

Programme d'exemple CPS3
Documentation
V1.1 du 29/05/2011

Documents de référence

1. PKCS #11 v2.20: Cryptographic Token Interface Standard

Sommaire

1	Introduction	5
2	Présentation du programme d'exemple	6
3	Installation	8
4	Démarrage du programme	9
5	Module Fonctions PKCS#11	10
5.1	Conventions	10
5.1.1	Fonctions	10
5.1.2	Code retour	10
5.1.3	Indications techniques	11
5.1.4	Librairie PKCS#11	11
5.2	Général	12
5.2.1	Initialiser la librairie	12
5.2.2	Terminer la librairie	12
5.2.3	Obtenir des informations sur la librairie	13
5.3	Slots et cartes	14
5.3.1	Lister les slots	14
5.3.2	Obtenir des informations sur un slot	15
5.3.3	Obtenir des informations sur une carte	17
5.3.4	Attendre un évènement slot	19
5.3.5	Obtenir la liste des algorithmes supportés par la carte	20
5.3.6	Obtenir des informations sur un algorithme	21
5.3.7	Initialiser le code porteur de la carte	23
5.3.8	Modifier le code porteur de la carte	24
5.4	Sessions	26
5.4.1	Ouvrir une session	26
5.4.1.1	Accès Lecture seule	26
5.4.1.2	Accès Lecture/Ecriture	26
5.4.1.3	Résultats	26
5.4.1.4	Passage d'une session à l'autre	27
5.4.2	Obtenir des informations sur la session	27
5.4.3	Fermer la session	28
5.4.4	Fermer toutes les sessions actives	29
5.4.5	S'authentifier	30
5.4.6	Se "désauthentifier"	32
5.5	Objets	33
5.5.1	Gestion d'objets	33
5.5.1.1	Créer l'objet de données applicatives	33
5.5.1.2	Copier l'objet de données applicatives	34
5.5.1.3	Supprimer l'objet de données applicatives	35
5.5.1.4	Obtenir la taille de l'objet de données applicatives	36
5.5.1.5	Afficher un attribut de l'objet de données applicatives	37
5.5.1.6	Modifier un attribut de l'objet de données applicatives	39
5.5.2	Recherche d'objets	40
5.6	Empreintes	42
5.6.1	Générer l'empreinte d'un message court	42
5.6.2	Générer l'empreinte d'un fichier	44
5.7	Signatures	47

5.7.1	Signer un message court.....	47
5.7.2	Signer un fichier.....	49
5.8	Vérification de données signées.....	52
5.8.1	Vérifier la signature du message court.....	52
5.8.2	Vérifier la signature du fichier.....	54
5.9	Générateur de nombre pseudo-aléatoire.....	56
5.9.1	Normale.....	56
5.9.2	Couplée à une « graine » (seed).....	57
6	Module Traitements spécifiques.....	59
6.1	Authentification.....	59
6.1.1	Statut d'authentification par rapport à une carte.....	59
6.1.2	Assistant d'authentification.....	60
6.1.3	Après retrait carte.....	61
6.1.4	Après retrait lecteur.....	62
6.2	Gestion du code porteur.....	64
6.2.1	Etat de saisie du code porteur.....	64
6.2.2	Recyclage de la carte avec le code de déblocage (PUK).....	66
6.3	Gestion de jetons.....	67
6.3.1	Lecture d'un jeton d'établissement.....	67
6.3.2	Modification d'un jeton d'établissement.....	67
7	Description des sources.....	70
7.1	Sources Java.....	70
7.1.1	Présentation.....	70
7.1.2	Prise en main.....	70
7.1.3	Description.....	71
7.1.3.1	Les objets manipulés.....	72
7.1.3.2	Code relatif au module Fonctions PKCS#11.....	72
7.1.3.3	Code relatif au module Traitement spécifiques.....	73
7.1.3.4	Classe utilitaire.....	73
7.1.4	Génération du fichier JAR.....	73
7.1.4.1	Avec le script de génération.....	73
7.1.4.2	Depuis Eclipse.....	73
7.1.4.3	Utilisation.....	74
7.2	Sources C.....	75
7.2.1	Présentation.....	75
7.2.2	Prise en main.....	75
7.2.2.1	Commun.....	75
7.2.2.2	Windows 32bits.....	75
7.2.2.3	Linux.....	75
7.2.2.4	Remarque.....	75
7.2.3	Description.....	75
7.2.3.1	Points d'entrées des traitements spécifiques (Couche JNI).....	76
7.2.3.2	Traitements spécifiques et fonctions annexes (Couche « pure » C).....	78
7.2.4	Génération.....	79
7.2.4.1	Windows.....	79
7.2.4.2	Linux.....	79
	Table des illustrations.....	81

1 Introduction

La Direction de l'ASIP Santé a acté la réalisation d'une nouvelle version de ses librairies cryptographiques (ou CryptoLib) afin d'intégrer l'accès à la CPS3. Cette CryptoLib offrira une compatibilité ascendante pour la prise en compte de la CPS2ter par les applications actuellement sur le terrain.

Cette CryptoLib est disponible pour les environnements Windows et Linux.

Un objectif secondaire est d'engager la fin de vie des API-CPS ; lors de la migration, tous les développements CPS3 devront se faire en s'appuyant sur la nouvelle CryptoLib CPS3 adoptant le standard PKCS#11.

Le présent document s'attache à :

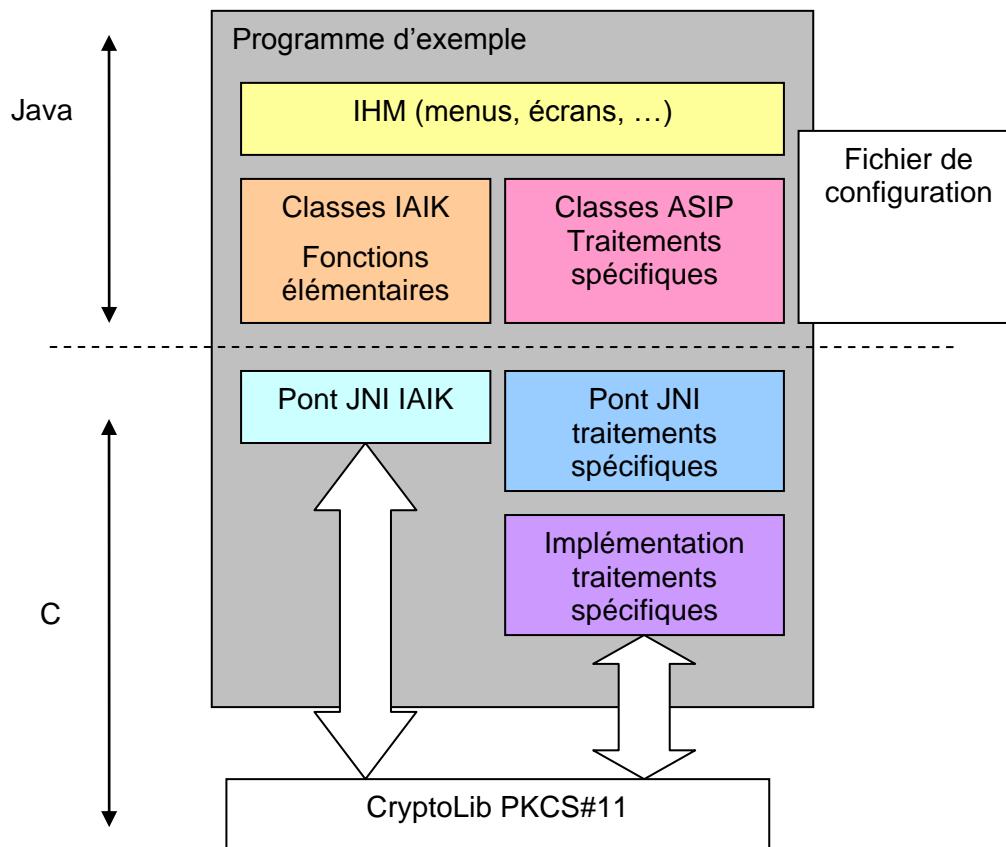
- Présenter les grandes lignes du programme d'exemple de la CryptoLib de la CPS3.
- Décrire la procédure d'installation.
- Décrire l'utilisation de chacune des fonctionnalités du programme d'exemple.
- Décrire le contenu des sources.

2 Présentation du programme d'exemple

Le programme comprend 2 principaux modules :

- Le module *Fonctions PKCS#11* qui permet de tester les fonctions élémentaires de l'interface PKCS#11 de la CryptoLib.
- Le module *Traitements spécifiques* qui permet de tester des traitements plus complexes, qui combinent plusieurs appels aux fonctions PKCS#11 de la CryptoLib, dans le but de réaliser une opération spécifique (ex : authentification après retrait lecteur/carte).

L'architecture générale du programme d'exemple est indiquée dans le schéma ci-dessous :



Le programme d'exemple comporte une couche écrite en java pour réaliser tous les traitements de l'interface Homme Machine (IHM) tels que la gestion des menus, l'affichage des écrans. La couche graphique est portable sur l'ensemble des systèmes cibles (Windows, Linux). Cette couche accède à l'ensemble des fonctions élémentaires de l'interface PKCS#11 via les classes du projet opensource IAIK. La couche IAIK comporte d'une part des classes proposant dans le monde java l'équivalent de l'interface PKCS#11 et d'autre part un pont JNI (Java Native Interface) qui assure la liaison avec la librairie CryptoLib PKCS#11 qui elle dépend du système cible.

Le programme d'exemple propose également dans le mode java une interface vers les traitements spécifiques. Il s'agit de classes développées par l'ASIP qui au travers d'un pont

JNI accèdent à l'implémentation en langage C de tous les traitements usuels qu'une application peut avoir à réaliser sur l'interface PKCS#11.

Le programme d'exemple est constitué des composants suivants :

A l'exception des classes IAIK, l'ensemble des classes du programme d'exemple et le fichier de configuration sont regroupées dans une archive programmeExempleCPS3.jar.

La couche IAIK se décompose en une archive iaikPkcs11Wrapper.jar et une librairie native pkcs11wrapper.dll (Windows) ou libpkcs11wrapper.so (Linux) qui contient le pont JNI IAIK.

Le pont JNI et l'implémentation des traitements spécifiques sont regroupés dans la librairie native pgm_exemple_jni_w32.dll (Windows) ou libpgm_exemple_jni_lux.so (Linux).

Il est important de noter que les deux interfaces proposées par le programme d'exemple (fonctions élémentaires PKCS#11 et traitements spécifiques) sont totalement indépendantes l'une de l'autre. Il n'est donc pas nécessaire de disposer des composants de la couche IAIK pour réutiliser le code d'exemple des traitements spécifiques et réciproquement.

3 Installation

L'installation du programme d'exemple se limite à la copie d'un répertoire sur le disque du poste cible.

Les composants du programme d'exemple se présentent sous la forme d'un répertoire contenant :

- Le fichier jar : programmeExempleCPS3.jar
- Le fichier de lancement du jar : lancementJAR.bat (Windows) ou lancementJAR.sh (Linux)
- Les librairies natives dont le programme a besoin :
 - cps3_pkcs11_w32.dll (Windows) ou libcps3_pkcs11_lux.so (Linux): La librairie PKCS#11 de la carte CPS3.
 - pkcs11wrapper.dll (Windows) ou libpkcs11wrapper.so (Linux) : librairie native de la couche iaik utilisée pour le module Fonctions PKCS#11
 - pgm_exemple_jni_w32.dll (Windows) ou libpgm_exemple_jni_lux.so (Linux) : librairie native utilisée par le module des Traitements spécifiques.
- Un répertoire /lib contenant les fichiers jar dont le programme a besoin :
 - iaikPkcs11Wrapper.jar : Couche IAIK utilisée pour le module Fonctions PKCS#11.
 - log4j-1.2.16.jar : Qui permet au programme d'exemple de logger, en cas d'erreur.

Pour lancer le programme d'exemple, il suffit de double cliquer sur le fichier lancementJAR.bat sous Windows ou de lancer le script lancementJAR.sh dans une fenêtre de Terminal sous Linux.

4 Démarrage du programme

Au démarrage du programme d'exemple, ce dernier tente de charger la librairie PKCS#11 par défaut. La librairie définie est dans le fichier de configuration du programme, il s'agit de la librairie PKCS#11 de la carte CPS3. Si un problème intervient au chargement de cette librairie, l'utilisateur est notifié et est invité à charger une librairie depuis son système de fichier.

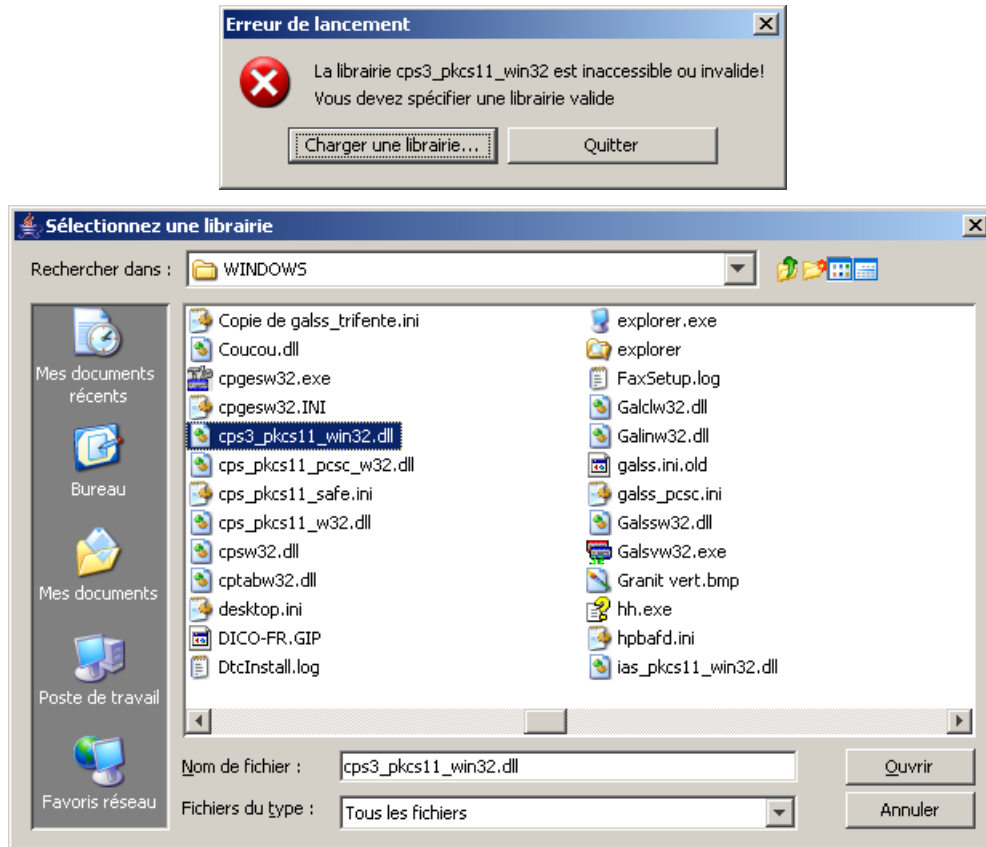


Figure 1 : Erreur de lancement et chargement explicite de librairie PKCS#11

Tant que le chargement de la librairie n'est pas possible (librairie inaccessible ou invalide) les boîtes de dialogues d'erreur et de sélection de librairie apparaîtront. Si l'utilisateur souhaite quitter le programme il peut cliquer sur le bouton *Quitter* de la boîte de dialogue d'erreur, ou sur le bouton *Annuler* de la boîte de dialogue de sélection de la librairie.

5 Module Fonctions PKCS#11

Dans la spécification RSA, les fonctions de l'interface PKCS#11 sont regroupées par catégories. L'interface graphique reprend ces catégories dans son menu *Fonctions PKCS#11*.

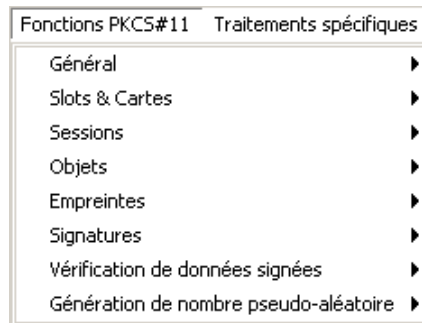


Figure 2 : Menu Fonctions PKCS#11

5.1 Conventions

5.1.1 Fonctions

Il est à noter que les fonctions s'activent dans l'interface dès que les conditions d'utilisation sont remplies. Par exemple au lancement du programme d'exemple, seule la fonction d'initialisation de la librairie est accessible.

En effet selon la spécification PKCS#11 la seule fonction qu'un programme peut invoquer en premier est C_Initialize pour l'initialisation.

Une fois que la fonction d'initialisation a été réalisée, certains éléments des menus deviennent accessibles.

Ce document prend en compte la logique PKCS#11 et les items de menu non accessibles sont représentés désactivés (en gris) tandis que les items accessibles sont représentés activés (en noir).

Toutefois, une option permet de déverrouiller le menu, pour ignorer cette logique :

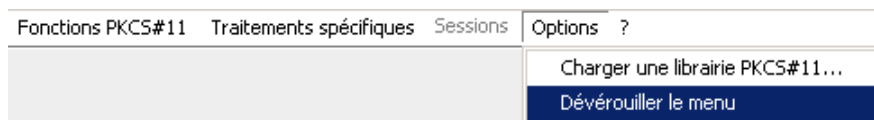


Figure 3 : Le menu est verrouillé

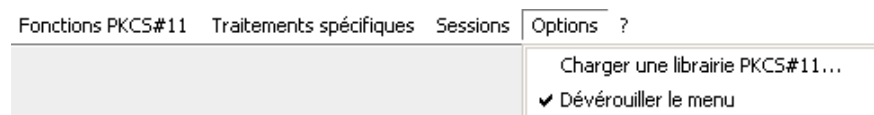


Figure 4 : Le menu est déverrouillé

5.1.2 Code retour

Les codes retour mentionnés dans la description des fonctions sont ceux des fonctions PKCS#11. Le programme, pour plus de cohérence, est conçu de façon à éviter de rencontrer un certain nombre de codes retour signalant une erreur. Par exemple il ne permet pas de

terminer une librairie tant que cette dernière n'a pas été initialisée. Il sera donc ainsi impossible de rencontrer le code retour CKR_CRYPTOKI_NOT_INITIALIZED.

Cependant si l'option *Déverrouiller le menu* est actionnée, la cohérence est « brisée », et tous les codes retour peuvent-être rencontrés.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.1.3 Indications techniques

L'écran de résultat fournit des informations techniques PKCS#11, pour repère, sur l'opération et le retour. L'opération est suffixée du nom de la fonction PKCS#11 testée, et le retour est suffixé du code retour en valeur hexadécimale (cf. copies d'écrans de résultats ci-après).

5.1.4 Librairie PKCS#11



Figure 5 : Chargement d'une librairie PKCS#11

Le programme offre la possibilité via son menu *Options* de charger une librairie PKCS#11. La librairie chargée par défaut est la librairie PKCS#11 de la carte CPS3.

Lorsque l'utilisateur invoque la fonction *Charger une librairie PKCS#11...* un explorateur de fichiers apparaît (cf. Figure ci-dessous), permettant de sélectionner la librairie à charger. A l'issue de ce choix le programme tente de charger la librairie. Voici les deux issues possibles à cette tentative :

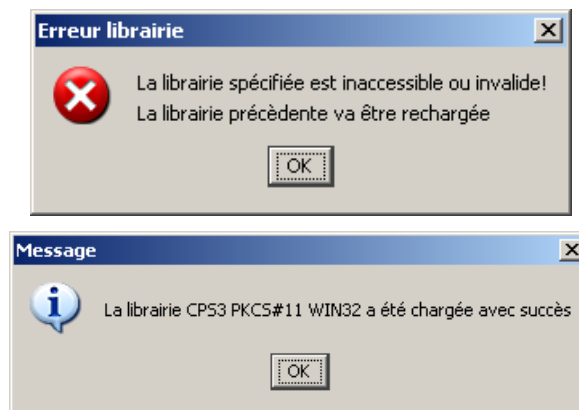


Figure 6 : Les deux issues possibles au chargement d'une librairie PKCS#11

Une barre en pied de page du programme permet d'identifier la librairie en cours d'utilisation. Le libellé de cette barre sera actualisé dès lors que l'utilisateur choisit une autre librairie valide.

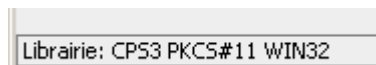


Figure 7 : Barre d'état mentionnant la librairie PKCS#11 en cours d'utilisation

5.2 Général

5.2.1 Initialiser la librairie

Le but de cette fonctionnalité est de tester la fonction C_Initialize de la librairie PKCS#11.

Action pré-requise :

- Aucune ou Terminer la librairie si elle est déjà initialisée

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

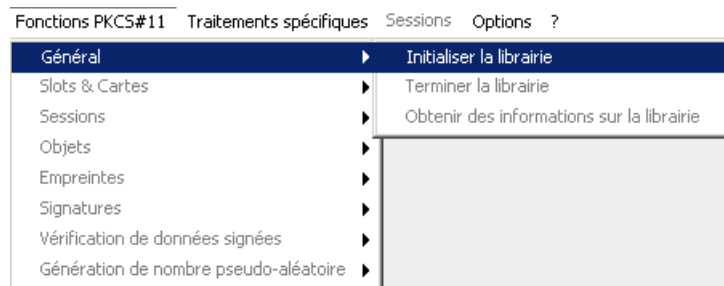


Figure 8 : Accès au menu Initialiser la librairie

Lorsque la fonction *Initialiser la librairie* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

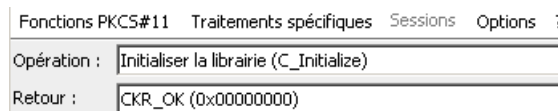


Figure 9 : Ecran de résultat d'opération: Initialiser la librairie

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CANT_LOCK, CKR_CRYPTOKI_ALREADY_INITIALIZED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_NEED_TO_CREATE_THREADS.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.2.2 Terminer la librairie

Le but de cette fonctionnalité est de tester la fonction C_Finalize de la librairie PKCS#11.

Cette fonction est à appeler en dernier. Plus aucune fonction, à l'exception de l'initialisation n'est actionnable après terminaison de la librairie.

Action pré-requise :

- Initialiser la librairie.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

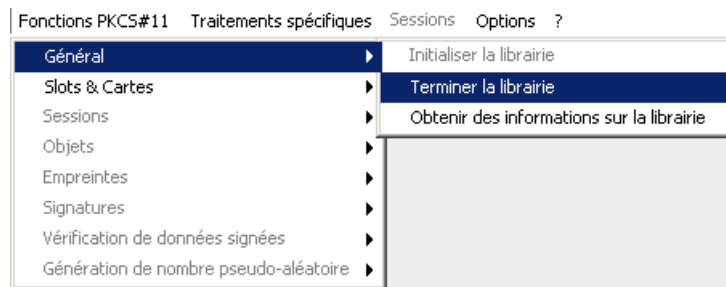


Figure 10 : Accès à la fonction de terminaison de la librairie

Lorsque la fonction *Terminer la librairie* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

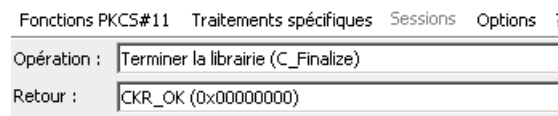


Figure 11 : Ecran de résultat d'opération : Terminer la librairie

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.2.3 Obtenir des informations sur la librairie

Le but de cette fonctionnalité est de tester la fonction C_GetInfo de la librairie PKCS#11.

