PC/SC

Documentation Partiel de l'API

www.springcard.com

Document Classification: **Public**

Status: **N/A**

Date: **23/08/18**

Document identification

Reference - Version	PMDZ061 - BA
Category	Developer's Guide
Family	PC/SC Couplers (all)
Original file name	[PMDZ061-BA] PCSC partial documentation of the API FR.odt
Date	23/08/18
Classification	Public
Status	N/A
Keywords	PC/SC, couplers, smartcards, winscard
Description	This document is a quick reference of the most useful functions provided by the PC/SC API.

Revision history

Version	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	17/01/08	JDA				Early draft
AB	30/05/08	JDA				Almost complete
AC	05/09/08	JDA				New SpringCard template
AD	01/04/09	ECL	LTX	LTX		Adding SCardGetGCRFamily
AE	24/09/09	ECL				Adding SCardPresent and SCardSetReaderPower
AF	30/03/16	HTH				
BA	09/11/16	JDA				New SpringCard template

All information in this document is subject to the standard disclaimer stated on last page

Table of content

All information in this document is subject to the standard disclaimer stated on last page

All information in this document is subject to the standard disclaimer stated on last page

All information in this document is subject to the standard disclaimer stated on last page

All information in this document is subject to the standard disclaimer stated on last page

1. Introduction

1.1. Résumé

PC/SC est le standard pour interfacer les ordinateurs personels avec les cartes à puce (et les lecteurs de cartes à puce évidemment). Cet API de pointe standardisée permet au développeur de se concentrer sur la carte à puce elle-même, sans avoir à gérer les divers – et propriétaires – aspects de chaque lecteur de carte à puce.
PC/SC est disponible sur les ordinateurs Windows (stack PC/SC intégré dans OS disponible via la librairie `winscard.dll`). Grâce au projet open-source PCSC-Lite, le PC/SC c'est implanté sur de nombreuses plateformes Unix (comme GNU/Linux et Mac OS X).
Sur la plateforme PocketPC (construction légère du Windows CE qui n'a pas de stack PC/SC), le "SpringCard API" de SpringCard (la libraire `springcard.dll` est disponible dans notre SDK) fournit les mêmes possibilités de manière pratique.

Ce document est une courte référence aux fonctions les plus utiles de l'API PC/SC. Attention: il peut exister quelques différences d'une implémentation à une autre. Toujours ce référer à la documentation du stack PC/SC avec lequel vous travaillez.

1.2. Produits supportés

Ce document s'applique à tous les coupleurs PC/SC SpringCard.

1.3. Public

Ce document est créé pour les développeurs d'applications. Il suppose que le lecteur à des connaissances expertes en développement informatique.

1.4. Support et mises à jour

Les documents liés (fiche technique du produit, notes d'applications, échantillon logiciel, HOWTO, FAQ..) sont disponibles sur le site de SpringCard :

https://www.springcard.com

Les versions mises à jour de ce document seront mises en ligne sur le site dès qu'elles seront disponibles.
Pour des demandes au support technique, merci d'utiliser ce formulaire

https://www.springcard.com/en/contact?request=support

All information in this document is subject to the standard disclaimer stated on last page

1.5. Liens utiles

- La documentation de référence Microsoft PC/SC est incluse dans la plupart des systèmes d'aide Visual Studio, et est disponible en ligne à http://msdn.microsoft.com . Entrer les mots clés “winscard” ou “SCardTransmit” dans la boîte de recherche.
- Projet MUSCLE PCSC-Lite : http://pcsclite.alioth.debian.org/musclecard.com/info.html (lien direct au stack PC/SC : http://pcsclite.alioth.debian.org)
- PC/SC workgroup : http://www.pcscworkgroup.com

1.6. Remerciements

La pluparts des explications et exemples de ce documents ont été inspirés par la documentation PCSC-Lite. Le document original est ici :

http://pcsclite.alioth.debian.org/api/group__API.html

Nous remercions les auteurs de leur bon travail.

