DIRECTION GÉNÉRALE DES IMPÔTS
DIRECTION GÉNÉRALE DE LA COMPTABILITÉ PUBLIQUE

Utilisation du client Java du SVC de niveau 2.5

060835
Version 0.01

THALES
INFORMATION SYSTEMS

Circuit de validation

	Nom	Organisation	Date	Visa
Rédigé par :	Jérôme Lubrez	CP SVC	24/11/2006	
Vérifié par :	Sébastien Levesque	Responsable développement SVC	NN/11/2006	
Approuvé par :		Directeur de projet		

Historique des évolutions

Ver	Date	Auteur	Justificatif
0.01	24/11/2006	J. Lubrez	Création du document par copie du document de la version précédente SVC 2.

Sommaire

1 Présentation

Le schéma XML des requêtes envoyées et des réponses n'ont pas changé entre le SVC de niveau 2 et 2.5. Pour différencier les répondeurs SVC de niveau 2 et de niveau 2.5, le message XML possède un élément svc avec un attribut *version* dans la réponse : < *svc version="2.50" />*

Les clients java de niveau 2 et de niveau 2.5 sont supportés par les répondeurs du SVC 2.5.

Une application souhaitant interroger le SVC de niveau 2.5 pour obtenir la validation d'un certificat doit effectuer la démarche suivante :

- équipe de développement : disposer de la librairie cliente (fichier `allclientsvc.jar`),
- maîtrise d'ouvrage : définir, conjointement avec le bureau SI3, le ou les domaines de confiance et la ou les politiques de validation applicables,
- équipe d'exploitation : paramétrer le SVC de niveau 2 pour y configurer le ou les domaines de confiance et la ou les politiques de validation définis.

Le présent document s'adresse à l'équipe de développement. Il présuppose que les domaines de confiance et les politiques de validation aient été définis et que les OID de ces éléments soient connus.

Pour mémoire, de façon générale,

- un OID de domaine de confiance est de la forme : 1.2.250.1.131.1.5.4.7.1.dc (où dc est un entier positif),
- un OID de politique de validation est de la forme : 1.2.250.1.131.1.5.4.6.1.pv.v (avec pv un entier positif correspondant à l'identifiant de la politique de validation et v sa version).

Une politique de validation définit les contrôles à effectuer. Les contrôles disponibles sont les contrôless VERCER (vérification du certificat), VALREV (vérification de non-révocation), VERCAC (vérification des chemins de certification) et SIGREP (signature de la réponse).

La librairie `allclientsvc.jar` fournit la classe ClientSVC avec les méthodes publiques suivantes :

- méthodes `sendXmlRequest` renvoyant un flux XML sous forme de String :
  - si l'URL est passée dans l'appel :

```
public static String sendXmlRequest(String urlServer, X509Certificate x509,
String oidTrustdom, String oidValidationpolicy, String nonce) throws
IOException
```

  - si l'URL est positionnée dans un fichier de configuration :

```
public static String sendXmlRequest(X509Certificate x509, String
oidTrustdom, String oidValidationpolicy, String nonce) throws IOException
```

- méthodes sendAndParseXmlRequest renvoyant un objet SvcSimpleResponse qui offre des accesseurs aux différents champs du message XML :
  - si l'URL est passée dans l'appel :

```
public static SvcSimpleResponse sendAndParseXmlRequest(String urlServer,
X509Certificate x509, String oidTrustdom, String oidValidationpolicy,
String nonce) throws SAXException, IOException
```

- si l'URL est positionnée dans un fichier de configuration :

```
public static SvcSimpleResponse sendAndParseXmlRequest(X509Certificate
x509, String oidTrustdom, String oidValidationpolicy, String nonce) throws
SAXException, IOException
```

- méthode de vérification de signature :

```
public static boolean verify(SvcSimpleResponse sp) throws
NoSuchAlgorithmException, InvalidKeyException, SignatureException
```

Pour des informations sur les exceptions, voir la classe `java.security.signature` de Sun (méthode `verify` de cette classe).

Note : pour plus d'informations techniques il est possible de se référer à la Javadoc du client Java.

2 Fichier de configuration

L'URL des répondeurs SVC peut être positionnée dans l'appel Java ou dans un fichier de configuration externe.

Ce fichier doit s'appeler `clientSvc.properties` et être situé dans le `classpath`.