Action pré-requise :
- *Initialiser la librairie.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

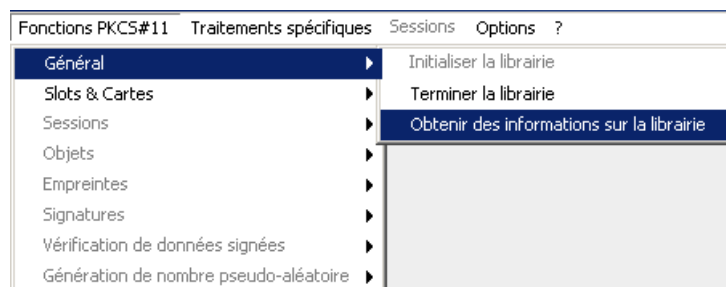


Figure 12 : Accès à la fonction d'obtention des informations sur la librairie

Lorsque la fonction *Obtenir des informations sur la librairie* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

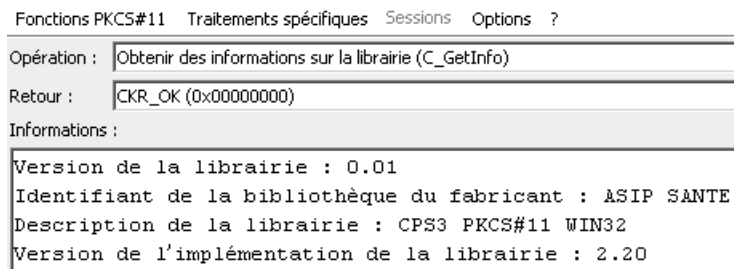


Figure 13 : Ecran de résultat d'opération : Obtenir des informations sur la librairie

Le code retour est accompagné de l'ensemble des informations suivantes :

- Version de la librairie
- Identifiant de la bibliothèque du fabricant
- Description de la librairie
- Version de l'implémentation de la librairie

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.3 Slots et cartes

5.3.1 Lister les slots

Le but de cette fonctionnalité est de tester la fonction `C_GetSlotList` qui permet d'obtenir la liste de tous les slots raccordés au système (*Tous*) ou la liste de tous les slots raccordés au système contenant une carte supportée (*Avec carte*) par la librairie utilisée.

Action pré-requise :
- *Initialiser la librairie.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

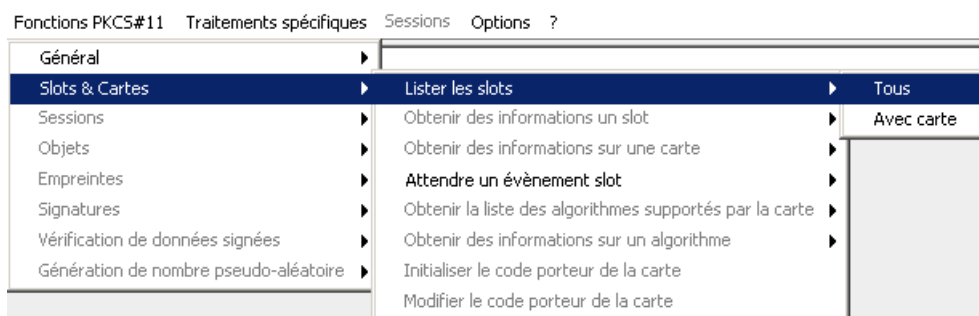


Figure 14 : Accès à la fonction d'énumération des slots

Lorsque la fonction *Lister les slots* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

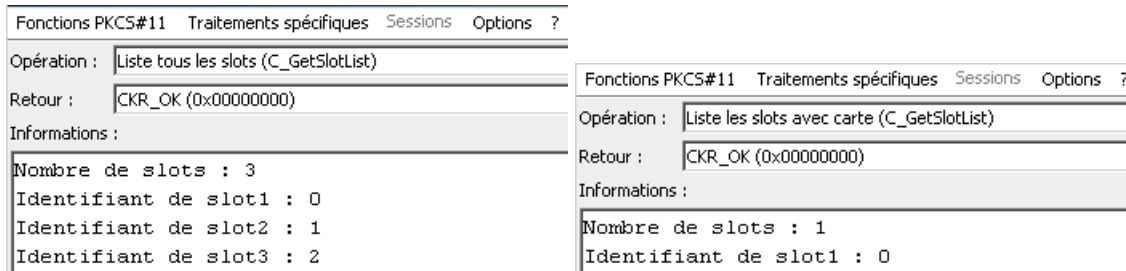


Figure 15 : Ecran de résultat d'opération : Liste tous les slots / Liste les slots avec carte

Le code retour est accompagné de l'ensemble des informations suivantes :

- Nombre de slots
- Liste des identifiants de slots

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

L'action de Lister tous les slots active un ensemble d'items de menus dont la fonction est dépendante de la présence de slots (ex : Obtenir des informations sur un slot).

L'action de Lister tous les slots avec carte active un ensemble d'items de menus dont la fonction est dépendante de la présence de cartes (ex : Obtenir des informations sur une carte).

5.3.2 Obtenir des informations sur un slot

Le but de cette fonctionnalité est de tester la fonction *C_GetSlotInfo* qui permet d'obtenir des informations sur un des slots raccordés au système.

Actions pré-requises :
 - *Initialiser la librairie.*
 - *Lister les slots et obtenir au moins un slot.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

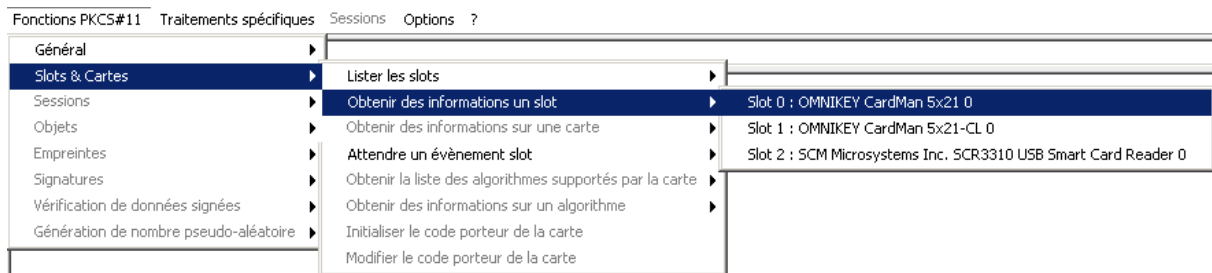


Figure 16 : Accès à la fonction d'obtention des informations sur un slot

Lorsque la fonction *Obtenir des informations sur un slot* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

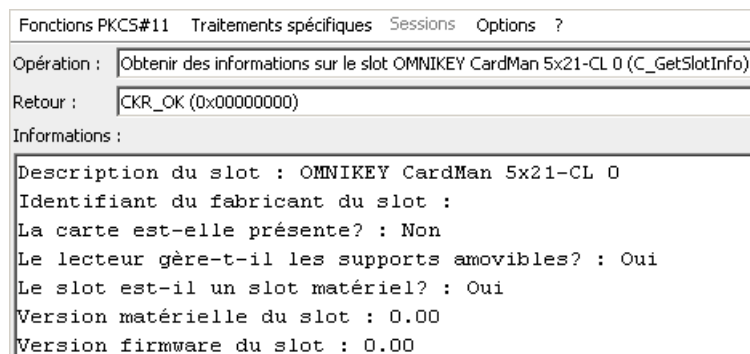


Figure 17 : Ecran de résultat d'opération : Obtenir des informations sur le slot

Le code retour est accompagné de l'ensemble des informations suivantes :

- Description du slot
- Identifiant du fabricant du slot
- Un ensemble de flags permettant de répondre aux questions suivantes :
 - La carte est-elle présente ? Oui/Non
 - Le lecteur gère-t-il les supports amovibles ? Oui/Non
 - Le slot est-il un slot matériel ? Oui/Non
- Version matérielle du slot
- Version firmware du slot

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SLOT_ID_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.3.3 Obtenir des informations sur une carte

Le but de cette fonctionnalité est de tester la fonction C_GetTokenInfo qui permet d'obtenir des informations sur une carte présente dans un slot.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

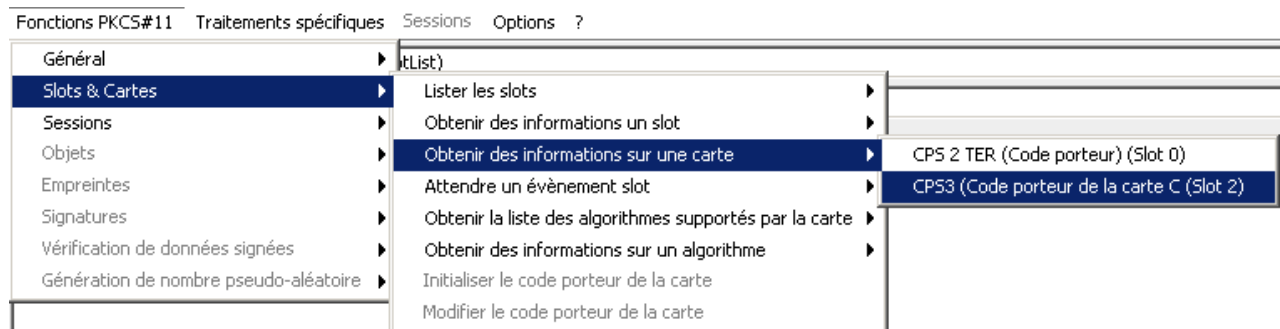


Figure 18 : Accès à la fonction d'obtention des informations sur une carte

Lorsque la fonction *Obtenir des informations sur une carte* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

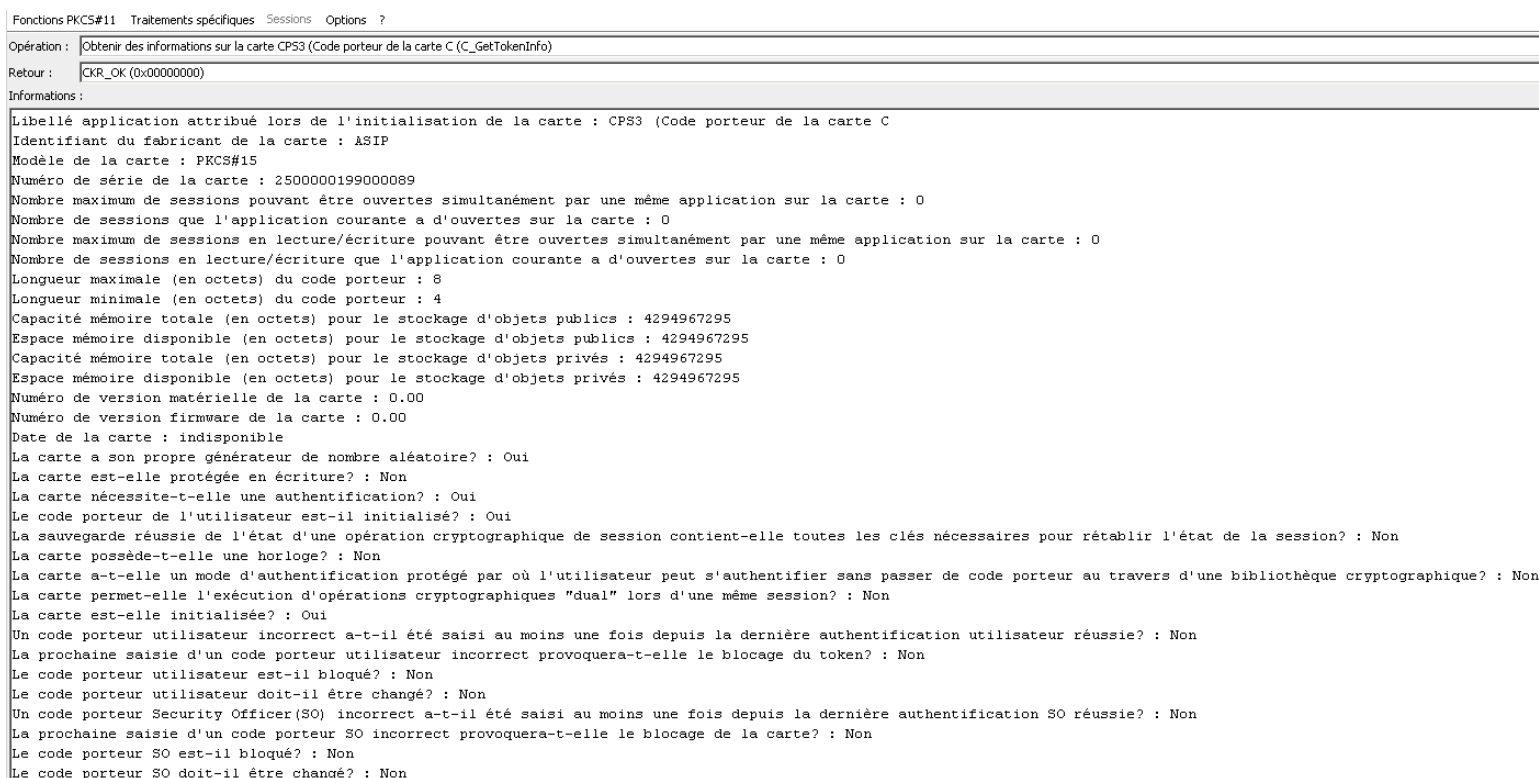


Figure 19 : Ecran de résultat d'opération : Obtenir des informations sur la carte

Le code retour est accompagné de l'ensemble des informations suivantes :

- Libellé application attribué lors de l'initialisation de la carte
- Identifiant du fabricant de la carte
- Modèle de la carte
- Numéro de série de la carte
- Un ensemble de flags permettant de répondre aux questions suivantes :
 - La carte a son propre générateur de nombre aléatoire ? Oui/Non
 - La carte est-elle protégée en écriture ? Oui/Non
 - La carte nécessite-t-elle une authentification ? Oui/Non
 - Le code porteur de l'utilisateur est-il initialisé ? Oui/Non
 - La sauvegarde réussie de l'état d'une opération cryptographique de session contient-elle toutes les clés nécessaires pour rétablir l'état de la session ? Oui/Non
 - La carte possède-t-elle une horloge ? Oui/Non
 - La carte a-t-elle un mode d'authentification protégé par l'intermédiaire duquel l'utilisateur peut s'authentifier sans passer de code porteur au travers d'une bibliothèque cryptographique ? Oui/Non
 - La carte permet-elle l'exécution d'opérations cryptographiques "dual" lors d'une même session ? Oui/Non
 - La carte est-elle initialisée ? Oui/Non
 - Un code porteur utilisateur incorrect a-t-il été saisi au moins une fois depuis la dernière authentification utilisateur réussie ? Oui/Non
 - La prochaine saisie d'un code porteur utilisateur incorrect provoquera-t-elle le blocage du token ? Oui/Non
 - Le code porteur utilisateur est-il bloqué ? Oui/Non
 - Le code porteur utilisateur doit-il être changé ? Oui/Non
 - Un code porteur Security Officer(SO) incorrect a-t-il été saisi au moins une fois depuis la dernière authentification SO réussie ? Oui/Non
 - La prochaine saisie d'un code porteur SO incorrect provoquera-t-elle le blocage du token ? Oui/Non
 - Le code porteur SO est-il bloqué ? Oui/Non
 - Le code porteur SO doit-il être changé ? Oui/Non
- Nombre maximum de sessions pouvant être ouvertes simultanément par une même application sur la carte
- Nombre de sessions que l'application courante a ouvertes sur la carte
- Nombre maximum de sessions en lecture/écriture pouvant être ouvertes simultanément par une même application sur la carte
- Nombre de sessions en lecture/écriture que l'application courante a ouvertes sur la carte
- Longueur maximale (en octets) du code porteur
- Longueur minimale (en octets) du code porteur
- Capacité mémoire totale (en octets) pour le stockage d'objets publics

- Espace mémoire disponible (en octets) pour le stockage d'objets publics
- Capacité mémoire totale (en octets) pour le stockage d'objets privés
- Espace mémoire disponible (en octets) pour le stockage d'objets privés
- Numéro de version matérielle de la carte
- Numéro de version firmware de la carte

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

`CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SLOT_ID_INVALID, CKR_TOKEN_NOT_PRESENT, CKR_TOKEN_NOT_RECOGNIZED, CKR_ARGUMENTS_BAD.`

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.3.4 Attendre un évènement slot

Le but de cette fonctionnalité est de tester la fonction `C_WaitForSlotEvent` qui se met en attente d'un évènement slot tel que l'insertion ou le retrait d'une carte. Cette fonction peut-être invoquée selon 2 modes :

- Mode bloqué
- Mode non bloquant

Dans le mode non bloquant, la fonction `C_WaitForSlotEvent` se base sur le flag d'évènement des slots (1 flag par slot) pour retourner directement un identifiant de slot. Dans le mode bloqué la fonction se base également sur ces flags, mais elle attend que l'un d'entre eux soit mis à jour.

Attention dans le *Mode bloqué* le programme se met en attente d'un évènement slot. L'utilisateur est alerté et les autres fonctionnalités du programme ne sont pas accessibles tant qu'un évènement slot n'est pas survenu.

Action pré-requise :
- *Initialiser la librairie.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

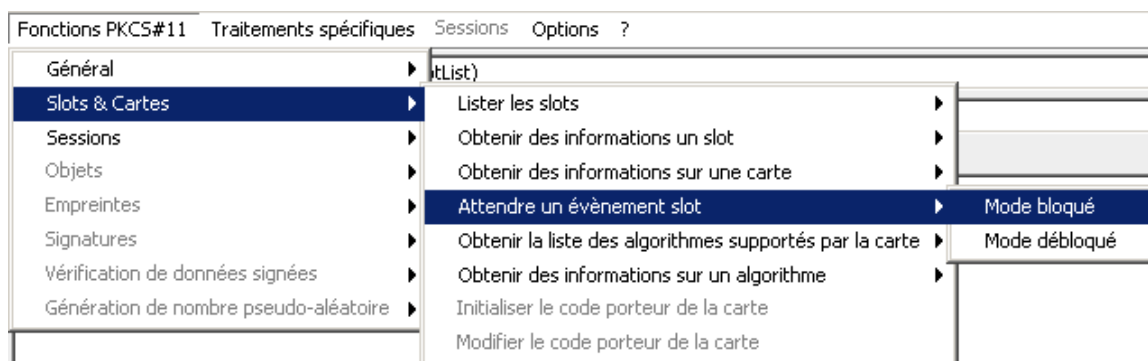


Figure 20 : Accès à la fonction d'attente d'un évènement slot

Lorsque la fonction *Attendre un évènement slot* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

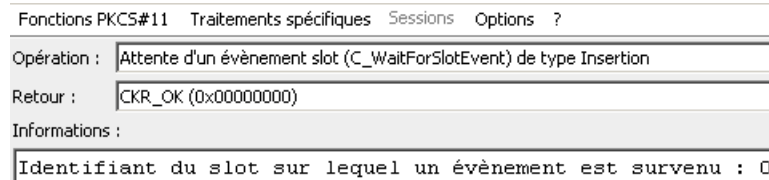


Figure 21 : Ecran de résultat d'opération : Attente d'un évènement slot

Le code retour est accompagné de l'ensemble des informations suivantes :

- Identifiant du slot sur lequel un évènement est survenu

Attention, en cas d'évènement sur plusieurs slot, un seul identifiant de slot est retourné.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_NO_EVENT.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.3.5 Obtenir la liste des algorithmes supportés par la carte

Le but de cette fonctionnalité est de tester la fonction *C_GetMechanismList* qui permet d'obtenir la liste des algorithmes supportés par la carte.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

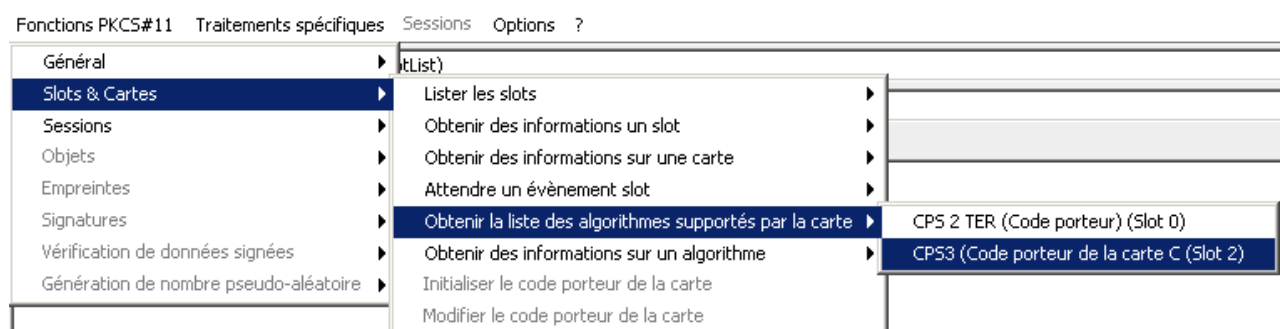


Figure 22 : Accès à la fonction d'Obtention des algorithmes supportés par la carte

Lorsque la fonction *Obtenir la liste des algorithmes supportés par la carte* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

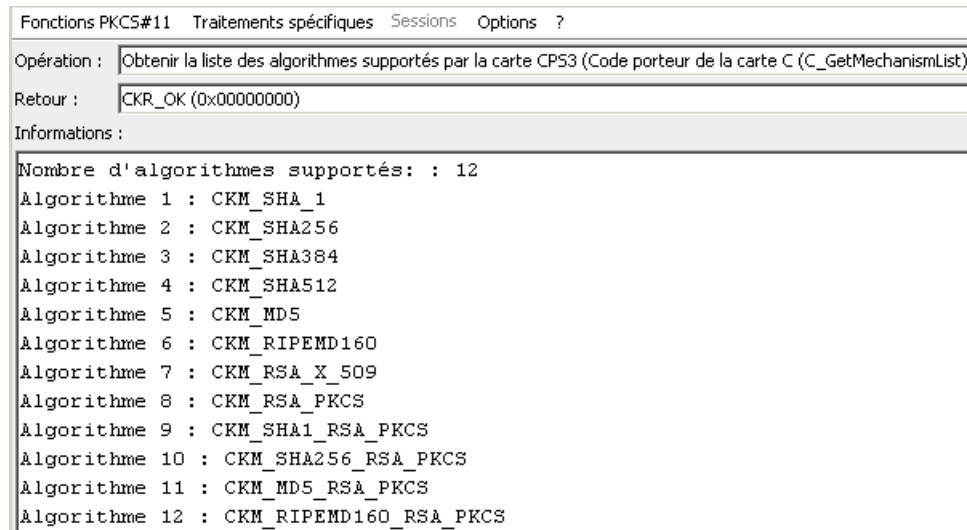


Figure 23 : Ecran de résultat d'opération : Obtenir la liste des algorithmes supportés par la carte

Le code retour est accompagné de l'ensemble des informations suivantes :

- Nombre d'algorithmes supportés
- Liste des types d'algorithmes supportés par la carte

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SLOT_ID_INVALID, CKR_TOKEN_NOT_PRESENT, CKR_TOKEN_NOT_RECOGNIZED, CKR_ARGUMENTS_BAD.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

L'action sur le menu Obtenir la liste des algorithmes supportés par la carte active un ensemble d'items correspondant aux algorithmes supportés par la carte. Des sous menus reprenant, pour toute ou partie, la liste de ces algorithmes, s'active également, et permettent d'exécuter de nouvelles opérations (ex : Empreinte, Signature).