All information in this document is subject to the standard disclaimer stated on last page

2. Liste des fonctions PC/SC

2.1. SCardEstablishContext

La fonction **SCardEstablishContext** instancie un contexte pour l'application dans le manager de ressource PC/SC. Elle doit être la première fonction appellée dans une application PC/SC. Dans une application multifilière chaque filière doit instancier son propre contexte.

2.1.1. Synopsis

```
LONG SCardEstablishContext(DWORD dwScope,
                           LPCVOID pvReserved1,
                           LPCVOID pvReserved2,
                           LPSCARDCONTEXT phContext);
```

2.1.2. Paramètres

Name	*Way*	*Description*
dwScope	in	See below
pvReserved1	in	Must be NULL.
pvReserved2	in	Must be NULL.
phContext	out	Pointer to receive the handle to the PC/SC resource manager

dwScope dit à la librairie PC/SC si votre application veut accéder aux coupleurs disponibles pour l'utilisateur actuel (SCARD_SCOPE_USER) ou si elle doit agir comme un service système (SCARD_SCOPE_SYSTEM). Les deux valeurs sont définies dans 3.2.

Sur PCSC-LiteOn PCSC-Lite, **dwScope** doit être SCARD_SCOPE_SYSTEM.

2.1.3. Valeur de retour

Cette fonction répond SCARD_S_SUCCESS lors d'un succès, ou l'un des codes d'erreur listé dans 3.1.

All information in this document is subject to the standard disclaimer stated on last page

2.2. SCardReleaseContext

La fonction **SCardReleaseContext** détruit le contexte de l'application dans le manager de ressource PC/SC et libère toutes les ressources associées. Cela doit être la dernière fonction appelée dans une application PC/SC. Dans une application multifilière, chaque filière ayant son propre contexte PC/SC doit appeler cette fonction lorsqu'elle termine.

2.2.1. Synopsis

```
LONG SCardReleaseContext(SCARDCONTEXT hContext);
```

2.2.2. Paramètres

Name	*Way*	*Description*
`hContext`	in	Handle to the PC/SC resource manager as obtained by a successful call to **SCardEstablishContext**

2.2.3. Valeur de retour

La fonction renvoie `SCARD_S_SUCCESS` lors d'un succès, ou l'un des codes d'erreur listés en 3.1.

All information in this document is subject to the standard disclaimer stated on last page

2.3. SCardIsValidContext

Le ScardIsValidContext détermine si le contexte de gestion d'une carte à puce est toujours valide. Après qu'un contexte de gestion de carte à puce est été créé par SCardEstablishContext, il peut devenir invalide sur le manager des ressources a été éteint.

2.3.1. Synopsis

```
LONG SCardIsValidContext(SCARDCONTEXT hContext);
```

2.3.2. Paramètres

Name	*Way*	*Description*
`hContext`	in	Handle to the PC/SC resource manager as obtained by a successful call to SCardEstablishContext

2.3.3. Valeur de retour

La fonction répond `SCARD_S_SUCCESS` si le contexte est toujours valide, ou `SCARD_E_INVALID_HANDLE` s'il ne l'est pas. Les codes d'erreur sont listés en 3.1.

All information in this document is subject to the standard disclaimer stated on last page

2.4. SCardListReaders

La fonction **SCardListReaders** renvoie une liste des lecteurs actuellement disponible dans le système, ils pourront éventuellement être filtrés par un ensemble de groupes de lecteurs nommés.

2.4.1. Synopsis

```
LONG SCardListReaders(SCARDCONTEXT hContext,
                      LPCSTR mszGroups,
                      LPSTR mszReaders,
                      LPDWORD pcchReaders);
```

2.4.2. Paramètres

Name	*Way*	*Description*
`hContext`	in	Handle to the PC/SC resource manager as obtained by a successful call to **SCardEstablishContext**
`mszGroups`	in	List of groups to list readers
`mszReaders`	out	Multi-string to receive the list of readers
`pcchReaders`	in	Max. length of **mszReaders**
	out	Actual length of **mszReaders** including all '\0's characters

2.4.3. Valeur de retour

La fonction renvoie `SCARD_S_SUCCESS` lors d'un succès, ou l'un des codes d'erreur listés dans 3.1.