Il doit contenir l'URL sous le format suivant :

```
#----------------------------------------
#fichier de parametres du client svc 2.0
#----------------------------------------
#url du serveur svc (Repondeur SVC)
#url=http://192.168.100.10/dgi/srvsvc

url=http://172.20.5.45/dgi/srvsvc
```

3 Description de l'interface (API)

3.1 Paramètres d'appel

Quelle que soit la méthode utilisée, les paramètres d'appel sont les suivants :

Paramètre	Type Java	Signification
urlServer	String	URL du serveur Répondeur (optionnel)
x509	X509Certificate	certificat à vérifier
oidTrustdom	String	OID du domaine de confiance
oidValidationpolicy	String	OID de la politique de validation
Nonce	String	chaîne « nonce » que le serveur renvoie à l'identique

3.2 Données retournées

3.2.1 sendXmlRequest

Le contenu de la chaîne XML retournée dépend des contrôles définis dans la politique de validation passée en paramètre.

```
<?xml version="1.0"? encoding="UTF-8">
<certvalidation>
<svcresponse>
     <svc version="2.50"/>
     <certid>numéro de série du certificat</certid>
     <validationpolicy>OID (identifiant.version) de la politique de
validation utilisée</validationpolicy>
     <responsestatus>validated|rejected</responsestatus>
     <error type="S|F|T" code="xxx">message d'erreur</error>
     <producedat>date système de la réponse</producedat>
     <nonce>valeur du nonce (vide en absence de nonce)</nonce>
     bloc revocation si appel au service VALREV
     bloc certpath si appel au service VERCAC
</svcresponse>
bloc signature si appel au service SIGREP
</certvalidation>
```

Les données sont de façon systématique renvoyés dans cet ordre.

La requête HTTP de réponse contient le `content-type` suivant : « application/svc-response ».

Le statut de la réponse (bloc <responsestatus></responsestatus>) est « `validated` » si le certificat est validé. Il est « `rejected` » dans le cas d'une erreur fonctionnelle ou d'une erreur technique.

Le bloc XML <error></error> est toujours présent. En l'absence d'erreur, il contient les données (type = S, code = 0, message vide). En cas d'erreur technique, le type d'erreur est T. En cas d'erreur fonctionnelle, le type d'erreur est F. Dans les deux cas, le message d'erreur `est` libellé en français.

Si la politique de validation fait appel au module VALREV, le bloc suivant est présent dans la réponse :

```
     <revocation>
          <version>version du protocole OCSP</version>
          <responderid>identifiant du répondeur OCSP</responderid>
          <certstatus>statut du certificat</certstatus>
          <thisupdate>date thisupdate présente dans la CRL</thisupdate>
          <nextupdate>date nextupdate présente dans la CRL</nextupdate>
          <revocationtime>date de révocation</revocationtime>
          <downloadtime>date de récupération de la CRL</downloadtime>
          <nocheck></nocheck>
     </revocation>
```

Le bloc XML <revocation></revocation> n'est présent que si un appel à VALREV a été effectué. S'il est présent, tous les champs du niveau inférieur sont présents.

Si la politique de validation fait appel au module VERCAC, le bloc suivant est présent dans la réponse :

```
<certpath>
      <cert order="1">certificat de l'AC racine (PEM)</cert>
      ...
      <cert order="n">certificat de l'AC terminale (PEM)</cert>

</certpath>
```

Le bloc XML <certpath></certpath> n'est présent que si un appel à VERCAC a été effectué. S'il est présent, tous les champs du niveau inférieur sont présents. Les certificats renvoyés sont systématiquement ordonnés dans l'ordre de la chaîne de certification, de l'AC racine à l'AC terminale.

Si la politique de validation fait appel au module SIGREP, le bloc suivant est présent dans la réponse :

```
<signature>signature (optionnelle) de l'ensemble de la réponse par
l'instance de SVC ayant effectué la validation</signature>
<certs>
      <certificate>certificat de l'instance de SVC</certificate>
      <cacertificate>certificat de son AC terminale</cacertificate>
</certs>
```

Le bloc XML de signature n'est présent que si un appel à SIGREP a été effectué. Il consiste en l'ajout de la signature de l'empreinte SHA-1 de la réponse <svcresponse></svcresponse>.