5.3.6 Obtenir des informations sur un algorithme

Le but de cette fonctionnalité est de tester la fonction C_GetMechanismInfo qui permet d'obtenir des informations sur un algorithme particulier d'une carte.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Obtenir la liste des algorithmes supportés par la carte avec au moins un algorithme retourné.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

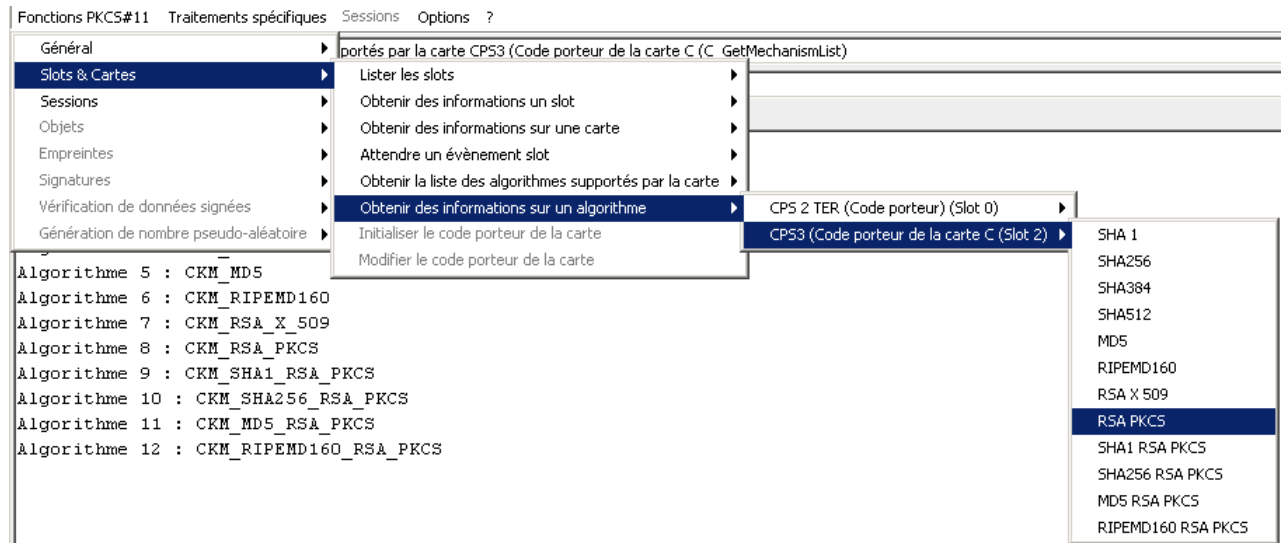


Figure 24 : Accès à la fonction d'obtention des informations sur un algorithme supporté par une carte

Lorsque la fonction *Obtenir des informations sur un algorithme* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

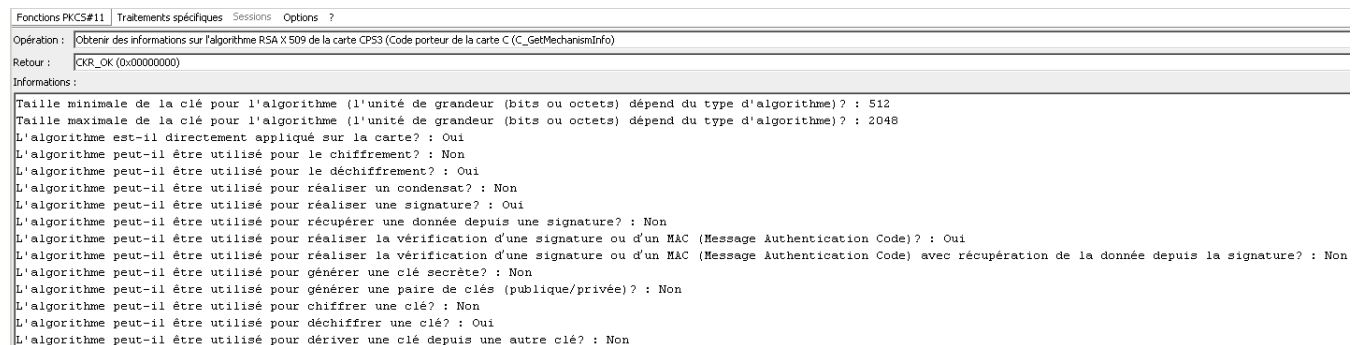


Figure 25 : Ecran de résultat d'opération : Obtenir des informations sur un algorithme

Le code retour est accompagné de l'ensemble des informations suivantes :

- Taille minimale de la clé pour l'algorithme (l'unité de grandeur (bits ou octets) dépend du type d'algorithme)
- Taille maximale de la clé pour l'algorithme (l'unité de grandeur (bits ou octets) dépend du type d'algorithme)
- Un ensemble de flags permettant de répondre aux questions suivantes :
 - L'algorithme est-il directement exécuté par la carte ? Oui/Non
 - L'algorithme peut-il être utilisé pour le chiffrement ? Oui/Non
 - L'algorithme peut-il être utilisé pour le déchiffrement ? Oui/Non
 - L'algorithme peut-il être utilisé pour réaliser un condensat ? Oui/Non
 - L'algorithme peut-il être utilisé pour réaliser une signature ? Oui/Non

- L'algorithme peut-il être utilisé pour récupérer une donnée depuis une signature ? Oui/Non
- L'algorithme peut-il être utilisé pour réaliser la vérification d'une signature ou d'un MAC (Message Authentication Code) ? Oui/Non
- L'algorithme peut-il être utilisé pour réaliser la vérification d'une signature ou d'un MAC (Message Authentication Code) avec récupération de la donnée depuis la signature ? Oui/Non
- L'algorithme peut-il être utilisé pour générer une clé secrète ? Oui/Non
- L'algorithme peut-il être utilisé pour générer une paire de clés (publique/privée) ? Oui/Non
- L'algorithme peut-il être utilisé pour chiffrer une clé ? Oui/Non
- L'algorithme peut-il être utilisé pour déchiffrer une clé ? Oui/Non
- L'algorithme peut-il être utilisé pour dériver une clé depuis une autre clé ? Oui/Non

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

`CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_MECHANISM_INVALID, CKR_SLOT_ID_INVALID, CKR_TOKEN_NOT_PRESENT, CKR_TOKEN_NOT_RECOGNIZED, CKR_ARGUMENTS_BAD.`

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.3.7 Initialiser le code porteur de la carte

Le but de cette fonctionnalité est de tester la fonction `C_InitPIN` qui permet d'initialiser le code porteur d'une carte. Dans la pratique, cette fonction permet de débloquent le code porteur de la carte en s'authentifiant préalablement avec le code PUK (SO).

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session en Lecture/Ecriture sur la carte associée à la session courante.**
- **S'authentifier sur la carte en tant que Security Officer(SO) sur la carte associée à la session courante.**

La carte concernée est la carte associée à la session courante.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

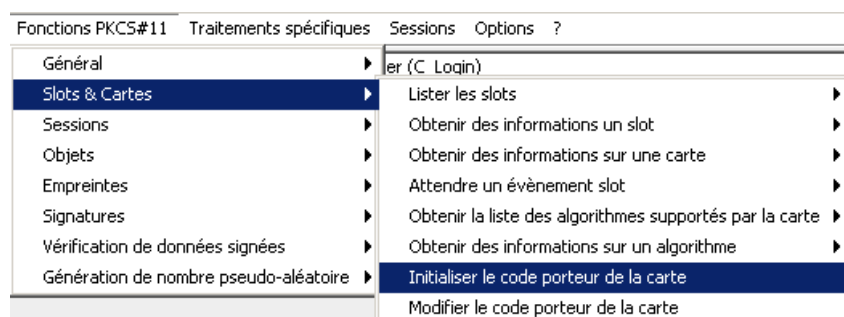
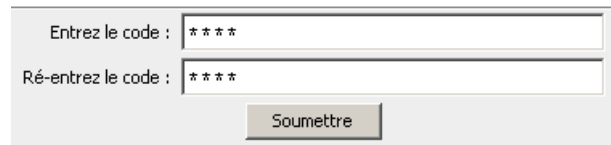


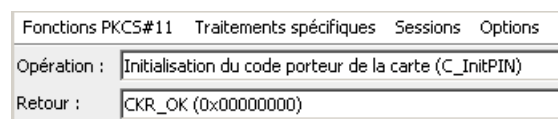
Figure 26 : Accès à la fonction d'initialisation du code porteur de la carte

L'utilisateur doit alors fournir l'information suivante :

- Code porteur à attribuer

**Figure 27 : Ecran de saisie : Initialisation du code porteur de la carte**

Lorsque la fonction *Initialiser le code porteur de la carte* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

**Figure 28 : Ecran de résultat d'opération : Initialisation du code porteur de la carte**

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_PIN_INVALID, CKR_PIN_LEN_RANGE, CKR_SESSION_CLOSED, CKR_SESSION_READ_ONLY, CKR_SESSION_HANDLE_INVALID, CKR_TOKEN_WRITE_PROTECTED, CKR_USER_NOT_LOGGED_IN, CKR_ARGUMENTS_BAD.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.3.8 Modifier le code porteur de la carte

Le but de cette fonctionnalité est de tester la fonction `C_SetPIN` qui permet de modifier le code porteur d'une carte.

Actions pré-requises :

- *Initialiser la librairie.*
- *Lister les slots avec carte et obtenir au moins un slot.*
- *Ouvrir une session en Lecture/Ecriture sur la carte associée à la session courante.*

La carte concernée est la carte associée à la session courante.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

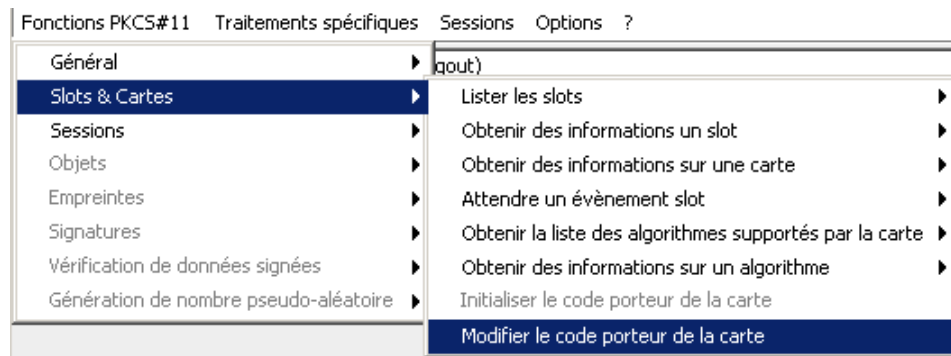


Figure 29 : Accès à la fonction de modification du code porteur de la carte

L'utilisateur doit alors fournir les informations suivantes :

- Ancien code porteur
- Nouveau code porteur à attribuer (à 2 reprises)

 The image shows a form for modifying the PIN. It has three input fields, each preceded by a label: 'Entrez le code PIN actuel :', 'Entrez le nouveau code PIN', and 'Entrez le nouveau code PIN à nouveau'. Each field contains four asterisks (****). Below the fields is a button labeled 'Soumettre'.

Figure 30 : Ecran de saisie : Modification du code porteur de la carte

Lorsque la fonction *Modifier le code porteur de la carte* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

 The image shows a result screen with a top bar containing 'Fonctions PKCS#11', 'Traitements spécifiques', 'Sessions', 'Options', and '?'. Below the bar, there are two rows. The first row is labeled 'Opération :' and contains the text 'Modification du code porteur de la carte (C_SetPIN)'. The second row is labeled 'Retour :' and contains the text 'CKR_OK (0x00000000)'.

Figure 31 : Ecran de résultat d'opération : Modification du code porteur de la carte

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_PIN_INCORRECT, CKR_PIN_INVALID, CKR_PIN_LEN_RANGE, CKR_PIN_LOCKED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SESSION_READ_ONLY, CKR_TOKEN_WRITE_PROTECTED, CKR_ARGUMENTS_BAD.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.4 Sessions

5.4.1 Ouvrir une session

Le but de cette fonctionnalité est de tester la fonction C_OpenSession qui permet d'établir une session entre l'application et la carte.

Deux types de sessions sont possibles. Les sessions en lecture seule, et les sessions en lecture/écriture.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**

5.4.1.1 Accès Lecture seule

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

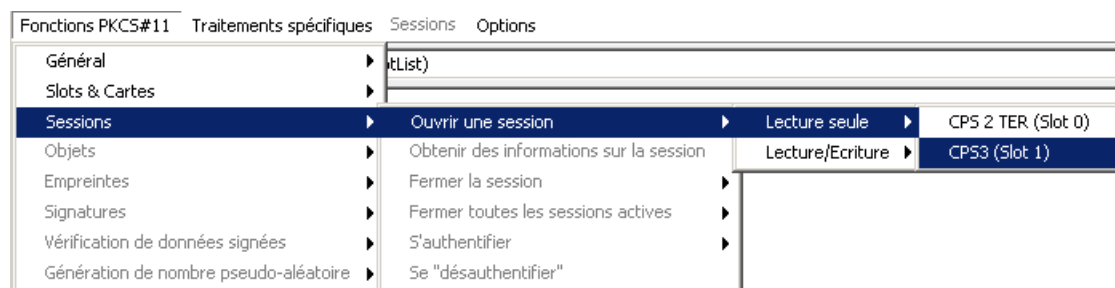


Figure 32 : Accès à la fonction d'ouverture de session Lecture seule

5.4.1.2 Accès Lecture/Ecriture

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

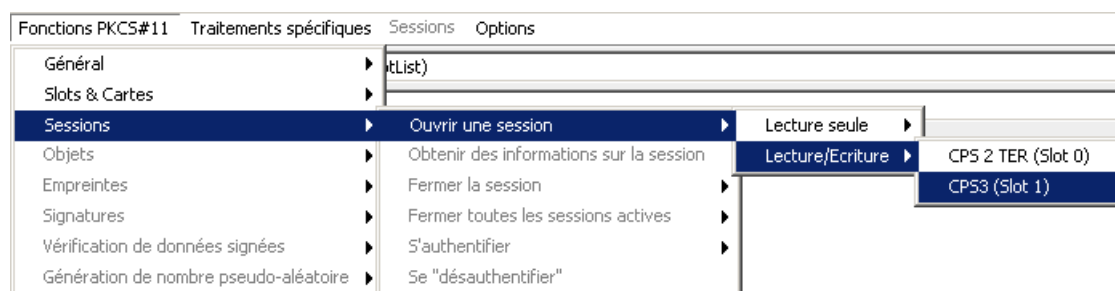


Figure 33 : Accès à la fonction d'ouverture de session en Lecture/Ecriture

5.4.1.3 Résultats

Lorsque la fonction *Ouvrir une session* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Opération :	Ouverture d'une session en lecture seule (C_OpenSession)			
Retour :	CKR_OK (0x00000000)			
Informations :				
Identifiant de la session ouverte : 1				

Figure 34 : Ecran de résultat d'opération : Ouvrir une session en lecture seule

Le code retour est accompagné de l'information suivante :

- Identifiant de la session ouverte

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

`CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SESSION_COUNT, CKR_SESSION_PARALLEL_NOT_SUPPORTED, CKR_SESSION_READ_WRITE_SO_EXISTS, CKR_SLOT_ID_INVALID, CKR_TOKEN_NOT_PRESENT, CKR_TOKEN_NOT_RECOGNIZED, CKR_TOKEN_WRITE_PROTECTED, CKR_ARGUMENTS_BAD.`

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.4.1.4 Passage d'une session à l'autre

Le programme d'exemple permet de passer d'une session à l'autre très aisément grâce au menu **Sessions** qui liste l'ensemble des sessions actives. La session active est ci-après représentée en gras. L'utilisateur peut choisir dans le menu la session avec laquelle il désire travailler.

YYYYMMDD hh :ii :ss représente le Timestamp de l'ouverture de la session (ex : 20100407 14:25:12 pour le 7 avril 2010 à 14:25:12). Cela peut aider l'utilisateur à différencier des sessions actives présentant les mêmes attributs : mode d'accès, slot et utilisateur.

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Opération :	Ouverture d'une session en lecture	Session 5 (RO public/Slot 0/20100706 10:16:27)		
Retour :	CKR_OK (0x00000000)	Session 6 (RW public/Slot 2/20100706 10:16:31)		

Figure 35 : Accès à la fonctionnalité de bascule de sessions

5.4.2 Obtenir des informations sur la session

Le but de cette fonctionnalité est de tester la fonction `C_GetSessionInfo` qui permet d'obtenir des informations sur la session.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session.**

La session considérée est la session courante.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options
Général			seule (C_OpenSession)
Slots & Cartes			
Sessions		Ouvrir une session	
Objets		Obtenir des informations sur la session	
Empreintes		Fermer la session	
Signatures		Fermer toutes les sessions actives	
Vérification de données signées		S'authentifier	
Génération de nombre pseudo-aléatoire		Se "désauthentifier"	

Figure 36 : Accès à la fonction d'obtention d'informations sur la session courante

Lorsque la fonction *Obtenir des informations sur la session* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

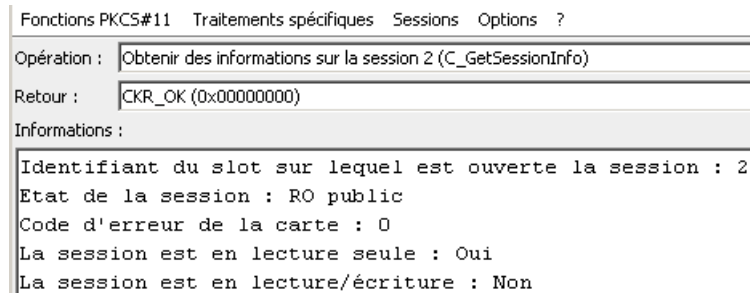


Figure 37 : Ecran de résultat d'opération : Obtenir des informations sur la session

Le code retour est accompagné des informations suivantes :

- Identifiant du slot sur lequel est ouverte la session
- Etat de la session
- Code d'erreur de la carte
- Un flag permettant de déterminer si la session est en lecture seule.
- Un flag permettant de déterminer si la session est en lecture/écriture.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_ARGUMENTS_BAD.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.4.3 Fermer la session

Le but de cette fonctionnalité est de tester la fonction *C_CloseSession* qui permet de fermer une session.

Actions pré-requises :

- *Initialiser la librairie.*
- *Lister les slots avec carte et obtenir au moins un slot.*
- *Ouvrir une session.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

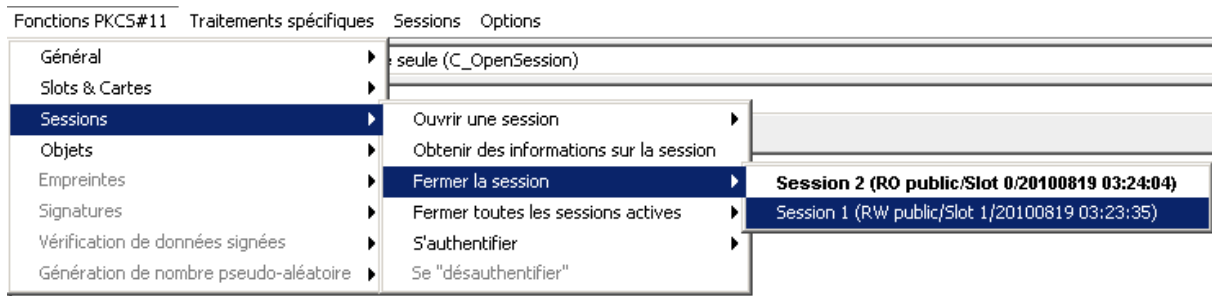


Figure 38 : Accès à la fonction de fermeture d'une session active

Lorsque la fonction *Fermer la session* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

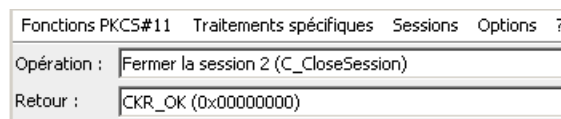


Figure 39 : Ecran de résultat d'opération : Fermer la session

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.4.4 Fermer toutes les sessions actives

Le but de cette fonctionnalité est de tester la fonction C_CloseAllSessions qui permet de fermer toutes les sessions ouvertes sur une carte.

Actions pré-requises :

- *Initialiser la librairie.*
- *Lister les slots avec carte et obtenir au moins un slot.*
- *Ouvrir une voire plusieurs sessions.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

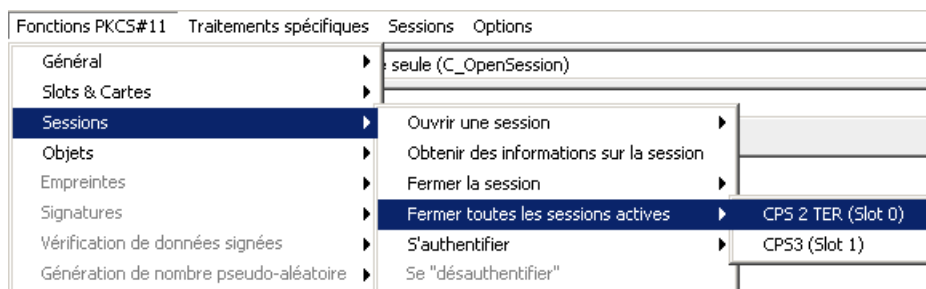


Figure 40 : Accès à la fonction de fermeture de toutes les sessions actives sur une carte

Lorsque la fonction *Fermer toutes les sessions actives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Opération :	Fermer toutes les sessions actives de la carte CPS3 (Code porteur de la carte C (C_CloseAllSessions))			
Retour :	CKR_OK (0x00000000)			

Figure 41 : Ecran de résultat d'opération : Fermer toutes les sessions actives

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SLOT_ID_INVALID, CKR_TOKEN_NOT_PRESENT.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.4.5 S'authentifier

Le but de cette fonctionnalité est de tester la fonction C_Login qui permet d'authentifier un utilisateur sur une carte.

Il y a deux types d'utilisateurs :

- Security Officer (SO)
- Utilisateur final (porteur de la carte)

Cette fonctionnalité passe la session courante à l'état authentifié.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session.**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

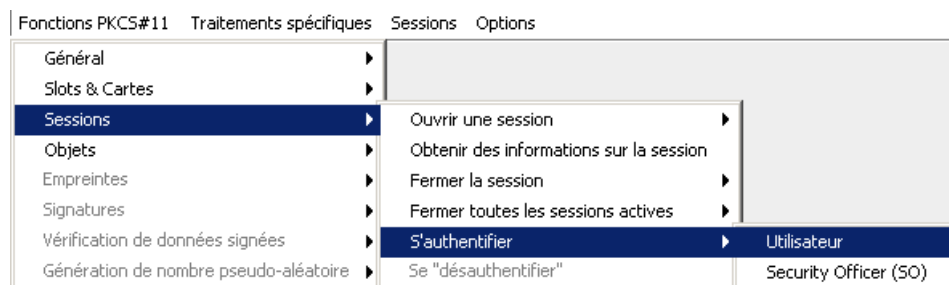


Figure 42 : Accès à la fonction d'authentification

L'utilisateur doit alors fournir l'information suivante :

- Code porteur associé au type d'utilisateur choisit.

Entrez le code PUK :	*****
<input type="button" value="Soumettre"/>	

Figure 43 : Ecran de saisie : Authentification Security Officer (SO)

Lorsque la fonction *S'authentifier* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Opération : Authentification d'un utilisateur (C_Login)				
Retour : CKR_OK (0x00000000)				

Figure 44 : Ecran de résultat d'opération : S'authentifier

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_PIN_INCORRECT, CKR_PIN_LOCKED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SESSION_READ_ONLY_EXISTS, CKR_USER_ALREADY_LOGGED_IN, CKR_USER_ANOTHER_ALREADY_LOGGED_IN, CKR_USER_PIN_NOT_INITIALIZED, CKR_USER_TOO_MANY_TYPES, CKR_USER_TYPE_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

Lorsqu'un utilisateur est authentifié le programme met à jour le libellé de la session correspondante dans les menus sessions (cf. RO utilisateur (au lieu de RO public) sur la Figure ci-dessous).