2.4.4. Détails

Cette fonctione renvoie la liste des lecteurs actuellement disponibles dans le système. **mszReaders** est l'indicateur du fil d'un caractère qui est alloué par l'application. Si l'application envoie **szReaders** comme `NULL` alors cette fonction renvoie la taille de la mémoire tampon nécessaire pour répartir dans **pcchReaders**.

Si la valuer indiquée par **pcchReaders** est spécifiée comme `SCARD_AUTOALLOCATE`, alors **mszReaders** est considérée comme indicateur à un indicateur et reçoit l'adresse d'un bloc mémoire contenant le fil caractère, alloué par l'API PC/SC. Le bloc de mémoire doit être désaffecté par la suite par l'appel de l'application en utilisant **SCardFreeMemory**.

Lors d'un succès, **mszReaders** est multi-fil et séparé par un caractère nul ('\0') et terminé par un caractère double nul (e.g. "`Reader A\0Reader B\0\0`").

Attention 1
Noter que dans **winscard.dll** il y a actuellement deux fonctions:
- **SCardListReadersA** où tous les STR sont gérées comme des fils ASCII (`char *`)

All information in this document is subject to the standard disclaimer stated on last page

- **SCardListReadersW** où tous les STR sont gérés comme des fils UTF16 (`wchar_t *`)
winscard.h cartographie **SCardListReaders** chaque foncion dépend de la compilation de `_UNICODE`.

Attention 2

Sur Windows 10, le nombre de coupleurs qui peuvent être supportés par le système au même moment est limité à 10. **SCardListReader** peuvent agir de manière incorrecte si vous avez plus de 10 coupleurs PC/SC liés à l'ordinateur.

Attention 3

PCSC-Lite ne supporte pas le `SCARD_AUTOALLOCATE` pour **pcchReaders** et s'attend à ce que le **mszGroups** soit `NULL`.

All information in this document is subject to the standard disclaimer stated on last page

2.5. SCardGetStatusChange

La fonction **SCardGetStatusChange** bloque l'exécution jusqu'à ce que la disponibilité des cartes dans un ensemble spécifique de lecteurs change.

1. SpringCard API spécifique

Merci d'utiliser Erreur : source de la référence non trouvée pour les API SpringCard

2. Synopsis

```
LONG SCardGetStatusChange(SCARDCONTEXT hContext,
                          DWORD dwTimeout,
                          LPSCARD_READERSTATE rgReaderStates,
                          DWORD cReaders);
```

3. Paramètres

hContext	in	Connection context to the PC/SC Resource Manager
dwTimeout	in	Maximum waiting time (in miliseconds) for status change, zero (or INFINITE) for infinite.
rgReaderStates	inout	Array of SCARD_READERSTATE structures (one for each reader)
cReader	in	Number of SCARD_READERSTATE structures in rgReaderStates

4. Note d'usage

L'interlocuteur fournit une liste de lecteurs qui devront être surveillés par un ensemble SCARD_READERSTATE et la durée maximum (en millisecondes) qu'il est prêt à attendre pour qu'une action se produise sur l'un des lecteurs listé.
La fonction répond lorsqu'il y a un changement dans la disponibilité (ex: une carte est insérée ou retirée), en ayant rempli les membres dwEventState du rgReaderStates de manière appropriée.
Si rgReaderStates est NULL, et cReader est égal à 0, alors la fonction bloquera jusqu'à ce que le lecteur soit ajouté au système (nécessite un support de branchement à chaud).

SCardGetStatusChange utilise la valeur fournie par l'utilisateur dans les membres dwCurrentState de l'ensemble rgReaderStates SCARD_READERSTATE comme définition de l'état actuel des lecteurs. Fournir une valeur invalide empêchera le ScardGetStatusChange de fonctionner correctement.