Dans les cas suivants :

- domaine de confiance inconnu,
- politique de validation inconnue,
- politique de validation et domaine de confiance non associés,
- aucune version active de la politique de validation demandée,

la réponse donne le message d'erreur correspondant et n'est pas signée. La réponse est décrite ci-dessous :

```
<?xml version="1.0"? encoding="UTF-8">
<certvalidation>
<svcresponse>
     <svc version="2.50"/>
     <certid>numéro de série du certificat</certid>
     <validationpolicy></validationpolicy>
     <responsestatus>rejected</responsestatus>
<error type="F" code="101"> Politique de validation ou domaine de confiance inconnus ou non associés, ou aucune version active de cette politique de validation</error>
     <producedat>date système de la réponse</producedat>
     <nonce>valeur du nonce (vide en absence de nonce)</nonce>
</svcresponse>
</certvalidation>
```

3.2.2 sendAndParseXmlRequest

La classe `SvcSimpleResponse` propose les accesseurs suivants :

Accesseur	Signification
SvcSimpleResponse(String xml) throws SAXException	constructeur
public String getSvcVersion()	version du service SVC
public BigInteger getCertId()	numéro de série du certificat de la requête
public String getValidationPolicy()	OID de la politique de validation que le serveur a effectivement traitée
public String getResponseStatus()	[validated | rejected]
public String getErrorType()	[S | F | T] : (S en cas de succès, F en cas d'erreur fonctionnelle ou T en cas d'erreur technique)
public int getErrorCode()	Code d'erreur
public String getErrorMessage()	Message d'erreur
public Date getProduceAt()	date de production de la réponse SVC
public String getNonce()	identifiant nonce passé dans l'appel
public int getCertStatus()	statut de révocation du certificat
public Date getThisUpdate() {	date thisUpdate de la LCR
public Date getNextUpdate()	date nextUpdate de la LCR
public Date getRevocationTime()	date de révocation du certificat
public Date getDownloadTime()	date de récupération effective de la LCR par le serveur
public String getNoCheck()	champ noCheck (null si vide)
public Vector getCertpath()	vecteur de X509certificate des certificats des AC de la chaîne de certification

public String getSignature()	signature du message XML, en chaîne
public byte[] getBytesSignature()	signature du message XML, en octets
public X509Certificate getSvcCertificate()	certificat du répondeur SVC interrogé
public X509Certificate getCaCertificate()	certificat de l'AC émettrice du certificat du répondeur SVC interrogé
public String getXmlSvcresponse()	corps de la reponse svc (contenu entre les balises <svcresponse> et </svcresponse>, balises incluses
public String getXmlCertValidationresponse()	corps de la reponse svc (contenu entre les balises <certvalidation> et </ certvalidation>, balises incluses
public String getXmlResponse()	intégralité de la reponse svc

3.3 Exceptions

3.3.1 sendXmlRequest

Emission d'une IOException :

en cas de problème de connexion au serveur, ou si le code d'erreur de la réponse HTTP est supérieur à 400, ou en cas de problème dans l'encodage du certificat :

3.3.2 sendAndParseXmlRequest

Emission d'une IOException :

en cas de problème de connexion au serveur, ou si le code d'erreur de la réponse HTTP est supérieur à 400, ou en cas de problème dans l'encodage du certificat :

Emission d'une SAXException :

en cas de problème lors de l'analyse de la réponse XML (parsing).

4 Exemple d'utilisation

```
static public void main(String arg[]) {
	try {
	//url du répondeur
	String url = http://valcer/dgi/srvsvc;
	//certificat de test
	File file = new File("C:/usr/java/eclipse/workspace/Common/certs/cert_test2.pem") ;
	int len = (int)file.length() ;
	FileInputStream fis = new FileInputStream(file) ;
	CertificateFactory cf = CertificateFactory.getInstance("X.509", "SUN") ;
	X509Certificate x509 = (X509Certificate) cf.generateCertificate(fis) ;
	fis.close() ;
	//domaine de confiance 42
	String dc = "1.2.250.1.131.1.5.4.7.1.42" ;
	//politique de validation 22 en version 9
	String pv = "1.2.250.1.131.1.5.4.6.1.22.9" ;
	//nonce
	String nonce = "azerty77" ;

	//test avec flux xml
	String reponseXml = ClientSVC.sendXmlRequest(url,x509,dc,pv,nonce) ;
	System.out.println(reponseXml);

	//test avec l'objet SVCSimpleResponse (avec quelques accesseurs seulement)
	SVCSimpleResponse sp = ClientSVC.sendAndParseXmlRequest (url,x509,dc,pv,nonce) ;
	System.out.println("Identifiant du certificat:\t\t"+sp. getCertId()) ;
	System.out.println("Type d'erreur:\t\t"+sp.getErrorType());
	System.out.println("Code d'erreur:\t\t"+sp.getErrorCode());
	System.out.println("Message d'erreur:\t\t"+sp.getErrorMessage());
	System.out.println("Vérification de la signature:\t\t"+ClientSVC.verify(sp)) ;
	}
	catch(Exception e) {
		e.printStackTrace() ;
	}
}
```