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options
Opération : Authentification d'un utilisateur (C_Login)		Session 2 (RO utilisateur/Slot 0/20100819 03:24:04)	

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options
Général	▶	Login)	
Slots & Cartes	▶		
Sessions	▶	Ouvrir une session	
Objets	▶	Obtenir des informations sur la session	
Empreintes	▶	Fermer la session	Session 2 (RO utilisateur/Slot 0/20100819 03:24:04)
Signatures	▶	Fermer toutes les sessions actives	
Vérification de données signées	▶	S'authentifier	
Génération de nombre pseudo-aléatoire	▶	Se "désauthentifier"	

Figure 45 : Mise à jour du libellé de la session après authentification

5.4.6 Se "désauthentifier"

Le but de cette fonctionnalité est de tester la fonction C_Logout qui permet de mettre fin à l'authentification du porteur vis-à-vis de la carte.

Cette fonctionnalité passe la session courante à l'état anonyme.

Actions pré-requises :

- *Initialiser la librairie.*
- *Lister les slots avec carte et obtenir au moins un slot.*
- *Ouvrir une session.*
- *S'authentifier.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

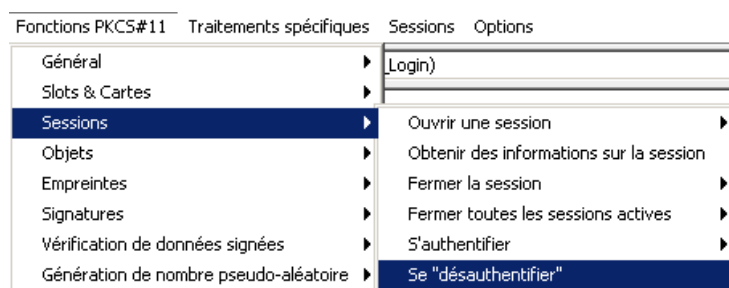


Figure 46 : Accès à la fonction de « désauthentification »

Lorsque la fonction Se « désauthentifier » s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options
Opération :	"Désauthentification" d'un utilisateur (C_Logout)		
Retour :	CKR_OK (0x00000000)		

Figure 47 : Ecran de résultat d'opération : « Désauthentification » d'un utilisateur

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.5 Objets

5.5.1 Gestion d'objets

Ce menu ne permet de travailler que sur des objets de données applicatives, qui sont les seuls types d'objets token « modifiables » supportés par la carte CPS3.

5.5.1.1 Créer l'objet de données applicatives

Le but de cette fonctionnalité est de tester la fonction C_CreateObject qui permet de créer un objet.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session en Lecture/Ecriture.**
- **S'authentifier.**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

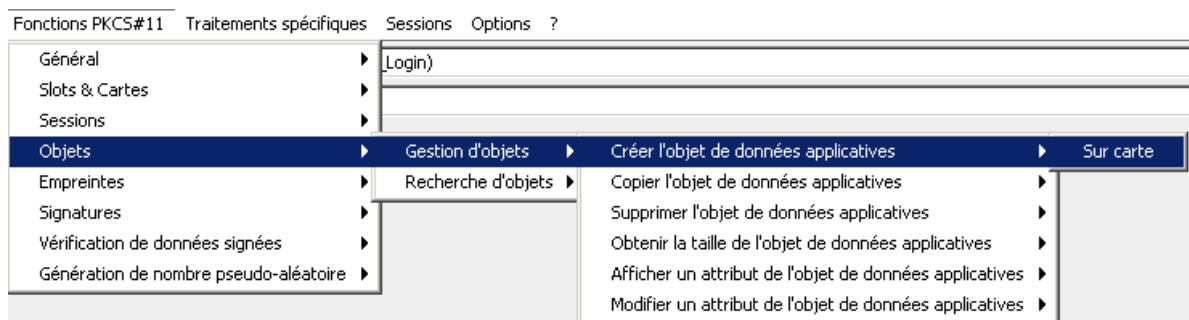


Figure 48 : Accès à la fonction de création d'objet

L'utilisateur se voit demander de saisir les informations suivantes :

 The screenshot shows a data entry form. At the top, there are tabs: 'Fonctions PKCS#11', 'Traitements spécifiques', 'Sessions', 'Options', and '?'. The form has two main input areas. The first is labeled 'Entrez un libellé pour l'objet :' and contains a text field with the value 'Mon Objet'. The second is labeled 'Entrez l'objet sous forme hexadécimale :' and contains a large text area filled with a grid of hexadecimal characters, primarily '00' and 'aa'. At the bottom right of the form is a button labeled 'Soumettre'.

Figure 49 : Ecran de saisie : Saisie d'un objet de données applicatives pour création

Lorsque la fonction *Créer l'objet de données applicatives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Opération :	Création de l'objet "Mon Objet" (C_CreateObject)			
Retour :	CKR_OK (0x00000000)			
Informations :				
Identifiant de l'objet :	3			

Figure 50 : Ecran de résultat d'opération : Création d'un objet de données applicatives

Le code retour est accompagné de l'ensemble des informations suivantes :

- Identifiant de l'objet

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_ATTRIBUTE_READ_ONLY, CKR_ATTRIBUTE_TYPE_INVALID, CKR_ATTRIBUTE_VALUE_INVALID, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_DOMAIN_PARAMS_INVALID, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_PIN_EXPIRED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SESSION_READ_ONLY, CKR_TEMPLATE_INCOMPLETE, CKR_TEMPLATE_INCONSISTENT, CKR_TOKEN_WRITE_PROTECTED, CKR_USER_NOT_LOGGED_IN.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

L'action de Création d'un objet active un ensemble d'items de menus représentant l'objet nouvellement créé et dont la fonction est dépendante de la présence d'un objet (ex : Copier l'objet de données applicatives...).

5.5.1.2 Copier l'objet de données applicatives

Le but de cette fonctionnalité est de tester la fonction C_CopyObject qui permet créer une copie d'un objet.

Attention, il est possible que la version de la carte utilisée ne supporte pas cette fonctionnalité. C'est le cas des cartes CPS3 dont la version est inférieure à la version 3. Un code retour signalant une erreur sera retourné dans ce cas.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Ouvrir une session en Lecture/Ecriture.
- S'authentifier.
- Recherche d'objets de données applicatives OU Copier l'objet de données applicatives OU Création d'un objet de données applicatives.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

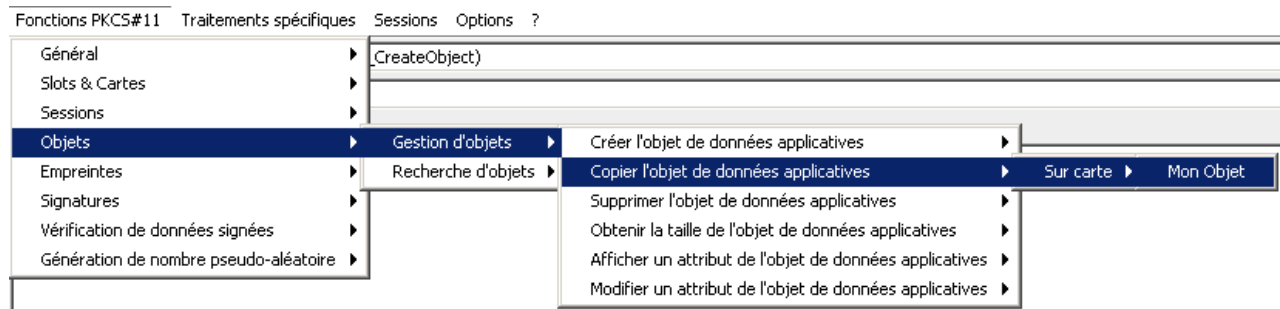


Figure 51 : Accès à la fonction de copie d'un objet de données applicatives

Lorsque la fonction *Copier l'objet de données applicatives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

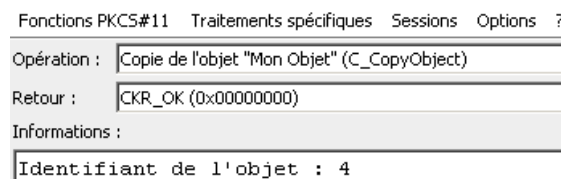


Figure 52 : Ecran de résultat d'opération : Copie d'un objet de données applicatives

Le code retour est accompagné de l'information suivante :

- Identifiant de l'objet

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_ATTRIBUTE_READ_ONLY, CKR_ATTRIBUTE_TYPE_INVALID, CKR_ATTRIBUTE_VALUE_INVALID, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OBJECT_HANDLE_INVALID, CKR_PIN_EXPIRED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SESSION_READ_ONLY, CKR_TEMPLATE_INCONSISTENT, CKR_TOKEN_WRITE_PROTECTED, CKR_USER_NOT_LOGGED_IN.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

L'action de Copie d'un objet active un ensemble d'items de menus représentant la copie de l'objet et dont la fonction est dépendante de la présence d'un objet (ex : Supprimer/Copier l'objet de données applicatives...).

Une fois l'objet copié l'item correspondant à la copie de l'objet apparaît dans les sous-menus appropriés (cf. Figure ci-dessous (« Copie de Mon Objet »)).

5.5.1.3 Supprimer l'objet de données applicatives

Le but de cette fonctionnalité est de tester la fonction *C_DestroyObject* qui permet de supprimer un objet.

Actions pré-requises :

- *Initialiser la librairie.*
- *Lister les slots avec carte et obtenir au moins un slot.*
- *Ouvrir une session en Lecture/Ecriture.*

- **S'authentifier.**
- **Recherche d'objets de données applicatives OU Copier l'objet de données applicatives OU Création d'un objet de données applicatives.**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

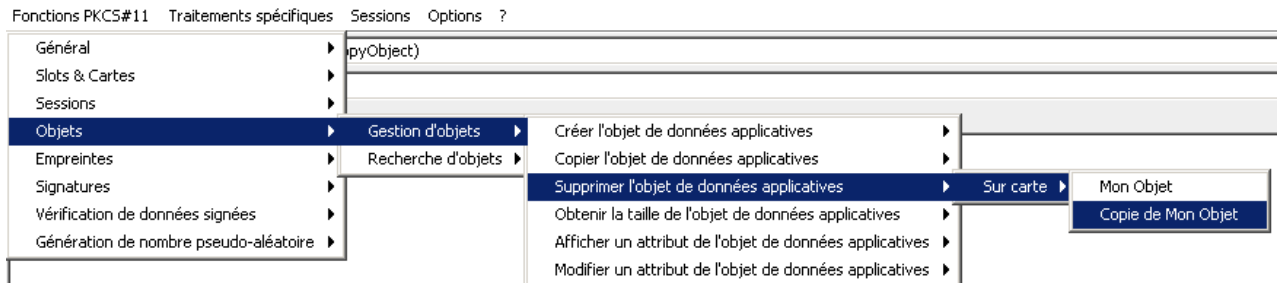


Figure 53 : Accès à la fonction de suppression d'un objet de données applicatives

Lorsque la fonction *Supprimer un objet de données applicatives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

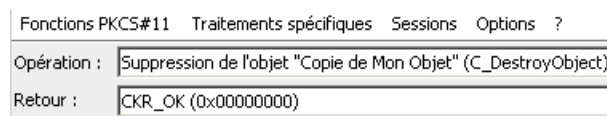


Figure 54 : Ecran de résultat d'opération : Suppression d'un objet de données applicatives

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OBJECT_HANDLE_INVALID, CKR_PIN_EXPIRED,, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SESSION_READ_ONLY, CKR_TOKEN_WRITE_PROTECTED.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

L'action de Copie d'un objet supprime un ensemble d'items de menus représentant la l'objet supprimé et dont la fonction est dépendante de la présence d'un objet (ex : Supprimer/Copier l'objet de données applicatives...).

5.5.1.4 Obtenir la taille de l'objet de données applicatives

Le but de cette fonctionnalité est de tester la fonction `C_GetObjectSize` qui permet d'obtenir la taille d'un objet.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session en lecture seule ou en Lecture/Ecriture, pour voir potentiellement plus d'objets.**
- **Optionnel : S'authentifier, pour obtenir potentiellement plus d'objets.**

**- Recherche d'objets de données applicatives OU Copier l'objet de données applicatives
OU Création d'un objet de données applicatives.**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

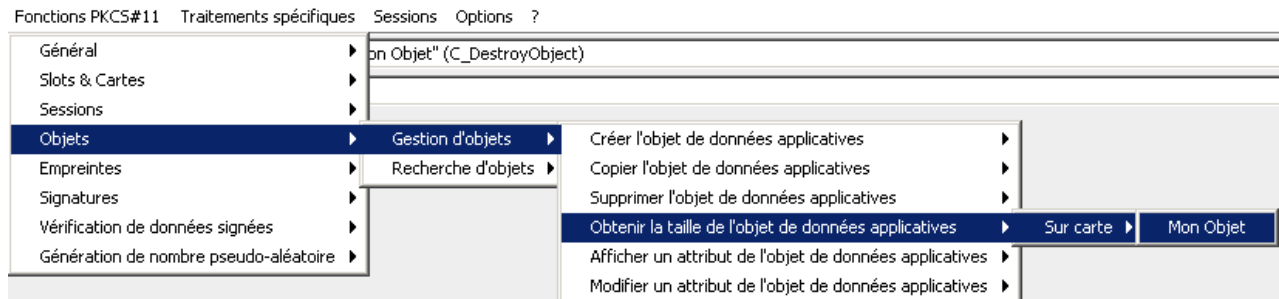


Figure 55 : Accès à la fonction d'obtention de la taille d'un objet de données applicatives

Lorsque la fonction *Obtenir la taille d'un objet de données applicatives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

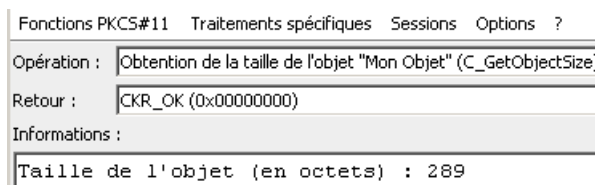


Figure 56 : Ecran de résultat d'opération : Obtention de la taille d'un objet de données applicatives

Le code retour est accompagné de l'information suivante :

- Taille de l'objet (en octets)

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_INFORMATION_SENSITIVE, CKR_OBJECT_HANDLE_INVALID, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.5.1.5 Afficher un attribut de l'objet de données applicatives

Le but de cette fonctionnalité est de tester la fonction C_GetAttributeValue qui permet d'obtenir la valeur d'un attribut d'un objet.

L'attribut considéré est CKA_VALUE.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Ouvrir une session en lecture seule ou en Lecture/Ecriture, pour voir potentiellement plus d'objets.

- *Optionnel : S'authentifier, pour obtenir potentiellement plus d'objets.*
- *Recherche d'objets de données applicatives OU Copier l'objet de données applicatives OU Création d'un objet de données applicatives.*

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

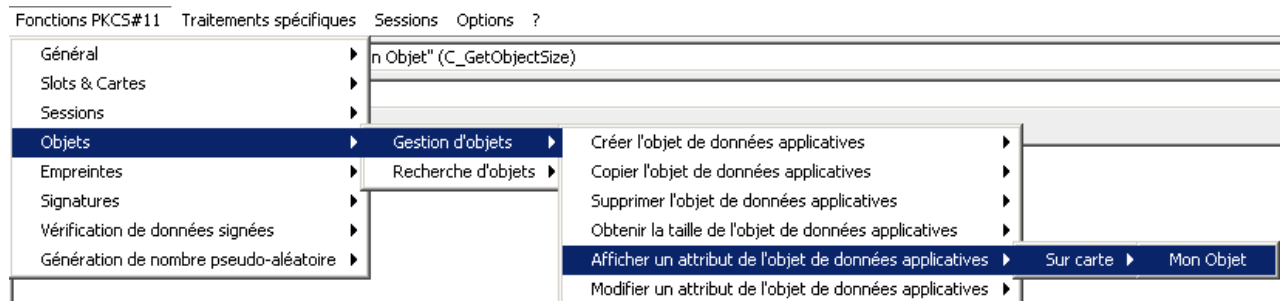


Figure 57 : Accès à la fonction d'affichage de l'attribut d'un objet de données applicatives

Lorsque la fonction *Afficher un attribut de l'objet de données applicatives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

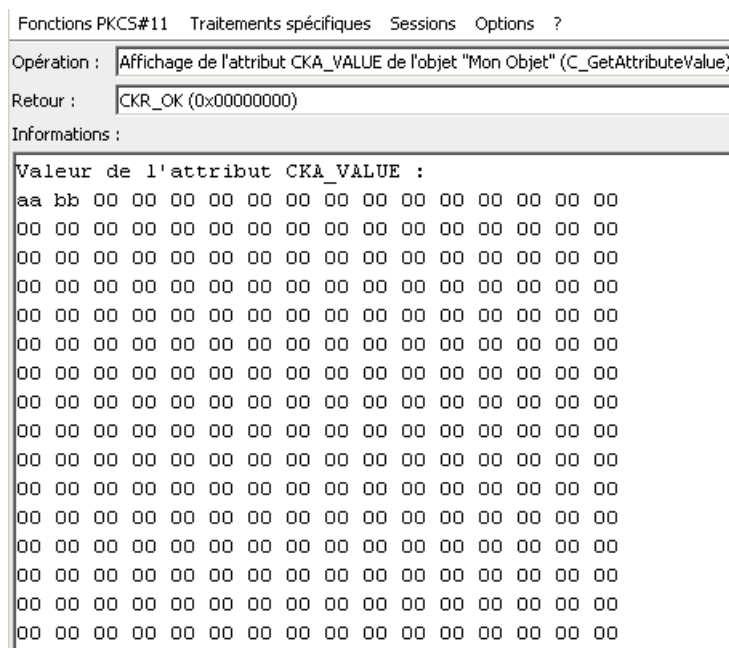


Figure 58 : Ecran de résultat d'opération : Affichage de l'attribut d'un objet de données applicatives

Le code retour est accompagné de l'information suivante :

- Valeur de l'attribut de l'objet

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_ATTRIBUTE_SENSITIVE, CKR_ATTRIBUTE_TYPE_INVALID,
CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OBJECT_HANDLE_INVALID,
CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.5.1.6 Modifier un attribut de l'objet de données applicatives

Le but de cette fonctionnalité est de tester la fonction `C_SetAttributeValue` qui permet modifier la valeur d'un attribut d'un objet.

L'attribut considéré est CKA_VALUE.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Ouvrir une session en Lecture/Ecriture
- S'authentifier
- Recherche d'objets de données applicatives OU Copier l'objet de données applicatives OU Création d'un objet de données applicatives.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

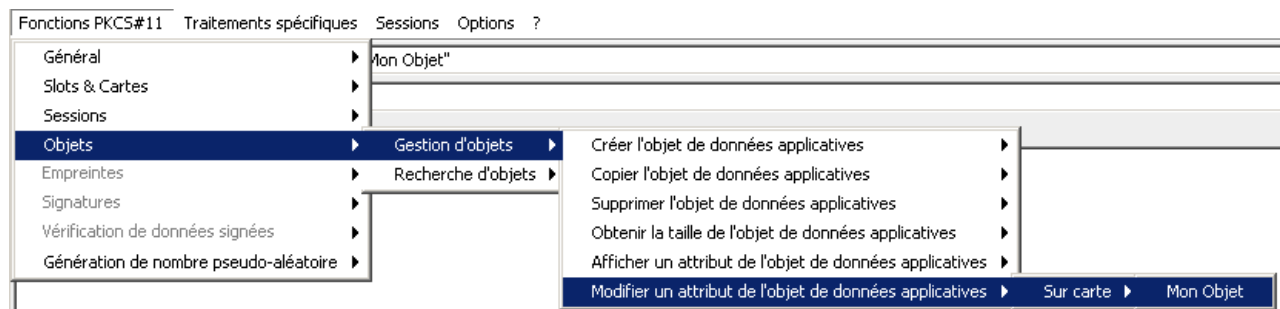


Figure 59 : Accès à la fonction de modification de l'attribut d'un objet de données applicatives

L'utilisateur se voit demander de saisir l'information suivante :

- Valeur de l'attribut de l'objet (sous forme hexadécimale).

Fonctions PKCS#11 Traitements spécifiques Sessions Options ?	
Libellé de l'objet :	Mon Objet
Modifier la valeur de l'objet (sous forme hexadécimale) :	aa bb cc dd 00
	<input type="button" value="Soumettre"/>

Figure 60 : Ecran de saisie : Saisie de la modification d'un attribut d'un objet de données applicatives

Lorsque la fonction *Modifier un attribut de l'objet de données applicatives* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Opération :	Modification de l'attribut CKA_VALUE de l'objet "Mon Objet" (C_SetAttributeValue)			
Retour :	CKR_OK (0x00000000)			

Figure 61 : Ecran de résultat d'opération : Modification de l'attribut d'un objet de données applicatives

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_ATTRIBUTE_READ_ONLY, CKR_ATTRIBUTE_TYPE_INVALID, CKR_ATTRIBUTE_VALUE_INVALID, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OBJECT_HANDLE_INVALID, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SESSION_READ_ONLY, CKR_TEMPLATE_INCONSISTENT, CKR_TOKEN_WRITE_PROTECTED, CKR_USER_NOT_LOGGED_IN.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.5.2 Recherche d'objets

Le but de cette fonctionnalité est de tester les fonctions C_FindObjectsInit, C_FindObjects et C_FindObjectsFinal qui permettent de rechercher des objets.

Les différents types d'objets que le programme d'exemple propose de rechercher sont :

- Certificats
- Clés publiques
- Clés privées (fonction accessible que pour un utilisateur authentifié)
- Objets de données applicatives

Pour la recherche, le programme d'exemple enchaîne l'appel aux 3 fonctions suivantes :

- Initialisation de la recherche (C_FindObjectsInit)
- Recherche (C_FindObjects)
- Clôture de la recherche (C_FindObjectsFinal)

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Ouvrir une session en lecture seule ou en Lecture/Ecriture, pour voir potentiellement plus d'objets.
- Optionnel : S'authentifier, pour obtenir potentiellement plus d'objets.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

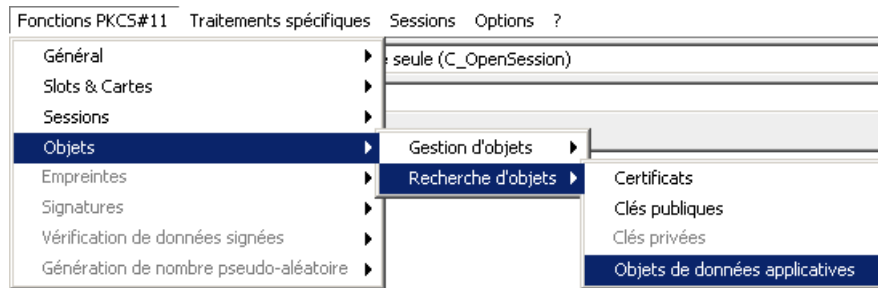


Figure 62 : Accès à la fonction de recherche d'objets

Lorsque la fonction *Recherche d'objets* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Operation 1 :	Recherche d'objet(s) de type certificat (Initialisation (C_FindObjectsInit))			
Retour 1 :	CKR_OK (0x00000000)			
Operation 2 :	Recherche d'objet(s) de type certificat (Recherche (C_FindObjects))			
Retour 2 :	CKR_OK (0x00000000)			
Operation 3 :	Recherche d'objet(s) de type certificat (Finalisation (C_FindObjectsFinal))			
Retour 3 :	CKR_OK (0x00000000)			
Informations :				
Nombre d'objets: : 2				
Identifiant de l'objet 1 : 2				
Identifiant de l'objet 2 : 5				

Figure 63 : Ecran de résultat d'opération : Recherche d'objet(s) de type certificat

Les codes retour sont accompagnés des informations suivantes :

- Nombre d'objets
- Identifiants des objets trouvés

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Initialisation (C_FindObjectsInit)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_ATTRIBUTE_TYPE_INVALID, CKR_ATTRIBUTE_VALUE_INVALID, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Recherche (C_FindObjects)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_FAILED,

`CKR_GENERAL_ERROR`, `CKR_HOST_MEMORY`, `CKR_OPERATION_NOT_INITIALIZED`,
`CKR_SESSION_CLOSED`, `CKR_SESSION_HANDLE_INVALID`.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Finalisation* (`C_FindObjectsFinal`), est la suivante :

`CKR_CRYPTOKI_NOT_INITIALIZED`, `CKR_DEVICE_ERROR`, `CKR_DEVICE_MEMORY`,
`CKR_DEVICE_REMOVED`, `CKR_FUNCTION_FAILED`, `CKR_GENERAL_ERROR`,
`CKR_HOST_MEMORY`, `CKR_OPERATION_NOT_INITIALIZED`, `CKR_SESSION_CLOSED`,
`CKR_SESSION_HANDLE_INVALID`.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

Lorsque les objets (clés) nécessaires à la réalisation d'une signature ou d'une vérification de signature ont été trouvés, ces fonctions peuvent être réalisées. On retrouve ces objets dans les sous-menus appropriés.