5. SCARD_READERSTATE structure

```
typedef struct {
    LPCSTR szReader;       /* Reader name */
    LPVOID pvUserData;     /* User defined data */
```

All information in this document is subject to the standard disclaimer stated on last page

```
    DWORD dwCurrentState; /* Current state of reader */
    DWORD dwEventState;   /* New state after a state change */
    DWORD cbAtr;          /* ATR length */
    BYTE rgbAtr[MAX_ATR_SIZE]; /* ATR of the card */
} SCARD_READERSTATE;

typedef SCARD_READERSTATE *PSCARD_READERSTATE,
                          **LPSCARD_READERSTATE;
```

6. Valeurs de dwCurrentState et dwEventState

- SCARD_STATE_UNAWARE :
  L'application ne connaît pas l'état actuel et voudrait le connaître. L'utilisation de cette valeur résulte dans un retour immédiat aux services de contrôle de transition d'état. Ceci est représenté par tous les bits à zéro.
- SCARD_STATE_IGNORE :
  Le lecteur doit être ignoré.
- SCARD_STATE_CHANGED :
  Il y a une différence entre un état cru par l'application et l'état connu par le manager de ressources. Lorsqu'un bit est fixé l'application peut supposer qu'un changement d'état significatif s'est produit sur ce lecteur.
- SCARD_STATE_UNKNOWN :
  Le nom de lecteur donnée n'est pas reconnu par le manager de ressources. Si le bit est fixé, alors `SCARD_STATE_CHANGED` et `SCARD_STATE_IGNORE` seront également fixés.
- SCARD_STATE_UNAVAILABLE :
  Le statut actuel de ce lecteur n'est pas disponible. Si le bit est fixé, alors les bits suivants seront libres.
- SCARD_STATE_EMPTY :
  Il n'y a pas de carte dans le lecteur. Si ce bit est fixé, les bits suivants seront libres.
- SCARD_STATE_EXCLUSIVE :
  La carte dans le lecteur est allouée à l'utilisation exclusive par une autre application. Si le bit est fixé, `SCARD_STATE_PRESENT` sera aussi fixé.
- SCARD_STATE_INUSE :
  La carte dans le lecteur est utilisée par une ou plusieurs autres applications, mais peut être connectée en mode partagé. Si le bit est fixé, SCARD_STATE_PRESENT sera également fixé.
- SCARD_STATE_MUTE :
  Il y a une carte muette dans le lecteur.

All information in this document is subject to the standard disclaimer stated on last page

2.6. SCardConnect

Le **SCardConnect** crée une connexion entre l'application qui appelle et la carte à puce insérée dans le coupleur PC/SC. Dans certaines situations, il est également possible d'ouvrir une connexion directe avec un coupleur PC/SC où aucune carte à puce n'est insérée.

2.6.1. Synopsis

```
LONG SCardConnect(SCARDCONTEXT hContext,
                  LPCSTR szReader,
                  DWORD dwShareMode,
                  DWORD dwPreferredProtocols,
                  LPSCARDHANDLE phCard,
                  LPDWORD pdwActiveProtocol);
```

2.6.2. Paramètres

Name	*Way*	*Description*
`hContext`	in	Handle to the PC/SC resource manager as obtained by a successful call to **SCardEstablishContext**
`szReader`	in	List of groups to list readers
`dwShareMode`	out	Whether the card shall be shared or reserved for exclusive use. Values are listed in 3.3
`dwPreferredProtocols`	in	The expected card protocol(s). Values are listed in 3.4
`phCard`	out	Pointer to receive the handle to the PC/SC resource manager
`pdwActiveProtocol`	out	Pointer to receive the actual protocol that has been activated (on success). Values are listed in 3.4. On some versions of Windows, setting this parameter to `NULL` triggers an error.

2.6.3. Valeur de retour

La fonction répond `SCARD_S_SUCCESS` lors d'un succès, ou l'un des codes d'erreur listés dans 3.1.

2.6.4. Détails

Attention

Noter que dans **winscard.dll** il y a deux fonctions :
- **SCardListConnectA** où **szReader** est géré comme un fil ASCII (`char *`)
- **SCardListConnectW** où **szReader** est géré comme un fil UTF16 (`wchar_t *`)

winscard.h cartographie **SCardListConnect** chaque fonction dépend de la compilation de `_UNICODE`.

