres de sécurité, à savoir, par exemple, si le message SOAP doit contenir un horodatage et une signature, les parties qui doivent être signées, l'algorithme à utiliser, mais aussi le type d'élément BinarySecurityToken et le type de référence, le pointeur pour la classe du gestionnaire de rappel de mot de passe, etc. Tous ces paramètres sont configurés sous forme de paires clé/valeur dans une table de mappage. L'ensemble de la table de mappage est répertorié sous forme de balises WSHandler sur la page de configuration de WSS4J. Il convient de noter que bon nombre de clés ou réglages ont des valeurs par défaut.

### Intercepteurs WSS4J entrants et sortants − configuration XML-Spring

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:jaxws="http://cxf.apache.org/jaxws" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
 http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">

 <bean id="logInBound" class="org.apache.cxf.interceptor.LoggingInInterceptor" />
 <bean id="logOutBound" class="org.apache.cxf.interceptor.LoggingOutInterceptor" />
 <jaxws:client id="customsClient" serviceClass="org.unece.etir.ws.cusc.CustomsSEISecure"
 address="https://ece-dev-etir.unece.org/etir/services/CustomsToETIR-1">
 <jaxws:inInterceptors>
 <ref bean="logInBound" />
 <ref bean="inbound-security" />
 </jaxws:inInterceptors>
 <jaxws:outInterceptors>
 <ref bean="logOutBound" />
 <ref bean="outbound-security" />
 </jaxws:outInterceptors>
 </jaxws:client>

 <!-- WSS4JOutInterceptor for signing outbound SOAP messages -->
    <bean class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor" id="outbound-security">
        <constructor-arg>
            <map>
                <entry key="action" value="Signature"/>
                <entry key="user" value="client"/>
                <entry key="signaturePropFile" value="client-crypto.properties"/>
                <entry key="passwordCallbackClass" value="client.ClientPasswordCallback"/>
                <entry key="signatureParts"
                value="{Element}{http://schemas.xmlsoap.org/soap/envelope/}Body"/>
            </map>
        </constructor-arg>
    </bean>

    <!--  WSS4JInInterceptor for validating the signature of inbound SOAP messages -->
    <bean class="org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor" id="inbound-security">
        <constructor-arg>
            <map>
                <entry key="action" value="Signature"/>
                <entry key="signaturePropFile" value="client-crypto.properties"/>
                <entry key="passwordCallbackClass" value="client.ClientPasswordCallback"/>
            </map>
        </constructor-arg>
    </bean>

</beans>
```

### Intercepteurs WSS4J entrants et sortants − configuration Java

WSS4JOutInterceptor signant les messages SOAP sortants :

```
Map<String, Object> propsOut = new HashMap<String, Object>();
propsOut.put(WSHandlerConstants.ACTION, WSHandlerConstants.SIGNATURE);
propsOut.put(WSHandlerConstants.SIGNATURE_PARTS, "{Element}{http://www.w3.org/2003/05/soap-envelope}Body;");
propsOut.put(WSHandlerConstants.PW_CALLBACK_CLASS, PasswordCallback.class.getName());
propsOut.put(WSHandlerConstants.USER, "client");
propsOut.put(WSHandlerConstants.SIG_PROP_FILE, "META-INF/client-security.properties");
```

WSS4JInInterceptor validant la signature des messages SOAP entrants :

```
Map<String, Object> propsIn = new HashMap<String, Object>();
propsIn.put(WSHandlerConstants.ACTION, WSHandlerConstants.SIGNATURE);
propsIn.put(WSHandlerConstants.PW_CALLBACK_CLASS, PasswordCallback.class.getName());
propsIn.put(WSHandlerConstants.SIG_PROP_FILE, "META-INF/client-security.properties");

WSS4JOutInterceptor wssOut = new WSS4JOutInterceptor(propsOut);
WSS4JInInterceptor wssIn = new WSS4JInInterceptor(propsIn);
Client client = ClientProxy.getClient(port);
Endpoint endpoint = client.getEndpoint();
endpoint.getOutInterceptors().add(wssOut);
endpoint.getInInterceptors().add(wssIn);
endpoint.getInInterceptors().add(new LoggingInInterceptor());
endpoint.getOutInterceptors().add(new LoggingOutInterceptor());
```