La recherche d'objets de données applicatives introduit la notion de filtre de recherche sur le libellé. En effet l'utilisateur est sollicité afin de choisir un libellé comme illustré ci-dessous :

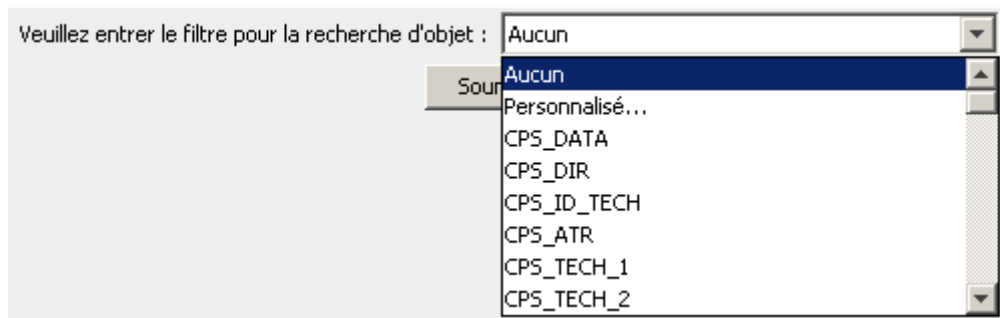


Figure 64 Ecran de saisie : Saisie du filtre de recherche pour les objets de données applicatives à rechercher

Aucun : Recherche tous les objets.

Personnalisé : Permet de saisir un libellé personnalisé.

Les filtres suivants sont des libellés prédéfinis dans le fichier de configuration du programme.

Remarque :

Certains filtres sont associés à un type de carte en particulier.

Exemple « CPS2TER_NAME_PS ».

Les fichiers de données métier de la carte CPS2Ter sont préfixés de CPS2TER_.

Les fichiers de données métier de la carte CPS3 sont préfixés de CPS_.

5.6 Empreintes

5.6.1 Générer l'empreinte d'un message court

Le but de cette fonctionnalité est de tester les fonctions `C_DigestInit` et `C_Digest` qui permettent de générer l'empreinte d'un message court en un seul appel.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Lister les algorithmes de la carte et obtenir au moins un algorithme permettant de réaliser un condensat.
- Ouvrir une session
- S'authentifier

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

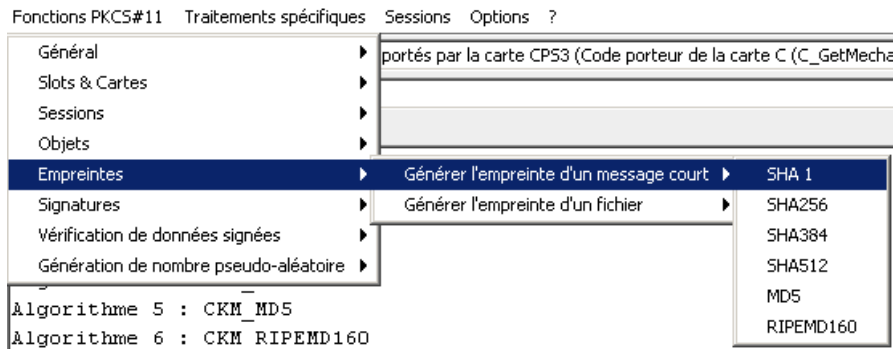


Figure 65 : Accès à la fonction de génération de l'empreinte d'un message court

L'utilisateur doit alors saisir l'information suivante :

- Le message court

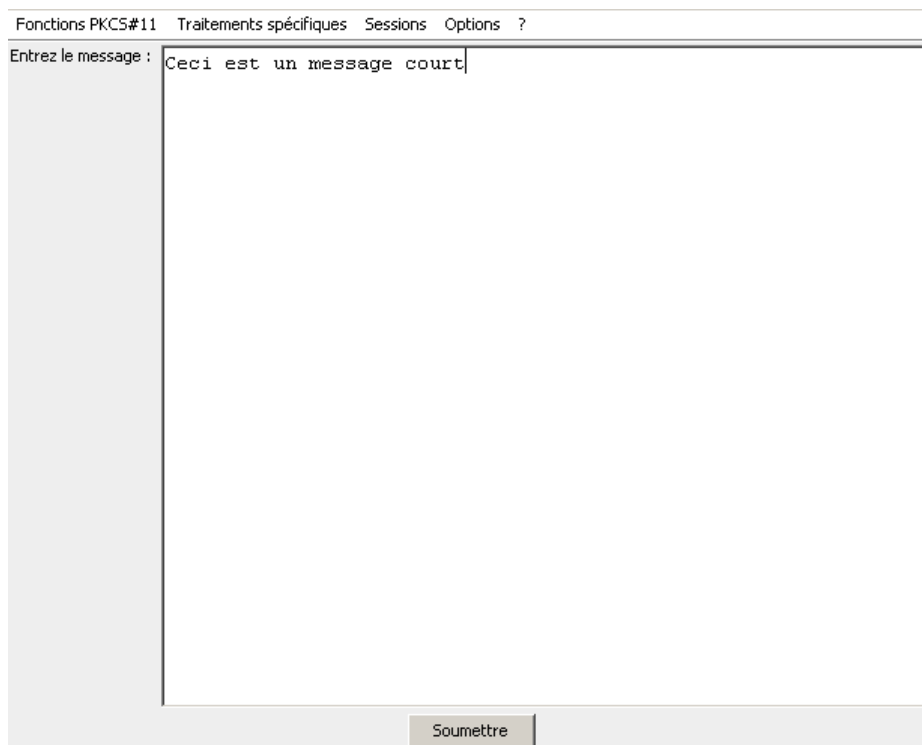


Figure 66 : Ecran de saisie : Saisie d'un message court

Lorsque la fonction *Générer l'empreinte d'un message court* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Operation 1 :	Génération de l'empreinte d'un message avec CKM_SHA_1 (Initialisation (C_DigestInit))			
Retour 1 :	CKR_OK (0x00000000)			
Operation 2 :	Génération de l'empreinte d'un message avec CKM_SHA_1 (Génération (C_Digest))			
Retour 2 :	CKR_OK (0x00000000)			
Empreinte :	69032ab70b5af6cfbf686a1e18c6fb32e7e5ebe2			

Figure 67 : Ecran de résultat d'opération : Génération de l'empreinte d'un message court

Les codes retour sont accompagnés de l'information suivante :

- L'empreinte

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Initialisation (C_DigestInit)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_MECHANISM_INVALID, CKR_MECHANISM_PARAM_INVALID, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Génération (C_Digest)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.6.2 Générer l'empreinte d'un fichier

Le but de cette fonctionnalité est de tester les fonctions *C_DigestInit*, *C_DigestUpdate* et *C_DigestFinal* qui permettent de générer une empreinte par plusieurs appels successifs à *C_DigestUpdate*.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Lister les algorithmes de la carte et obtenir au moins un algorithme permettant de réaliser un condensat.**
- **Ouvrir une session**
- **S'authentifier**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

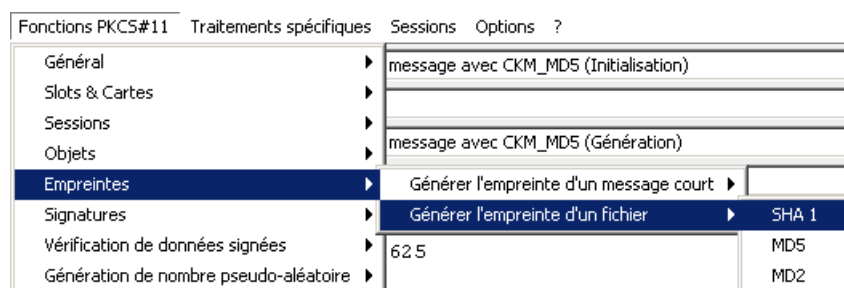


Figure 68 : Accès à la fonction de génération de l'empreinte d'un fichier

L'utilisateur doit alors saisir l'information suivante :

- Le fichier

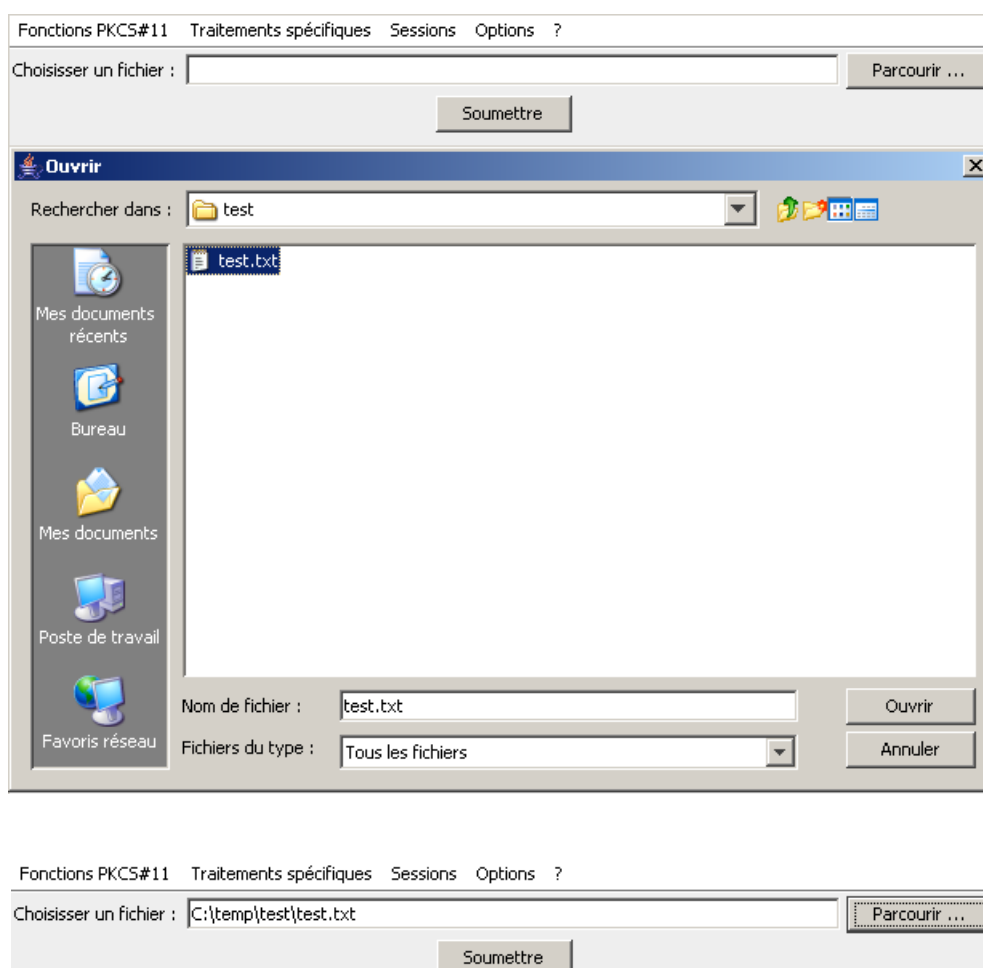


Figure 69 : Ecran de saisie : Saisie d'un fichier

Lorsque la fonction *Générer l'empreinte d'un fichier* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Operation 1 :	Génération de l'empreinte d'un fichier avec CKM_SHA_1 (Initialisation (C_DigestInit))			
Retour 1 :	CKR_OK (0x00000000)			
Operation 2 :	Génération de l'empreinte d'un fichier avec CKM_SHA_1 (Génération (C_DigestUpdate))			
Retour 2 :	CKR_OK (0x00000000)			
Operation 3 :	Génération de l'empreinte d'un fichier avec CKM_SHA_1 (Finalisation (C_DigestFinal))			
Retour 3 :	CKR_OK (0x00000000)			
Empreinte :				
98697d76fe837c8e41ab11feef90701aff96e523				

Figure 70 : Ecran de résultat d'opération : Génération de l'empreinte d'un fichier

Les codes retour sont accompagnés de l'information suivante :

- L'empreinte

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération Initialisation (C_DigestInit), est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_MECHANISM_INVALID, CKR_MECHANISM_PARAM_INVALID, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération Génération (C_DigestUpdate), est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération Finalisation (C_DigestFinal), est la suivante :

CKR_ARGUMENTS_BAD, CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.7 Signatures

5.7.1 Signer un message court

Le but de cette fonctionnalité est de tester les fonctions C_SignInit et C_Sign qui permettent de générer une signature en un seul appel.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Lister les algorithmes de la carte et obtenir au moins un algorithme permettant de réaliser une signature.
- Ouvrir une session
- S'authentifier
- Rechercher les objets Clés privées

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

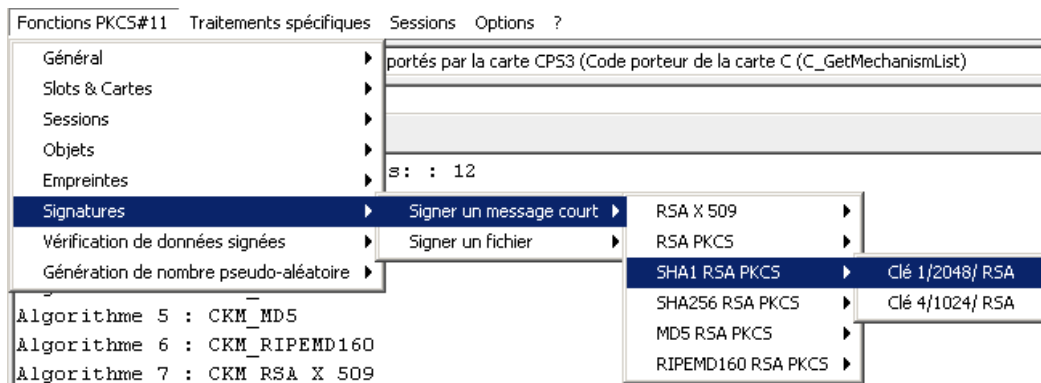


Figure 71 : Accès à la fonction de signature d'un message court

L'utilisateur doit alors saisir l'information suivante :

■ Le message court

Figure 72 : Ecran de saisie : Saisie d'un message court

Lorsque la fonction *Signer un message court* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

Fonctions PKCS#11 Traitements spécifiques Sessions Options ?	
Operation 1 :	Génération de la signature d'un message avec CKM_SHA1_RSA_PKCS (Initialisation (C_SignInit))
Retour 1 :	CKR_OK (0x00000000)
Operation 2 :	Génération de la signature d'un message avec CKM_SHA1_RSA_PKCS (Génération (C_Sign))
Retour 2 :	CKR_OK (0x00000000)
Signature :	
<pre> 34 54 4c a7 2b f3 70 e1 6f bd 9e 54 28 c9 05 b3 64 6f 42 d3 cd a3 4b 8c 5d 74 1e 41 e9 7c 7d ae 9d 57 db 7a f9 5f a3 ec 86 ae 78 ff 13 d9 65 18 05 b9 36 89 d6 c8 04 be c6 eb 6d da 45 86 69 e9 7c fd 7e 82 1a 9c c9 8a c8 67 25 a6 59 e0 0d e5 e3 12 1d e2 e8 94 9d 6d ac 53 df 85 90 01 a0 aa 97 a5 7e 33 c8 14 60 9b 84 2e c7 4a 12 d7 fb 6a 3e e9 95 4a 26 6b 95 9b 73 0f 8c 99 96 99 7b e7 cd 33 7a c8 cc 26 0f 63 2f be 17 a0 af 23 1b 7c bb bd 4c 36 b2 d1 e9 e4 15 e6 a6 63 4b 48 5b 96 4e 01 52 af cc 94 e3 ba b3 24 f2 3e 26 ad 38 0e aa 6e a5 87 2a dc 2a 98 67 84 23 ab 0f dd f0 46 ff 33 ec 90 e2 1c 9b ff ee d4 10 ea 67 27 3a 83 28 c6 43 1e 94 ef 96 7f c0 af 0c e0 8f 86 04 62 b3 7e e1 fb 9a 69 79 ec 00 a8 09 dd e2 c2 2c d5 d6 61 33 0c 13 87 e1 3d 18 ab e0 18 f1 e0 14 8f </pre>	

Figure 73 : Ecran de résultat d'opération : Signature d'un message court

Les codes retour sont accompagnés de l'information suivante :

- La signature

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Initialisation* (*C_SignInit*), est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_KEY_FUNCTION_NOT_PERMITTED, CKR_KEY_HANDLE_INVALID, CKR_KEY_SIZE_RANGE, CKR_KEY_TYPE_INCONSISTENT, CKR_MECHANISM_INVALID, CKR_MECHANISM_PARAM_INVALID, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Génération* (*C_Sign*), est la suivante :

CKR_ARGUMENTS_BAD, CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DATA_INVALID, CKR_DATA_LEN_RANGE, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN, CKR_FUNCTION_REJECTED.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

Une fois la signature réalisée, le message saisi ainsi que la signature obtenue sont enregistrés par le programme afin d'être présentés à l'utilisateur lorsqu'il lance la Vérification de la signature d'un message court.

Seuls le dernier message saisi et la dernière signature obtenue sont mémorisés.

5.7.2 Signer un fichier

Le but de cette fonctionnalité est de tester les fonctions *C_SignInit*, *C_SignUpdate* et *C_SignFinal* qui permettent de générer une signature par plusieurs appels successifs à *C_SignUpdate*.

Actions pré-requises :

- Initialiser la librairie.***
- Lister les slots avec carte et obtenir au moins un slot.***
- Lister les algorithmes de la carte et obtenir au moins un algorithme permettant de réaliser une signature.***
- Ouvrir une session***
- S'authentifier***
- Rechercher les objets Clés privées***

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

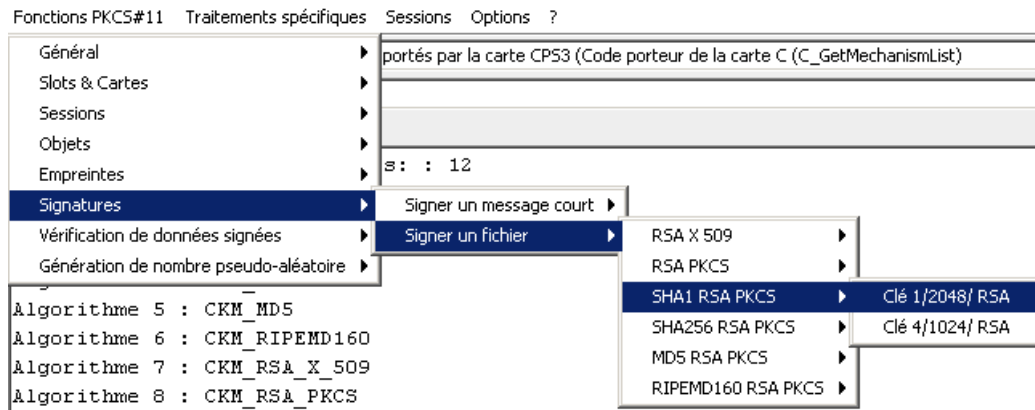


Figure 74 : Accès à la fonction de signature d'un fichier

L'utilisateur doit alors saisir l'information suivante :

- Le fichier

Lorsque la fonction *Signer un fichier* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

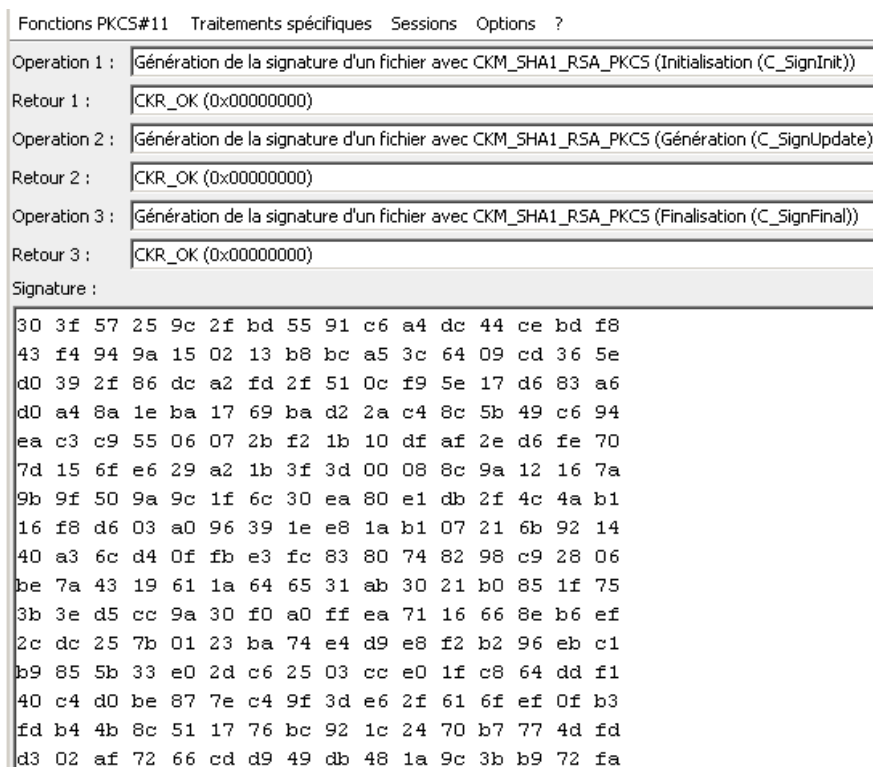


Figure 75 : Ecran de résultat d'opération : Signature d'un fichier

Les codes retour sont accompagnés de l'information suivante :

- La signature

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Initialisation (C_SignInit)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED,
CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY,
CKR_KEY_FUNCTION_NOT_PERMITTED, CKR_KEY_HANDLE_INVALID,
CKR_KEY_SIZE_RANGE, CKR_KEY_TYPE_INCONSISTENT, CKR_MECHANISM_INVALID,
CKR_MECHANISM_PARAM_INVALID, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED,
CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Génération (C_SignUpdate)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DATA_LEN_RANGE,
CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED,
CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR,
CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED,
CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Finalisation (C_SignFinal)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_BUFFER_TOO_SMALL, CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DATA_LEN_RANGE, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED,
CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN,
CKR_FUNCTION_REJECTED.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

Une fois la signature réalisée, le fichier sélectionné ainsi que la signature obtenue sont enregistrés par le programme afin d'être présentés à l'utilisateur lorsqu'il lance la Vérification de la signature d'un fichier.