All information in this document is subject to the standard disclaimer stated on last page

2.6.5. Ouvrir une connexion avec une carte

2.6.5.1. Sélection du protocole

Pour ouvrir une connexion avec une carte à puce, l'application doit fixer **dwPreferredProtocols** à l'une des valeurs suivantes:

- `SCARD_PROTOCOL_T0` si l'application s'attend à ce que la carte utilise le protocole ISO 7816-3 T=0,
- `SCARD_PROTOCOL_T1` si l'application s'attend à ce que la carte utilise le protocole ISO 7816-3 T=1,
- `SCARD_PROTOCOL_T0|SCARD_PROTOCOL_T1` si l'application s'attend à ce que la carte utilise l'un ou l'autre des protocoles.

Le protocole qui a été sélectionné par le coupleur et la carte seront retournées dans **pdwActiveProtocol**.

Pour les cartes sans-contact, le coupleur sans-contact émule le protocole T=1, donc l'application doit fixer **dwPreferredProtocols** pour `SCARD_PROTOCOL_T1` ou `SCARD_PROTOCOL_T0|SCARD_PROTOCOL_T1`.

Certains coupleurs sont capables de faire fonctionner des cartes à contact à logique câblée (S=9, S=10 etc) en fixant **`dwPreferredProtocols`** à `SCARD_PROTOCOL_RAW`, mais cette valeur n'est pas supportée par les coupleurs SpringCard.

2.6.5.2. Mode partage

Les paramètres **dwShareMode** disent si l'application souhaite partager la carte à puce avec d'autres applications (`SCARD_SHARE_SHARED`) ou pas (`SCARD_SHARE_EXCLUSIVE`).

Accepter de partager la carte à puce introduira probablement des conflits dans les données ou l'état interne de la carte à puce, et peuvent même créer des brèches de sécurité, sauf si l'application utilise **SCardBeginTransaction** / **SCardEndTransaction** comme moyen de protéger sa carte à puce contre des accès concurrents. Fixer **dwShareMode** à `SCARD_SHARE_EXCLUSIVE` fait exactement la même chose qu'une fonction appel seule.

Noter que lorsque votre application appelle **SCardConnect**, la carte à puce peut être déjà utilisée par une autre application. Sur Windows ≥ 7, le système lui-même pourra se connecter à chaque carte à puce immédiatement après son insertion, pour déterminer si la carte à puce est liée à un intergiciel cryptographique ou à un service d'aide. Si **SCardConnect** échoue avec erreur `SCARD_E_SHARING_VIOLATION`, signifiant que la carte à puce est déjà utilisée, alors l'application devra réessayer plus tard. L'implémentation typique serait d'appler **SCardGetStatusChange** et d'attendre que la carte à puce soit de nouveau disponible.

2.6.6. Ouvrir une connexion directe à un coupleur

Il est possible d'ouvrir une connexion directe à un coupleur – même s'il n'y a pas de carte

All information in this document is subject to the standard disclaimer stated on last page

dans le coupleur – en fixant

- **dwShareMode** = `SCARD_SHARE_DIRECT`
- **dwPreferredProtocols** = `0`

Dans ce casn aucune connexion avec la carte ne sera possible (**SCardTransmit** échouera toujours) mais l'application est capable d'envoyer les commandes propriétaires au coupleur en utilisant **SCardControl**.

All information in this document is subject to the standard disclaimer stated on last page

2.7. SCardReconnect

Compatibility	
Winscard	Yes
pcsc-lite	Yes
SpringCard	Yes

La fonction SCardReconnect ré-établit une connexion existante entre l'application qui appelle et la carte à puce. Cette fonction peut être utile pour changer la gestion d'une carte d'accès direct à accès général, ou pour reconnaître et dégager une condition d'erreur qui empêche un accès avancé à la carte à puce.