Seuls le dernier fichier sélectionné et la dernière signature obtenue sont mémorisés.

5.8 Vérification de données signées

5.8.1 Vérifier la signature du message court

Le but de cette fonctionnalité est de tester les fonctions C_VerifyInit et C_Verify qui permettent de vérifier une signature en un seul appel.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Lister les algorithmes de la carte et obtenir au moins un algorithme permettant de réaliser une vérification de signature.
- Ouvrir une session
- S'authentifier
- Optionnel : Rechercher les objets Clés Privées + Signature d'un message court
- Rechercher les objets Clés publiques

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

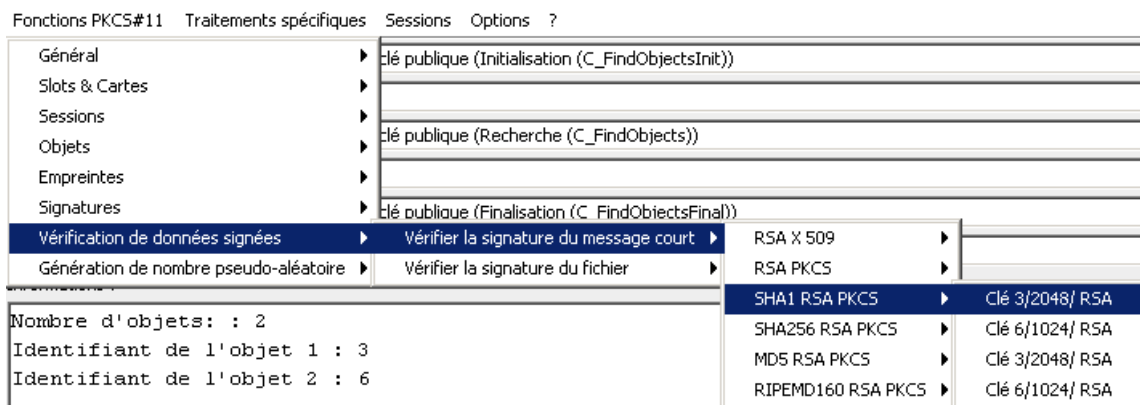


Figure 76 : Accès à la vérification de la signature du message court

L'utilisateur doit alors saisir les informations suivantes :

- Le message court
- La signature

Le message court et la signature pré remplis sont ceux qui ont été mémorisés lors de la dernière signature d'un message court. L'utilisateur est libre de les modifier avant d'effectuer l'opération de vérification.

Fonctions PKCS#11 Traitements spécifiques Sessions Options ?

Entrez le message : Ceci est un message court

Signature :

```
34 54 4c a7 2b f3 70 e1 6f bd 9e 54 28 c9 05 b3
64 6f 42 d3 cd a3 4b 8c 5d 74 1e 41 e9 7c 7d ae
9d 57 db 7a f9 5f a3 ec 86 ae 78 ff 13 d9 65 18
05 b9 36 89 d6 c8 04 be c6 eb 6d da 45 86 69 e9
7c fd 7e 82 1a 9c c9 8a c8 67 25 a6 59 e0 0d e5
e3 12 1d e2 e8 94 9d 6d ac 53 df 85 90 01 a0 aa
97 a5 7e 33 c8 14 60 9b 84 2e c7 4a 12 d7 fb 6a
3e e9 95 4a 26 6b 95 9b 73 0f 8c 99 96 99 7b e7
cd 33 7a c8 cc 26 0f 63 2f be 17 a0 af 23 1b 7c
bb bd 4c 36 b2 d1 e9 e4 15 e6 a6 63 4b 48 5b 96
```

Soumettre

Figure 77 : Ecran de saisie : Saisie pour la vérification de signature d'un message court

Lorsque la fonction *Vérifier la signature d'un message court* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

Fonctions PKCS#11 Traitements spécifiques Sessions Options ?	
Operation 1 :	Vérification de la signature d'un message avec CKM_SHA1_RSA_PKCS (Initialisation (C_VerifyInit))
Retour 1 :	CKR_OK (0x00000000)
Operation 2 :	Vérification de la signature d'un message avec CKM_SHA1_RSA_PKCS (Vérification (C_Verify))
Retour 2 :	CKR_OK (0x00000000)

Figure 78 : Ecran de résultat d'opération : Vérification de signature d'un message court

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Initialisation (C_VerifyInit)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR,
 CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED,
 CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY,
 CKR_KEY_FUNCTION_NOT_PERMITTED, CKR_KEY_HANDLE_INVALID,
 CKR_KEY_SIZE_RANGE, CKR_KEY_TYPE_INCONSISTENT, CKR_MECHANISM_INVALID,
 CKR_MECHANISM_PARAM_INVALID, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED,
 CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Vérification (C_Verify)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DATA_INVALID,
 CKR_DATA_LEN_RANGE, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY,
 CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED,
 CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED,
 CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SIGNATURE_INVALID,
 CKR_SIGNATURE_LEN_RANGE.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.8.2 Vérifier la signature du fichier

Le but de cette fonctionnalité est de tester les fonctions `C_VerifyInit`, `C_VerifyUpdate` et `C_VerifyFinal` qui permettent de vérifier une signature par plusieurs appels successifs à `C_VerifyUpdate`.

Actions pré-requises :

- Initialiser la librairie.
- Lister les slots avec carte et obtenir au moins un slot.
- Lister les algorithmes de la carte et obtenir au moins un algorithme permettant de réaliser une vérification de signature.
- Ouvrir une session
- S'authentifier
- Optionnel : Rechercher les objets Clés Privées + Signature d'un fichier
- Rechercher les objets Clés publiques

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

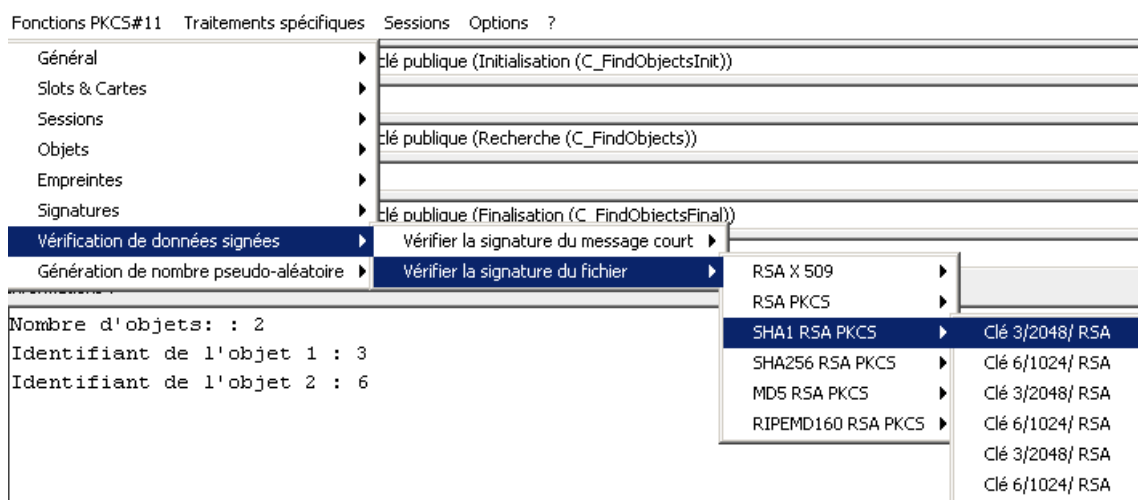


Figure 79 : Accès à la vérification de la signature du fichier

L'utilisateur doit alors saisir les informations suivantes :

- Le fichier
- La signature

Le fichier et la signature pré remplis sont ceux qui ont été mémorisés lors de la dernière signature d'un fichier. L'utilisateur est libre de les modifier avant d'effectuer l'opération de vérification.

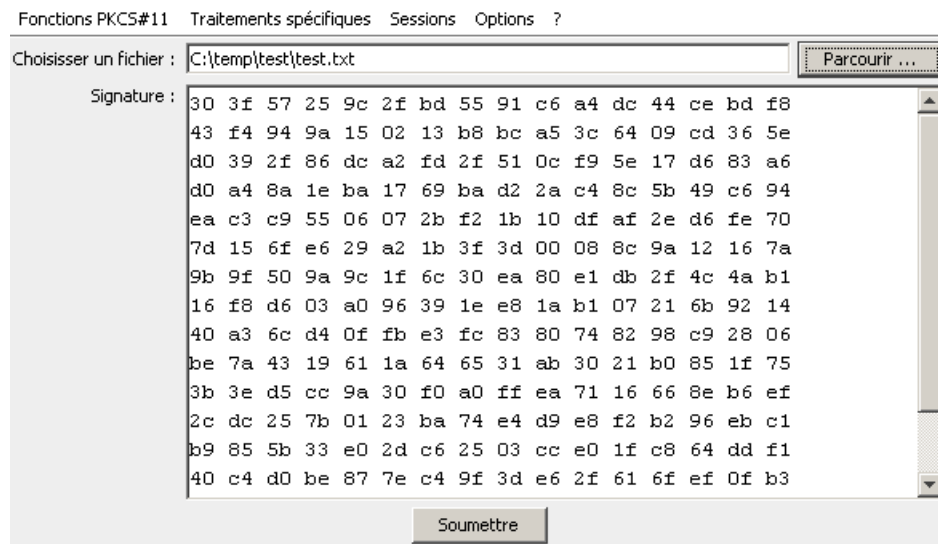


Figure 80 : Ecran de saisie : Saisie pour la vérification de la signature d'un fichier

Lorsque la fonction *Vérifier la signature d'un fichier* s'exécute sans erreur, les codes retour sont affichés comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Operation 1 :	Vérification de la signature d'un fichier avec CKM_SHA1_RSA_PKCS (Initialisation (C_VerifyInit))			
Retour 1 :	CKR_OK (0x00000000)			
Operation 2 :	Vérification de la signature d'un fichier avec CKM_SHA1_RSA_PKCS (Vérification (C_VerifyUpdate))			
Retour 2 :	CKR_OK (0x00000000)			
Operation 3 :	Vérification de la signature d'un fichier (Finalisation (C_VerifyFinal))			
Retour 3 :	CKR_OK (0x00000000)			

Figure 81 : Ecran de résultat d'opération : Vérification de la signature d'un fichier

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Initialisation (C_VerifyInit)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR,
 CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED,
 CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY,
 CKR_KEY_FUNCTION_NOT_PERMITTED, CKR_KEY_HANDLE_INVALID,
 CKR_KEY_SIZE_RANGE, CKR_KEY_TYPE_INCONSISTENT, CKR_MECHANISM_INVALID,
 CKR_MECHANISM_PARAM_INVALID, CKR_OPERATION_ACTIVE, CKR_PIN_EXPIRED,
 CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Vérification (C_VerifyUpdate)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DATA_LEN_RANGE,
 CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED,
 CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR,
 CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED,
 CKR_SESSION_HANDLE_INVALID.

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de l'opération *Finalisation (C_VerifyFinal)*, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DATA_LEN_RANGE, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_NOT_INITIALIZED, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_SIGNATURE_INVALID, CKR_SIGNATURE_LEN_RANGE.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.9 Générateur de nombre pseudo-aléatoire

5.9.1 Normale

Le but de cette fonctionnalité est de tester la fonction C_GenerateRandom qui permet de générer un nombre aléatoire depuis la carte.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session**
- **S'authentifier**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

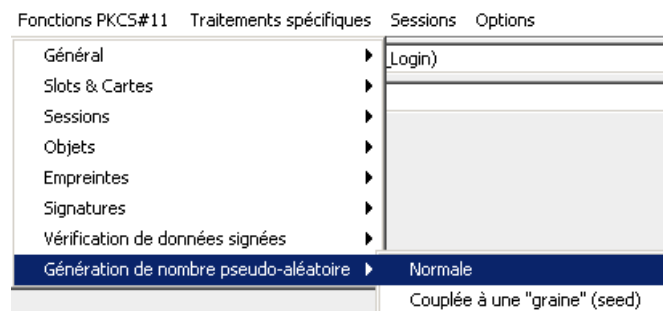


Figure 82 : Accès à la fonction de génération de nombre pseudo-aléatoire normale

Lorsque la fonction *Génération de nombre pseudo-aléatoire Normale* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

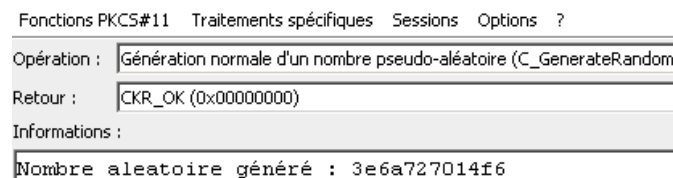


Figure 83 : Ecran de résultat d'opération : Génération normale de nombre pseudo-aléatoire

Le code retour est accompagné de l'information suivante :

- Le nombre aléatoire généré

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_ACTIVE, CKR_RANDOM_NO_RNG, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

5.9.2 Couplée à une « graine » (seed)

Le but de cette fonctionnalité est de tester la fonction C_SeedRandom qui permet de générer un nombre en mixant une « graine » (seed) et le nombre aléatoire généré depuis la carte.

Actions pré-requises :

- **Initialiser la librairie.**
- **Lister les slots avec carte et obtenir au moins un slot.**
- **Ouvrir une session**
- **S'authentifier**

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

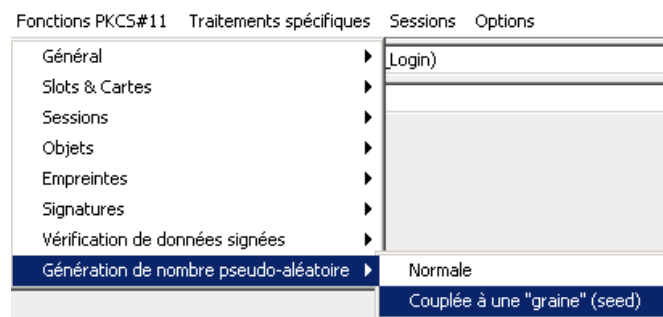


Figure 84 : Accès à la fonction de génération de nombre pseudo-aléatoire couplée à une graine

L'utilisateur doit alors saisir l'information suivante :

- La graine sous la forme d'une chaîne de caractère.

The image shows a dialog box with the title 'Saisir la graine (seed) :'. It contains a text input field with the value 'ceci est une graine' and a button labeled 'Soumettre'.

Figure 85 : Ecran de saisie : Saisie d'une graine (seed)

Lorsque la fonction *Génération de nombre pseudo-aléatoire* couplée à une graine s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options
Opération :	Génération couplée à une graine (seed) d'un nombre pseudo-aléatoire (C_SeedRandom)		
Retour :	CKR_OK (0x00000000)		
Informations :	Nombre aleatoire généré : 636563692065737420756e6520677261696e65		

Figure 86 : Ecran de résultat d'opération : Génération avec graine d'un nombre pseudo-aléatoire

Le code retour est accompagné de l'information suivante :

- Le nombre aléatoire généré

La liste des codes retour faisant référence à une erreur PKCS#11, et qu'il est possible de rencontrer à l'issue de l'exécution de cette opération, est la suivante :

CKR_ARGUMENTS_BAD, CKR_CRYPTOKI_NOT_INITIALIZED, CKR_DEVICE_ERROR, CKR_DEVICE_MEMORY, CKR_DEVICE_REMOVED, CKR_FUNCTION_CANCELED, CKR_FUNCTION_FAILED, CKR_GENERAL_ERROR, CKR_HOST_MEMORY, CKR_OPERATION_ACTIVE, CKR_RANDOM_SEED_NOT_SUPPORTED, CKR_RANDOM_NO_RNG, CKR_SESSION_CLOSED, CKR_SESSION_HANDLE_INVALID, CKR_USER_NOT_LOGGED_IN.

Pour plus d'information concernant la signification de ces codes retour vous pouvez vous référer au document de référence [1].

6 Module Traitements spécifiques

Dans ce module, l'utilisateur peut-être sollicité par l'intermédiaire de boîtes de dialogue. Il peut, dans ces cas là, s'il le désire, interrompre le traitement en cliquant sur le bouton « Annuler » prévu à cet effet. Cette action d'annulation se traduit par un code retour : « Interruption utilisateur ».

Certains codes retour son interprétés et retranscrits. Pour les autres, il convient de se référer au document de référence [1].

6.1 Authentification

Le but de ce menu est de regrouper un ensemble de traitements liés à l'authentification de l'utilisateur par rapport à une carte.

Ces traitements sont appliqués sur la « première carte ». La « première carte » est la carte que le programme d'exemple rencontre en premier lorsqu'il fait l'inventaire des cartes supportées et connectées.

6.1.1 Statut d'authentification par rapport à une carte

Le but de cette fonctionnalité est de déterminer l'état de l'authentification de l'utilisateur par rapport à une carte.

Aucune intervention de l'utilisateur n'est requise (ex : demande de branchement lecteur, introduction carte, saisie code porteur).

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

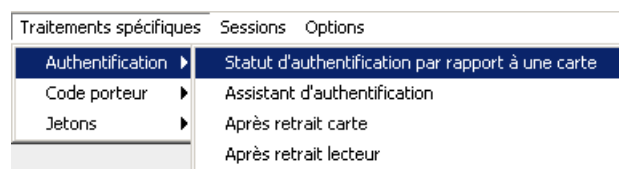


Figure 87 : Accès au traitement spécifique : Statut d'authentification

Lorsque la fonction Statut d'authentification s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous (ici, cas d'un utilisateur non authentifié) :

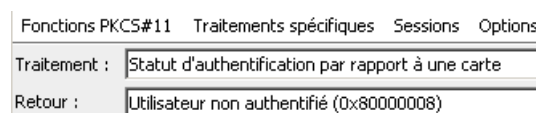


Figure 88 : Ecran de résultat de traitement spécifique : Statut d'authentification

Si le code retour est un code d'erreur PKCS#11 inconnu, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

6.1.2 Assistant d'authentification

Cette fonctionnalité exécute tout d'abord le traitement précédent. Tant que le code retour ne correspond pas au fait que l'utilisateur est authentifié et s'il ne s'agit pas d'un code d'erreur PKCS#11, l'utilisateur est sollicité pour une action (connexion lecteur/carte, saisie du code porteur).

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

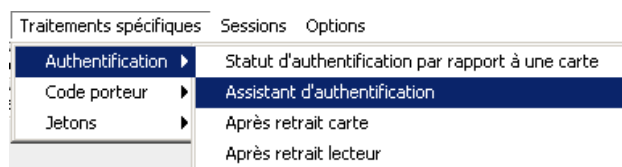


Figure 89 : Accès au traitement spécifique : Assistant d'authentification

L'utilisateur est sollicité comme suit :

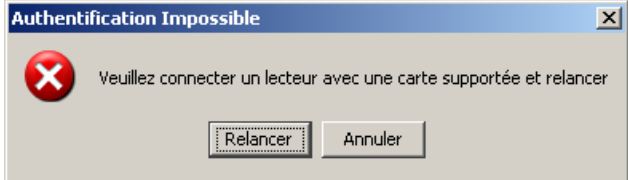
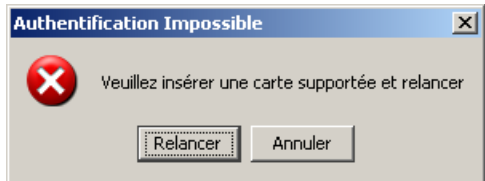
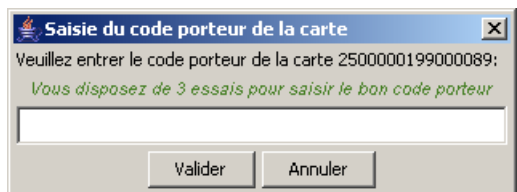
Code Retour	Instructions
Aucun lecteur présent	<p>Veillez connecter un lecteur avec une carte supportée.</p> 
Aucune carte supportée	<p>Veillez insérer une carte supportée.</p> 
Utilisateur non authentifié	<p>Saisie du code porteur</p> 

Figure 90 : Ecrans et boîtes de dialogue du traitement spécifique : Assistant d'authentification

Après réponse aux instructions ci-dessus, le traitement est invoqué à nouveau jusqu'à ce que l'issue en soit que l'utilisateur est authentifié.

La gestion de l'authentification protégée n'est pas traitée dans ce programme d'exemple.

Lorsque la fonction *Assistant d'authentification* s'exécute sans erreur et sans nécessité d'intervention supplémentaire de l'utilisateur, le code retour est affiché comme illustré sur la figure ci-dessous :

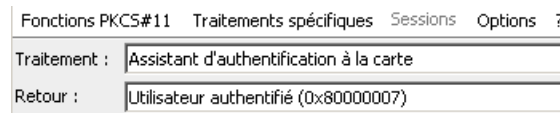


Figure 91 : Ecran de résultat de traitement spécifique : Assistant d'authentification

Si le code retour est un code d'erreur PKCS#11 l'assistant d'authentification s'arrête, et le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

L'assistant d'authentification intègre la politique d'avertissements décrite dans la partie Etat de saisie du code porteur ci-après, concernant le nombre d'essais restants. Il est considéré que le nombre de tentatives avant blocage est de 3.

6.1.3 Après retrait carte

Le but de cette fonctionnalité est de gérer la reprise après le retrait d'une carte. Le programme d'exemple doit assurer la « réauthentification ». Il est à considérer que la carte peut-être réinsérée dans un autre lecteur.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

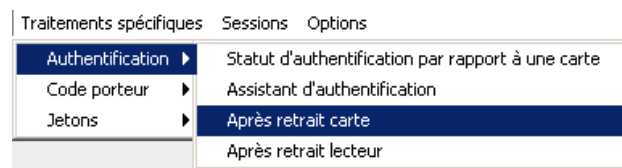
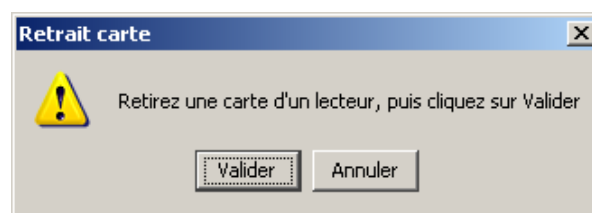


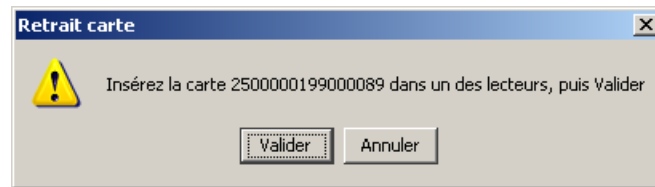
Figure 92 : Accès au traitement spécifique : Authentification après retrait carte

Cette fonctionnalité se décompose en plusieurs étapes :

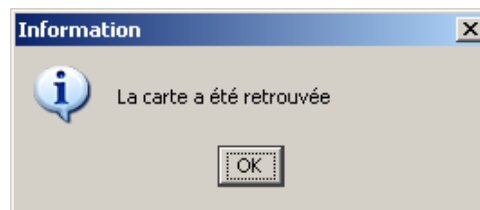
- Le programme sauvegarde l'état de raccordement lecteurs/cartes.
- Il est demandé à l'utilisateur de retirer une carte d'un lecteur.