1. Synopsis

```
LONG SCardReconnect(SCARDHANDLE hCard,
                    DWORD dwShareMode,
                    DWORD dwPreferredProtocols,
                    DWORD dwInitialization,
                    LPDWORD pdwActiveProtocol);
```

2. Paramètres

hCard	in	Card context as returned by SCardConnect
dwShareMode	in	Exclusive or shared connection
dwPreferredProtocols	in	Bit-mask specifying the list of acceptable card protocols
dwInitialization	in	Type of initialization that should be performed on the card.
PdwActiveProtocol	out	Actual protocol selected by the reader

1. *Valeurs pour dwShareMode*

Voir le paragraphe SCardConnect.

2. *Valeurs pour dwPreferredProtocols*

Voir le paragraphe SCardConnect.

3. *Valeurs pour dwInitialization*

- SCARD_LEAVE_CARD : Ne fait rien.
- SCARD_RESET_CARD : Garde la carte alimentée, et la réinitialise (réinitialisation à chaud).
- SCARD_UNPOWER_CARD : Éteint la carte et la réinitialise (réinitialisation à froid).

4. *Valeurs pour pdwActiveProtocol*

Voir le paragraphe SCardConnect.

3. Effets secondaires

1. Spécifiques SpringCard

Appeller la fonction SCardConnect alimente implicitement le lecteur ON SpringCard (s'il ne l'était pas).

All information in this document is subject to the standard disclaimer stated on last page

SCardDisconnect

Compatibility	
Winscard	Yes
pcsc-lite	Yes
SpringCard	Yes

La fonction **SCardDisconnect** termine une connexion ouverte précedemment entre l'application qui appelle et la carte à puce.

4. Synopsis

```
LONG SCardDisconnect(SCARDHANDLE hCard,
                     DWORD dwDisposition);
```

5. Paramètres

hCard	in	Card context as returned by SCardConnect
dwDisposition	in	Action to perform on the smartcard

1. *Valeurs pour dwDisposition*

- SCARD_LEAVE_CARD : Ne fait rien (la carte reste alimentée).
- SCARD_RESET_CARD : Réinitialise la carte (réinitialisation à chaud).
- SCARD_UNPOWER_CARD : Éteint la carte.
- SCARD_EJECT_CARD : Éjecte physiquement la carte (*si le lecteur est capable de le faire □).*

6. Effets secondaires

1. Spécifique SpringCard API

Appeler la fonction SCardDisconnect avec le paramètre dwDisposition fixé à SCARD_UNPOWER_CARD (ou SCARD_EJECT_CARD) alimentera implicitement le lecteur OFF SpringCard si la carte est la seule à être alimentée (sur un lecteur multi-ports).

All information in this document is subject to the standard disclaimer stated on last page

SCardStatus

Compatibility	
Winscard	Yes
pcsc-lite	Yes
SpringCard	Yes

La fonction **SCardStatus** renvoie le statut actuel de la carte à puce préalablement connectée avec SCardConnect.

7. Synopsis

```
LONG SCardStatus(SCARDHANDLE hCard,
                 LPSTR szReaderName,
                 LPDWORD pcchReaderLen,
                 LPDWORD pdwState,
                 LPDWORD pdwProtocol,
                 LPBYTE pbAtr,
                 LPDWORD pcbAtrLen);
```

8. Paramètres

hCard	in	Card context as returned by SCardConnect
szReaderName	out	Friendly name of the reader the card is in
pcchReaderLen	in	Length of szReaderName
	out	Actual length of szReaderName, including terminating NULL character
pdwState	out	Current state of the reader
pdwProtocol	out	Current protocol of the smartcard
pbAtr	in	Current ATR of the smartcard[1]
pcbAtrLen	in	Max. length of pbAtr
	out	Actual length of pbAtr

1.

1. Winscard et SpringCard API spécifique

Si la valeur indiquée par pcchReaderLen est spécifiée comme SCARD_AUTOALLOCATE, alors szReaderName est l'indicateur de l'indicateur, et reçoit l'adresse du bloc de mémoire contenant le fil caractère, alloué par l'API PC/SC. Ce bloc de mémoire doit être désaffecté par l'application qui appelle avec SCardFreeMemory.