- L'utilisateur retire une carte et clique sur Valider.
- Le programme vérifie qu'il s'agit bien d'un retrait et détecte la carte manquante à l'aide de la sauvegarde effectuée au préalable.
- Il est demandé à l'utilisateur de réintroduire cette carte.



- L'utilisateur introduit la carte et clique sur Valider.
- Le programme vérifie qu'il s'agit d'une insertion de la carte recherchée à l'aide de la sauvegarde effectuée au préalable.
- Si le code retour correspond à « Carte trouvée », l'utilisateur est averti comme ci-dessous :



- Lorsque l'utilisateur clique sur OK, le programme ouvre une session, en lecture, et demande le code porteur afin de « ré-authentifier » l'utilisateur. L'authentification est réalisée comme décrite au paragraphe *Assistant d'authentification* sur la carte "retrouvée".
- En cas d'échec de la recherche, l'écran suivant s'affiche:

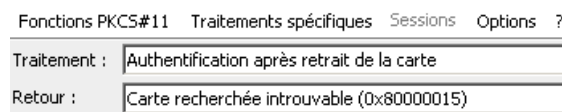


Figure 93 : Ecran de résultat de traitement spécifique : Echec de l'authentification après retrait de carte

Si le code retour du traitement est un code d'erreur PKCS#11 non retranscrit, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

6.1.4 Après retrait lecteur

Le but de cette fonctionnalité est de gérer la reprise après le retrait d'un lecteur. Le programme d'exemple doit assurer la « réauthentification ». Il est à considérer que la carte peut-être réinsérée dans un autre lecteur.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

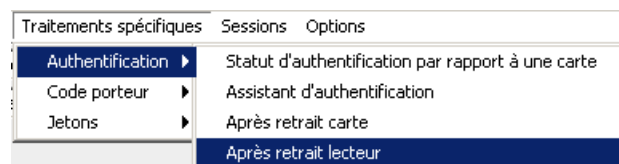
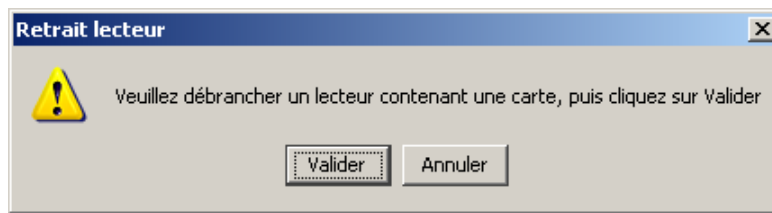


Figure 94 : Accès au traitement spécifique : Authentification après retrait lecteur

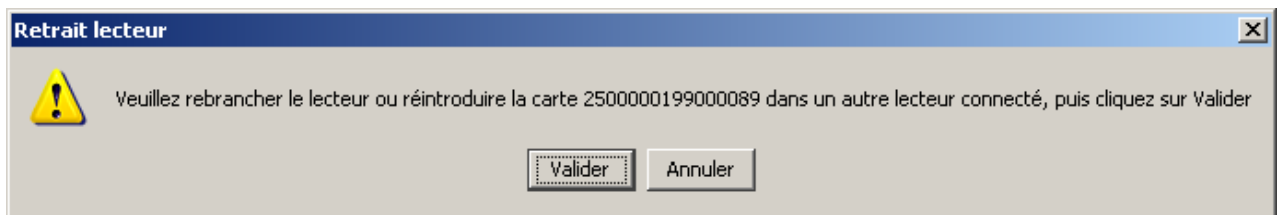
Cette fonctionnalité se décompose en plusieurs étapes :

- Le programme sauvegarde l'état de raccordement lecteurs/cartes.

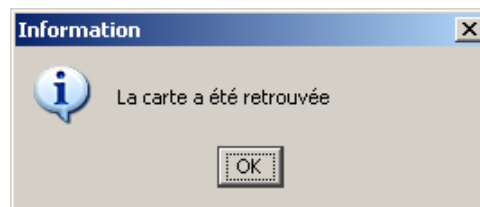
- Il est demandé à l'utilisateur de retirer un lecteur contenant une carte.



- L'utilisateur retire le lecteur et clique sur Valider.
- Le programme vérifie qu'il s'agit bien d'un retrait et détecte la carte manquante à l'aide de la sauvegarde effectuée au préalable.
- Il est alors demandé à l'utilisateur de rebrancher le lecteur ou de réintroduire cette carte dans un autre lecteur connecté.



- L'utilisateur rebranche le lecteur ou de réintroduit la carte dans un autre lecteur connecté et clique sur Valider.
- Le programme vérifie qu'il s'agit d'une insertion de la carte recherchée à l'aide de la sauvegarde effectuée au préalable.
- Si le code retour correspond à « Carte trouvée », l'utilisateur est averti comme ci-dessous :



- Lorsque l'utilisateur clique sur OK, le programme ouvre une session, en lecture, et demande le code porteur afin de ré-authentifier l'utilisateur. L'authentification est réalisée comme décrite au paragraphe *Assistant d'authentification*, mais sur la carte retrouvée.
- En cas d'échec de la recherche obtient l'écran suivant :

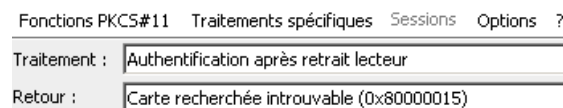


Figure 95 : Ecran de résultat de traitement spécifique : Echec de l'authentification après retrait de lecteur

Si le code retour du traitement est un code d'erreur PKCS#11 non retranscrit, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

6.2 Gestion du code porteur

Avant d'exécuter les 2 traitements suivants, le programme d'exemple ferme toutes les sessions ouvertes sur la « première carte ». Cela a pour effet de déconnecter toutes les sessions actives. L'utilisateur se retrouve alors dans les conditions initiales requises qui lui permettent de tester les 2 traitements suivants.

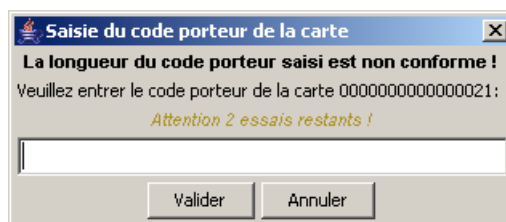
6.2.1 Etat de saisie du code porteur

Le but de cette fonctionnalité est d'illustrer comment transposer les statuts PKCS#11 des informations carte, en une information précise et parlante pour l'utilisateur.

Cette fonction réalise une opération d'authentification, en invitant l'utilisateur à entrer volontairement un code erroné afin de lui faire constater la transposition des statuts PKCS#11.

Cette fonction réalise également un contrôle de conformité sur le code porteur. Les erreurs possibles que peut générer ce contrôle sont les suivantes :

- Code porteur invalide
- La longueur du code porteur est invalide



L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

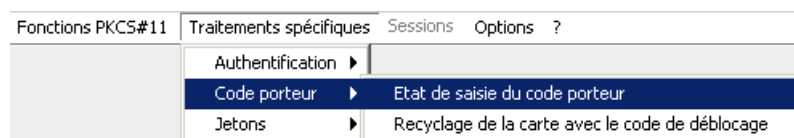


Figure 96 : Accès au traitement spécifique : Etat de saisie du code porteur

L'utilisateur est sollicité pour choisir une option qui détermine le nombre de tentatives possibles avant le blocage de la carte.

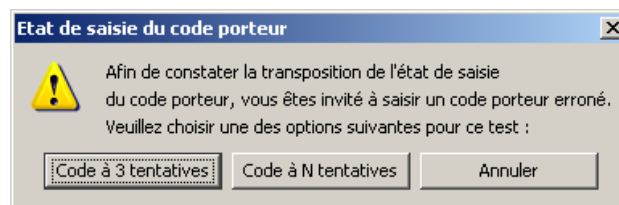


Figure 97 : Boîte de dialogue pour le traitement spécifique : Etat de saisie du code porteur

Les boîtes dialogue transposant le statut des informations cartes sont les suivantes :

Code à N tentatives	Code à 3 tentatives

Figure 98 : Boîtes de dialogue de saisie pour traitement spécifique : Etat de saisie du code porteur

Lorsque la fonction *Etat de saisie du code porteur* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Traitement :	Etat de saisie du code porteur			
Retour :	Utilisateur authentifié (0x80000007)			

Figure 99 Ecran de résultat de traitement spécifique : Etat de saisie du code porteur

Si le code retour du traitement est un code d'erreur PKCS#11 non retranscrit, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

6.2.2 Recyclage de la carte avec le code de déblocage (PUK)

Le but de cette fonctionnalité est de réaliser le recyclage du code porteur.

Avant tout, le programme d'exemple recherche la « première carte » puis ferme toutes les sessions ouvertes dessus. Cela a pour effet de déconnecter toutes les sessions actives. L'utilisateur se retrouve alors dans les conditions initiales requises qui lui permettent d'effectuer le recyclage (choix d'implémentation du programme d'exemple).

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

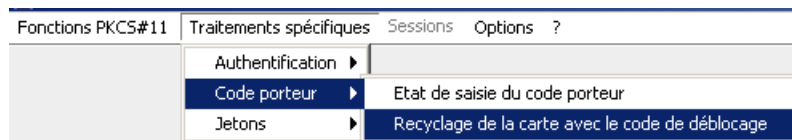


Figure 100 : Accès au traitement spécifique : Recyclage de la carte avec le code de déblocage

Cette fonctionnalité se décompose en plusieurs étapes :

- Il est demandé à l'utilisateur de fournir le code de déblocage (SO) et le nouveau code porteur de la carte (identifiée par son numéro de série).



Figure 101 : Boîte de dialogue de saisie pour traitement spécifique : Recyclage de la carte avec le code de déblocage

- Contrôle de la conformité du code de déblocage.
- Contrôle en vue de s'assurer que les 2 nouveaux codes porteur saisis sont identiques.
- Appel à la fonction de réinitialisation du code porteur.

Lorsque la fonction *Recyclage du code porteur* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

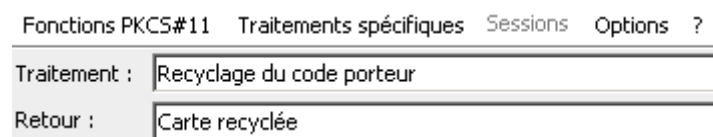


Figure 102 : Ecran de résultat de traitement spécifique : Recyclage de la carte avec le code de déblocage

Si le code retour du traitement est un code d'erreur PKCS#11 non retranscrit, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

Le code de déblocage dispose également d'un contrôle sur le nombre d'essais. Le nombre de tentatives considéré est du type n .

6.3 Gestion de jetons

6.3.1 Lecture d'un jeton d'établissement

Le but de cette fonctionnalité est de réaliser la lecture d'un jeton d'établissement (objet de données applicatives) stocké sur la carte. Elle se matérialise par l'affichage de l'attribut CKA_VALUE à l'écran.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

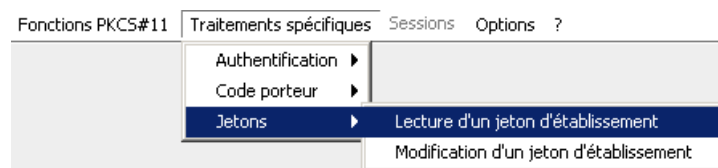


Figure 103 : Accès au traitement spécifique : Lecture d'un jeton d'établissement

Lorsque la fonction *Lecture d'un jeton d'établissement* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

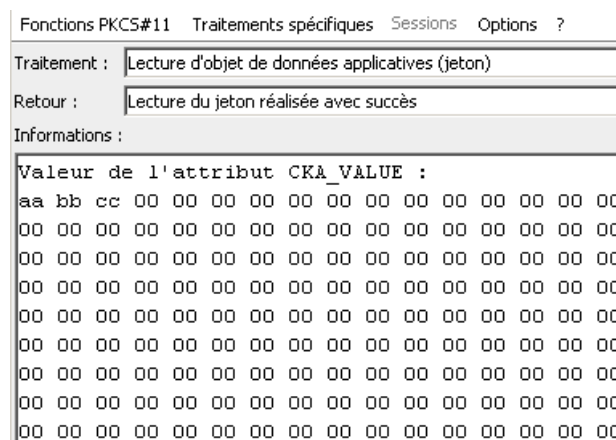


Figure 104 : Ecran de résultat de traitement spécifique : Lecture d'un jeton d'établissement

Le code retour est accompagné de l'information suivante :

- La valeur de l'attribut CKA VALUE.

Si le code retour du traitement est un code d'erreur PKCS#11 inconnu, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

6.3.2 Modification d'un jeton d'établissement

Le but de cette fonctionnalité est de réaliser la modification de l'attribut CKA_VALUE d'un jeton d'établissement (objet de données applicatives) de la carte.

Cette fonction n'est pas réalisable en mode sans contact.

L'accès à cette fonction se fait comme illustré sur la figure ci-dessous :

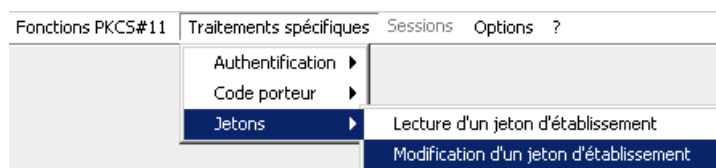


Figure 105 : Accès au traitement spécifique : Modification d'un jeton d'établissement

L'utilisateur doit alors saisir l'information suivante :

- La valeur de l'attribut CKA_VALUE sous forme hexadécimale.

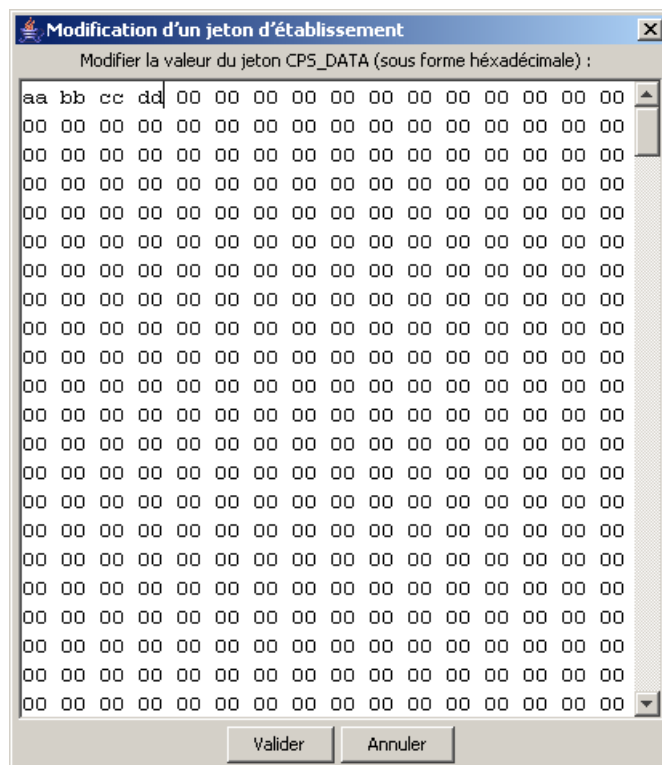


Figure 106 : Boîte de dialogue de saisie pour traitement spécifique : Modification d'un jeton d'établissement

Lorsque la fonction *Modification d'un jeton d'établissement* s'exécute sans erreur, le code retour est affiché comme illustré sur la figure ci-dessous :

Fonctions PKCS#11	Traitements spécifiques	Sessions	Options	?
Traitement : Modification d'objet de données applicatives (jeton)				
Retour : Modification du jeton réalisée avec succès				
Informations :				
Valeur de l'attribut CKA_VALUE après relecture : aa bb cc dd 00				

Figure 107 : Ecran de résultat de traitement spécifique : Modification d'un jeton d'établissement

Le code retour est accompagné de l'information suivante :

- La valeur de l'attribut CKA_VALUE après relecture de l'objet.

Si le code retour du traitement est un code d'erreur PKCS#11 non retranscrit, le résultat produit à l'écran mentionne le code retour tel quel. Pour plus d'information concernant la signification de ce code retour, vous pouvez vous référer au document de référence [1].

7 Description des sources

7.1 Sources Java

7.1.1 Présentation

Les sources java du programme d'exemple gèrent l'interface graphique et font le lien avec les interfaces PKCS#11. Ces interfaces sont au nombre de 2 :

- La première, la librairie open source IAIK, est dédiée au module *Fonctions PKCS#11*.
- La seconde, la librairie JNI développée en C, est dédiée au module des *Traitements spécifiques*.

7.1.2 Prise en main

Le répertoire de sources java contient les éléments suivants :

/bin	Répertoire destination des fichiers binaires.
/config	Répertoire consacré aux fichiers de configuration.
/lib	Répertoire contenant les fichiers .jar nécessaires à l'exécution du programme.
/src	Répertoire contenant les sources du projet.
cps3_pkcs11_w32.dll	Librairie PKCS#11 de la CPS3. C'est la librairie chargée par défaut par le programme au démarrage.
lancement.bat	Fichier de lancement du programme d'exemple. Attention : Ce fichier n'est pas à utiliser dans son emplacement d'origine. Il est destiné à être placé au même niveau que le fichier jar généré.
pkcs11wrapper.dll	Brique C JNI de la librairie IAIK
programmeExempleJNI.dll	Brique C JNI du programme d'exemple, consacrée aux traitements spécifiques
.classpath	Fichier classpath pour Eclipse
.project	Fichier projet pour Eclipse

Les sources Java sont livrées avec un fichier Eclipse .project.

Il suffit donc d'importer dans Eclipse le répertoire contenant les sources Java :

1. File>Import...>General>Existing project into Workspace
2. Sélectionner le répertoire contenant les sources Java pour le champ Select root directory.

3. Finish

Il est possible qu'Eclipse présente un problème de Java Build Path.

Il convient d'adapter le classpath (Project>Properties>Java Build Path>Librairies) en spécifiant la JRE java que vous utilisez.

Vous pouvez désormais éditer, compiler, débbugger, et exécuter les sources java du programme d'exemple.

Pour lancer le programme dans Eclipse, il faut exécuter la classe de lancement fr.asip.cps3.exemple.Lancement.java en faisant un « Run as » « Java application ».

7.1.3 Description

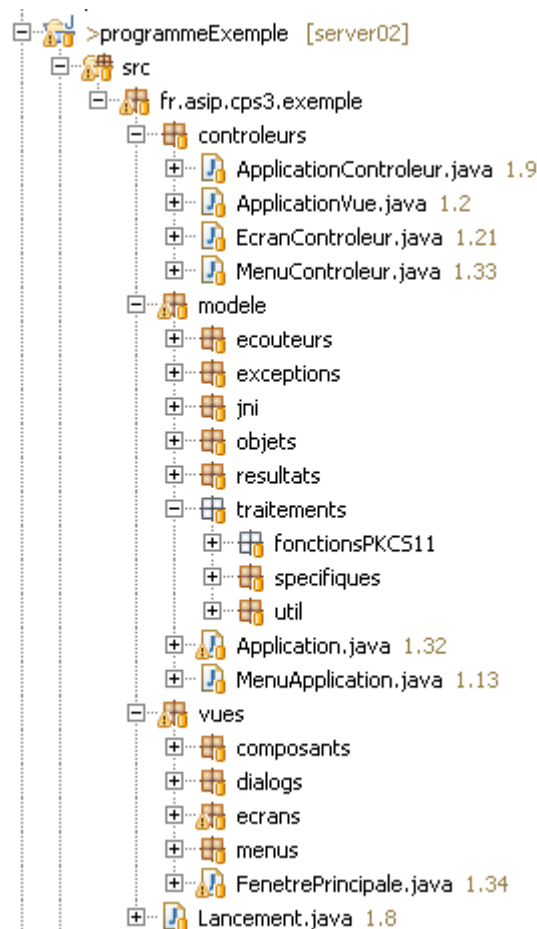


Figure 108 : Arborescence des sources Java du programme d'exemple

Le programme d'exemple s'articule selon trois couches :

- **modèle** : Contient toute la logique métier (traitements, objets manipulés...).
- **contrôleurs** : Contient les contrôleurs chargés de faire le lien entre le modèle et la vue.
- **vues** : Contient toutes les classes relatives à l'affichage.

En tant qu'éditeur de logiciels, les parties de code susceptibles de vous intéresser, et intimement liées à la librairie de la carte CPS3, sont décrites dans les paragraphes ci-après.

7.1.3.1 Les objets manipulés

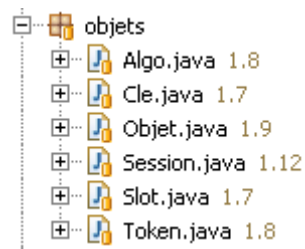


Figure 109 : Arborescence des sources Java des objets manipulés

Package : fr.asip.cps3.exemple.modele.objets

Algo : Classe modélisant un algorithme.

Cle : Classe modélisant une clé (Secrète/Privée/Publique).

Objet : Classe modélisant un objet de données applicatives.

Session : Classe modélisant un objet de données applicatives.

Token : Classe modélisant un token(carte).

7.1.3.2 Code relatif au module Fonctions PKCS#11

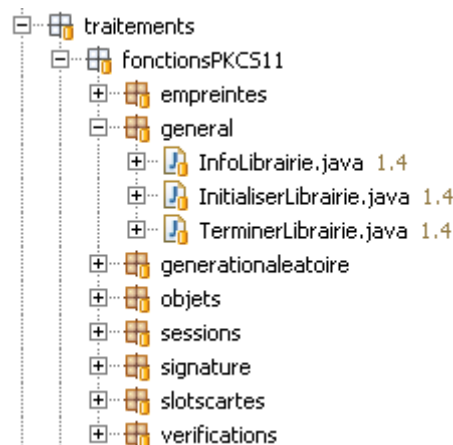


Figure 110 : Arborescence des sources Java du module Fonctions PKCS#11

Package : fr.asip.cps3.exemple.modele.traitements.fonctionsPKCS#11

Ce package est organisé comme le menu du programme qui est dédié à ce module. Au sein de chaque sous package (general, empreintes...) se situent les classes qui font « interface » vers une fonction élémentaire PKCS#11 de la CryptoLib CPS3. Ces classes font appel aux fonctions de la librairie IAIK et traitent le code retour.

7.1.3.3 Code relatif au module *Traitement spécifiques*

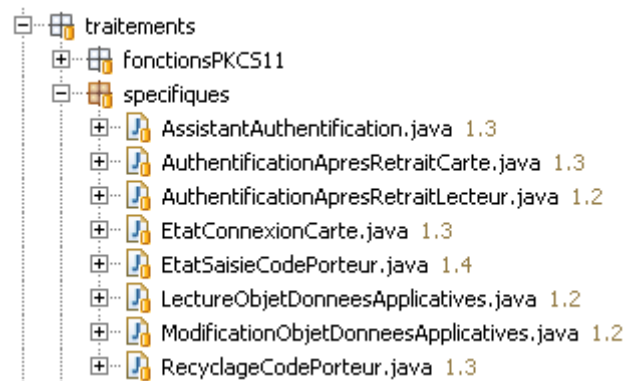


Figure 111 : Arborescence des sources Java du module Traitements spécifiques

Package : fr.asip.cps3.exemple.modele.traitements.specifiques

Dans ce package, à chaque traitement spécifique correspond une classe. Ces classes font appel aux fonctions de la brique JNI chargée de réaliser les traitements spécifiques au niveau C.

7.1.3.4 Classe utilitaire

Package : fr.asip.cps3.exemple.modele.traitements.util

La classe TraitementsUtil.java est utilisée par les classes de traitements pour notamment faire de la retranscription de code ou du formatage.

7.1.4 Génération du fichier JAR

7.1.4.1 Pré-requis

Le programme d'exemple a été réalisé en Java 1.5, et nécessite, de ce fait, d'un kit de développement Java (JDK) 1.5 afin de compiler les sources et de générer le fichier jar.

Les fichiers .bat et/ou .sh de génération (buildJAR) du fichier JAR doivent-être édités afin de s'assurer que les variables `JDK_HOME` et `JAR_HOME` sont correctement renseignées. C'est-à-dire qu'elles doivent référencer les chemins d'accès aux répertoires contenant les exécutables `javac` et `jar` sur la machine contenant les sources du programme d'exemple.

7.1.4.2 Avec le script de génération

- Sous Windows :

Lancer le script buildJAR.bat situé dans le répertoire des sources Java.

- Sous Linux :

Dans une fenêtre de Terminal, lancer le script buildJAR.sh situé dans le répertoire des sources Java.

7.1.4.3 Depuis Eclipse

Il est possible de générer le fichier JAR depuis Eclipse de la façon suivante :

1. Sélectionner le projet.

2. File>Export...>Java>JAR File puis Next.
3. Puis sélectionnez comme suit :

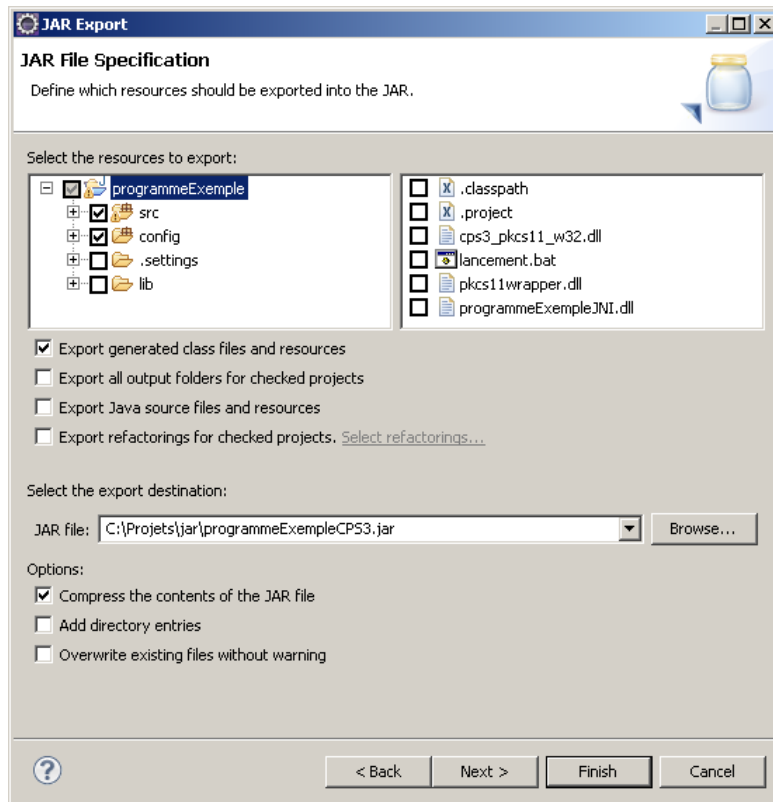


Figure 112 : Export des sources Java sous forme de fichier JAR

4. Puis 2 fois Next.
5. Puis sélectionnez Use existing manifest from workspace et spécifiez le fichier MANIFEST.MF situé dans le répertoire des sources Java.
6. Finish.

7.1.4.4 Utilisation

Pour pouvoir exécuter le fichier JAR il faut le positionner dans un répertoire tel que décrit au chapitre *Installation*. Les fichiers et répertoires nécessaires se situent dans le répertoire des sources Java du programme d'exemple (la racine du projet Java).

7.2 Sources C

7.2.1 Présentation

Les sources C du programme d'exemple gèrent les traitements spécifiques. Un pont JNI est établi entre la partie Java et la partie C.

Cette liaison permet :

- A la partie C, de déclencher des demandes d'actions de la part de l'utilisateur (insertion lecteurs/cartes, saisie du code porteur...).
- A la partie Java, de faire appel aux traitements spécifiques développés en C.

7.2.2 Prise en main

7.2.2.1 Commun

Le répertoire des sources C contient un sous répertoire /src pour les sources du code commun à toutes les plateformes.

7.2.2.2 Windows 32bits

Le répertoire des sources C contient un sous répertoire /win32 pour la version Windows. Ce sous répertoire contient le fichier Solution programmeExempleJNI.sln à ouvrir avec Microsoft Visual Studio.

Le répertoire /win32/src contient les sources du code spécifique plateforme Windows 32bits.

7.2.2.3 Linux

Le répertoire des sources C contient un sous répertoire /linux pour la version Linux. Ce sous répertoire contient les fichiers de génération de la librairie (build.sh, buildall.sh, et le makefile pgm_exemple_jni_lux.mak).

7.2.2.4 Remarque

Pour une description plus précise de ce qui est fait, veuillez vous reporter au code source qui est documenté.

7.2.3 Description

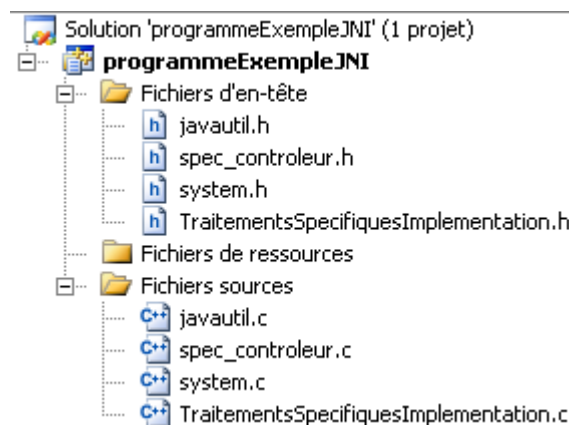


Figure 113 : Arborescence des sources C du programme d'exemple sous Visual Studio

Le programme d'exemple s'articule selon trois fichiers source C :

- `TraitementsSpecifiquesImplementation.c` (et son fichier entête `TraitementsSpecifiquesImplementation.h`) :

Couche JNI, regroupant l'implémentation des méthodes natives définies au niveau de la couche Java du programme d'exemple. Ces méthodes natives sont exclusivement dédiées au module Traitements spécifiques du programme d'exemple.

- `spec_controleur.c` (et son fichier entête `spec_controleur.h`) :

Couche « pure » C, qui réalise les traitements spécifiques sans aucune manipulation d'objets JNI.

- `javauttil.c` (et son fichier entête `javauttil.h`) :

Couche utilitaire qui comprend à la fois l'implémentation des fonctions callback pour invoquer des fonctions de la couche Java du programme d'exemple (demande de saisie du code porteur...), mais également des fonctions de conversion de types JNI vers C ou inversement, et des fonctions de log et de gestion d'exceptions.

- `system.c` (et son fichier entête `system.h`)

Couche dédiée au code spécifique plateforme (Windows, Linux...).

7.2.3.1 Points d'entrées des traitements spécifiques (Couche JNI)

Les points d'entrée des traitements spécifiques sont situés dans le fichier de source C `TraitementsSpecifiquesImplementation.c`. Ci-dessous pour chaque traitement spécifique vous trouverez la méthode appelée depuis la couche Java.

Etat de connexion carte	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_etatConnexionCarte</code>
Assistant d'authentification	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_assistantAuthentification</code>
Authentification après retrait carte	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_authentificationAprèsRetraitCarte</code>
Authentification après retrait lecteur	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_authentificationAprèsRetraitLecteur</code>
Etat de saisie du code porteur	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_etatSaisieCodePorteur</code>
Recyclage du code porteur	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_recyclageCodePorteur</code>
Lecteur d'objet de données applicatives (jeton)	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_lectureObjetDonneesApplicatives</code>
Modification d'objet de données applicatives (jeton)	<code>Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_modificationObjetDonneesApplicatives</code>

Deux fonctions natives supplémentaires ont été ajoutées, afin de réaliser le chargement et l'initialisation de la librairie PKCS#11. Il s'agit respectivement des fonctions :

```
Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_c  
onnecte
```

```
Java_fr_asip_cps3_exemple_modele_jni_TraitementsSpecifiquesImplementation_i  
nitialiseLibrairie
```

Chacune des méthodes du tableau ci-dessus, initialise un contexte spécifique, fait appel à une méthode de la couche « pure » C, traite le code retour, et, si besoin, lève une exception. Vous trouverez au paragraphe suivant les fonctions de la couche « pure » C qui sont invoquées selon les traitements spécifiques.

7.2.3.2 Traitements spécifiques et fonctions annexes (Couche « pure » C)

■ Traitements spécifiques

Etat de connexion carte	TS_etatConnexionCarte
Assistant d'authentification	TS_assistantAuthentification
Authentification après retrait carte	TS_authentificationApresRetrait <i>La fonction est invoquée avec un type de retrait en paramètre. Dans ce cas le type de retrait correspond au retrait carte.</i>
Authentification après retrait lecteur	TS_authentificationApresRetrait <i>La fonction est invoquée avec un type de retrait en paramètre. Dans ce cas le type de retrait correspond au retrait lecteur.</i>
Etat de saisie du code porteur	TS_etatSaisieCodePorteur
Recyclage du code porteur	TS_recyclageCodePorteur
Lecteur d'objet de données applicatives (jeton)	TS_lectureObjetDonneesApplicatives
Modification d'objet de données applicatives (jeton)	TS_modificationObjetDonneesApplicatives

Certaines de ces fonctions partagent du code, au travers de fonctions annexes, afin d'améliorer la lisibilité et éviter la duplication. Il est possible de distinguer les fonctions annexes des fonctions de traitements spécifiques. En effet, ces dernières sont préfixées de TS_ (TS pour Traitement Spécifique). Vous trouverez ci-dessous la liste des fonctions annexes accompagnées de leur description.

■ Fonctions annexes

authentification	Détermine le statut d'authentification de l'utilisateur
authentificationNecessaire	Détermine si une authentification est nécessaire
chercheLecteurContenantCarte	Scanne tous les slots avec carte pour trouver la carte avec le numéro de série spécifié en argument
identifieTypeActionRetrait	Détermine le type d'action retrait Lecteur/Carte qui est survenue
infoConformiteCode	Mémoire dans le contexte, sur la base des informations carte, la (non)conformité du code
infoNbEssaisCode	Mémoire dans le contexte, sur la base des informations carte, le nombre d'essais restants avant blocage
instantane	Scanne tous les slots avec carte pour créer un instantané mémorisant les associations

	(identifiantSlot/numeroSerieCarte)
litAttributValeurObjet	Lit l'attribut valeur d'un objet de données applicatives (jeton)
login	Appelle C_Login et traite le code retour
ouvreSession	Ouvre une session sur la carte mémorisée dans le contexte
ouvreSessionSurPremiereCarte	Ouvre une session sur la première carte supportée rencontrée
recuperationIdentifiantObjetDonneesApplicatives	Récupère l'identifiant d'un objet de données applicatives (jeton)
recuperePremiereCarte	Scanne tous les slots et mémorise dans le contexte la première carte supportée rencontrée
reinitialiseIdentifiantsEtInfosContexteTraitementsSpecifiques	Réinitialise les identifiants (slot/session) et informations (carte/session) du contexte de réalisation des appels spécifiques

7.2.4 Génération

7.2.4.1 Windows

Il est possible de générer le fichier .dll depuis Microsoft Visual Studio de la façon suivante :

Générer > Générer la solution.

Le fichier pgm_exemple_jni_w32.dll est alors généré dans le répertoire associé à la cible définie. Exemple /debug dans la figure ci-dessous.

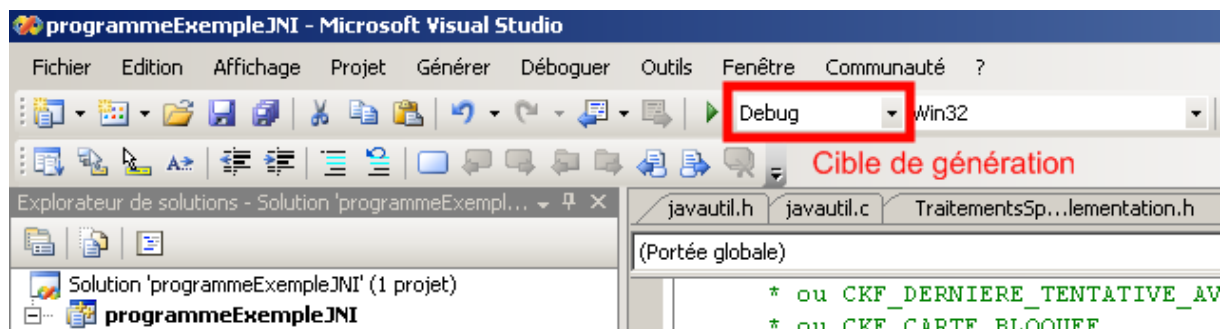


Figure 114 : Cible de génération de la librairie native du programme d'exemple

Le fichier généré doit être positionné à la racine du projet Java du programme d'exemple pour être utilisé et pris en compte.

7.2.4.2 Linux

Il est possible de générer le fichier .so grâce aux fichiers de génération (build.sh, buildall.sh, et le makefile pgm_exemple_jni_lux.mak) mis à disposition dans le sous répertoire /linux situé dans le répertoire des sources C du programme d'exemple.

Dans une fenêtre de Terminal, se positionner dans le répertoire /linux puis exécuter la commande :

- ./buildall.sh (génère toutes les versions Release et Debug)
- ou

- `. build.sh d clean` (génère uniquement la version Debug).
- ou
- `. build.sh r clean` (génère uniquement la version Release).

Le fichier `libpgm_exemple_jni_lux.so` est alors généré dans le répertoire associé à la cible définie (`/debug` ou `/release` ou les 2).

Le fichier généré est à positionner à la racine du projet Java du programme d'exemple pour être utilisé et pris en compte.

Table des illustrations

Figure 1 : Erreur de lancement et chargement explicite de librairie PKCS#11	9
Figure 2 : Menu Fonctions PKCS#11	10
Figure 3 : Le menu est verrouillé	10
Figure 4 : Le menu est déverrouillé	10
Figure 5 : Chargement d'une librairie PKCS#11	11
Figure 6 : Les deux issues possibles au chargement d'une librairie PKCS#11	11
Figure 7 : Barre d'état mentionnant la librairie PKCS#11 en cours d'utilisation	11
Figure 8 : Accès au menu Initialiser la librairie	12
Figure 9 : Ecran de résultat d'opération: Initialiser la librairie	12
Figure 10 : Accès à la fonction de terminaison de la librairie	13
Figure 11 : Ecran de résultat d'opération : Terminer la librairie	13
Figure 12 : Accès à la fonction d'obtention des informations sur la librairie	13
Figure 13 : Ecran de résultat d'opération : Obtenir des informations sur la librairie	14
Figure 14 : Accès à la fonction d'énumération des slots	14
Figure 15 : Ecran de résultat d'opération : Liste tous les slots / Liste les slots avec carte	15
Figure 16 : Accès à la fonction d'obtention des informations sur un slot	16
Figure 17 : Ecran de résultat d'opération : Obtenir des informations sur le slot	16
Figure 18 : Accès à la fonction d'obtention des informations sur une carte	17
Figure 19 : Ecran de résultat d'opération : Obtenir des informations sur la carte	17
Figure 20 : Accès à la fonction d'attente d'un évènement slot	19
Figure 21 : Ecran de résultat d'opération : Attente d'un évènement slot	20
Figure 22 : Accès à la fonction d'Obtention des algorithmes supportés par la carte	20
Figure 23 : Ecran de résultat d'opération : Obtenir la liste des algorithmes supportés par la carte	21
Figure 24 : Accès à la fonction d'obtention des informations sur un algorithme supporté par une carte	22
Figure 25 : Ecran de résultat d'opération : Obtenir des informations sur un algorithme	22
Figure 26 : Accès à la fonction d'initialisation du code porteur de la carte	24
Figure 27 : Ecran de saisie : Initialisation du code porteur de la carte	24
Figure 28 : Ecran de résultat d'opération : Initialisation du code porteur de la carte	24
Figure 29 : Accès à la fonction de modification du code porteur de la carte	25
Figure 30 : Ecran de saisie : Modification du code porteur de la carte	25
Figure 31 : Ecran de résultat d'opération : Modification du code porteur de la carte	25
Figure 32 : Accès à la fonction d'ouverture de session Lecture seule	26
Figure 33 : Accès à la fonction d'ouverture de session en Lecture/Ecriture	26
Figure 34 : Ecran de résultat d'opération : Ouvrir une session en lecture seule	26
Figure 35 : Accès à la fonctionnalité de bascule de sessions	27
Figure 36 : Accès à la fonction d'obtention d'informations sur la session courante	27
Figure 37 : Ecran de résultat d'opération : Obtenir des informations sur la session	28

Figure 38 : Accès à la fonction de fermeture d'une session active	29
Figure 39 : Ecran de résultat d'opération : Fermer la session	29
Figure 40 : Accès à la fonction de fermeture de toutes les sessions actives sur une carte	29
Figure 41 : Ecran de résultat d'opération : Fermer toutes les sessions actives.....	30
Figure 42 : Accès à la fonction d'authentification	30
Figure 43 : Ecran de saisie : Authentification Security Officer (SO).....	31
Figure 44 : Ecran de résultat d'opération : S'authentifier	31
Figure 45 : Mise à jour du libellé de la session après authentification.....	31
Figure 46 : Accès à la fonction de « désauthentification »	32
Figure 47 : Ecran de résultat d'opération : « Désauthentification » d'un utilisateur	32
Figure 48 : Accès à la fonction de création d'objet	33
Figure 49 : Ecran de saisie : Saisie d'un objet de données applicatives pour création	33
Figure 50 : Ecran de résultat d'opération : Création d'un objet de données applicatives	34
Figure 51 : Accès à la fonction de copie d'un objet de données applicatives.....	35
Figure 52 : Ecran de résultat d'opération : Copie d'un objet de données applicatives	35
Figure 53 : Accès à la fonction de suppression d'un objet de données applicatives	36
Figure 54 : Ecran de résultat d'opération : Suppression d'un objet de données applicatives.....	36
Figure 55 : Accès à la fonction d'obtention de la taille d'un objet de données applicatives.....	37
Figure 56 : Ecran de résultat d'opération : Obtention de la taille d'un objet de données applicatives..	37
Figure 57 : Accès à la fonction d'affichage de l'attribut d'un objet de données applicatives.....	38
Figure 58 : Ecran de résultat d'opération : Affichage de l'attribut d'un objet de données applicatives .	38
Figure 59 : Accès à la fonction de modification de l'attribut d'un objet de données applicatives.....	39
Figure 60 : Ecran de saisie : Saisie de la modification d'un attribut d'un objet de données applicatives	39
Figure 61 : Ecran de résultat d'opération : Modification de l'attribut d'un objet de données applicatives	40
Figure 62 : Accès à la fonction de recherche d'objets.....	41
Figure 63 : Ecran de résultat d'opération : Recherche d'objet(s) de type certificat.....	41
Figure 64 Ecran de saisie : Saisie du filtre de recherche pour les objets de données applicatives à rechercher.....	42
Figure 65 : Accès à la fonction de génération de l'empreinte d'un message court.....	43
Figure 66 : Ecran de saisie : Saisie d'un message court	43
Figure 67 : Ecran de résultat d'opération : Génération de l'empreinte d'un message court	44
Figure 68 : Accès à la fonction de génération de l'empreinte d'un fichier.....	45
Figure 69 : Ecran de saisie : Saisie d'un fichier	45
Figure 70 : Ecran de résultat d'opération : Génération de l'empreinte d'un fichier	46
Figure 71 : Accès à la fonction de signature d'un message court.....	47
Figure 72 : Ecran de saisie : Saisie d'un message court	48
Figure 73 : Ecran de résultat d'opération : Signature d'un message court	48
Figure 74 : Accès à la fonction de signature d'un fichier.....	50
Figure 75 : Ecran de résultat d'opération : Signature d'un fichier	50

Figure 76 : Accès à la vérification de la signature du message court.....	52
Figure 77 : Ecran de saisie : Saisie pour la vérification de signature d'un message court.....	53
Figure 78 : Ecran de résultat d'opération : Vérification de signature d'un message court.....	53
Figure 79 : Accès à la vérification de la signature du fichier.....	54
Figure 80 : Ecran de saisie : Saisie pour la vérification de la signature d'un fichier	55
Figure 81 : Ecran de résultat d'opération : Vérification de la signature d'un fichier	55
Figure 82 : Accès à la fonction de génération de nombre pseudo-aléatoire normale.....	56
Figure 83 : Ecran de résultat d'opération : Génération normale de nombre pseudo-aléatoire.....	56
Figure 84 : Accès à la fonction de génération de nombre pseudo-aléatoire couplée à une graine.....	57
Figure 85 : Ecran de saisie : Saisie d'une graine (seed).....	57
Figure 86 : Ecran de résultat d'opération : Génération avec graine d'un nombre pseudo-aléatoire.....	58
Figure 87 : Accès au traitement spécifique : Statut d'authentification.....	59
Figure 88 : Ecran de résultat de traitement spécifique : Statut d'authentification	59
Figure 89 : Accès au traitement spécifique : Assistant d'authentification	60
Figure 90 : Ecrans et boîtes de dialogue du traitement spécifique : Assistant d'authentification	60
Figure 91 : Ecran de résultat de traitement spécifique : Assistant d'authentification.....	61
Figure 92 : Accès au traitement spécifique : Authentification après retrait carte	61
Figure 93 : Ecran de résultat de traitement spécifique : Echec de l'authentification après retrait de carte.....	62
Figure 94 : Accès au traitement spécifique : Authentification après retrait lecteur	62
Figure 95 : Ecran de résultat de traitement spécifique : Echec de l'authentification après retrait de lecteur	63
Figure 96 : Accès au traitement spécifique : Etat de saisie du code porteur	64
Figure 97 : Boîte de dialogue pour le traitement spécifique : Etat de saisie du code porteur.....	64
Figure 98 : Boîtes de dialogue de saisie pour traitement spécifique : Etat de saisie du code porteur ..	65
Figure 99 Ecran de résultat de traitement spécifique : Etat de saisie du code porteur	65
Figure 100 : Accès au traitement spécifique : Recyclage de la carte avec le code de déblocage	66
Figure 101 : Boîte de dialogue de saisie pour traitement spécifique : Recyclage de la carte avec le code de déblocage	66
Figure 102 : Ecran de résultat de traitement spécifique : Recyclage de la carte avec le code de déblocage	66
Figure 103 : Accès au traitement spécifique : Lecture d'un jeton d'établissement	67
Figure 104 : Ecran de résultat de traitement spécifique : Lecture d'un jeton d'établissement.....	67
Figure 105 : Accès au traitement spécifique : Modification d'un jeton d'établissement	68
Figure 106 : Boîte de dialogue de saisie pour traitement spécifique : Modification d'un jeton d'établissement.....	68
Figure 107 : Ecran de résultat de traitement spécifique : Modification d'un jeton d'établissement	69
Figure 108 : Arborescence des sources Java du programme d'exemple	71
Figure 109 : Arborescence des sources Java des objets manipulés.....	72
Figure 110 : Arborescence des sources Java du module Fonctions PKCS#11	72
Figure 111 : Arborescence des sources Java du module Traitements spécifiques.....	73
Figure 112 : Export des sources Java sous forme de fichier JAR	74

Figure 113 : Arborescence des sources C du programme d'exemple sous Visual Studio	75
Figure 114 : Cible de génération de la librairie native du programme d'exemple	79