2. Valeurs pour pdwState

- SCARD_ABSENT : Pas de carte dans le lecteur.
- SCARD_PRESENT : Il y a une carte dans le lecteur, mais il n'a pas été mit en position pour utilisation (*si le lecteur le supporte* □).
- SCARD_SWALLOWED : Il y a une carte dans le lecteur en position d'usage. La carte n'est pas alimentée.
- SCARD_POWERED : La carte est alimentée, mais le driver du lecteur ne connaît pas le mode de la carte.
- SCARD_NEGOTIABLE : La carte a été réinitialisée et attend la négociation PTS.

[1] ISO 7816-3 says that the ATR has 32 bytes or less, there's no need to allocate more.

All information in this document is subject to the standard disclaimer stated on last page

- SCARD_SPECIFIC : La carte a été réinitialisée et des protocoles de communications spécifiques ont été établis.

3. *Valeurs pour le pdwProtocol*

Voir le paragraphe SCardConnect. La valeur est correcte seulement si la valeur pdwState est SCARD_SPECIFIC.

SCardSetTimeout

Compatibility	
Winscard	NO
pcsc-lite	Dummy
SpringCard	Yes

La fonction **SCardSetTimeout** définit la valeur du dépassement de délai pour SCardTransmit. C'est utile par exemple lorsque faire fonctionner une carte prend longtemps durant lequel le lecteur peut apparaître comme “gelé”.
Noter que cette fonction n'a jamais été supportée par Winscard, et a été dépréciée par PCSC-Lite. Nous continuons de le supporter dans SpringCard API.

9. Synopsis

```
LONG SCardSetTimeout(SCARDCONTEXT hContext,
                     DWORD dwTimeout);
```

10. Paramètres

hCard	in	Card context as returned by SCardConnect
dwTimeout	In	New value for SCardTransmit timeout (in seconds)

All information in this document is subject to the standard disclaimer stated on last page

SCardTransmit

Compatibility	
Winscard	Yes
pcsc-lite	Yes
SpringCard	Yes

La fonction **SCardTransmit** envoie une commande APDU à la carte à puce et retrouve sa réponse.

11. Synopsis

```
LONG SCardTransmit(SCARDHANDLE hCard,
                   LPCSCARD_IO_REQUEST pioSendPci,
                   LPCBYTE pbSendBuffer,
                   DWORD cbSendLength,
                   LPSCARD_IO_REQUEST pioRecvPci,
                   LPBYTE pbRecvBuffer,
                   LPDWORD pcbRecvLength);
```

12. Paramètres

hCard	in	Card context as returned by SCardConnect
pioSendPci	in/out	See below
pbSendBuffer	in	APDU to send to the card
cbSendLength	in	Length of the APDU
pioRecvPci	in/out	See below
pbRecvBuffer	out	*Response from the card*
pcbRecvLength	in	*Max. length of pbRecvBuffer*
	out	*Actual length of response in pbRecvBuffer*

1. Valeurs pour pioSendPci et pioRecvPci

Les deux paramètres sont des indicateurs de la structure disant au driver quel protocole doit être utilisé pour envoyer l'APDU et recevoir une réponse de la carte.
Dans la plupart des cas, vous pourrez utiliser l'une des valeurs globales pré-définies pour les deux paramètres :

- SCARD_PCI_T0 si le protocole actuel de la carte T=0.
- SCARD_PCI_T1 si le protocole actuel de la carte T=1.

Autrement vous pouvez fixer les deux valeurs à NULL et laisser l'API PC/SC vérifier quel protocole est actuellement utilisé.

1. Spécifique SpringCard API

Les paramètres pioSendPci et pioRecvPci sont ignorés. Les définitions simplifiées de SCARD_PCI_T0 et SCARD_PCI_T1 sont fournies afin de faciliter la portabilité du code.

All information in this document is subject to the standard disclaimer stated on last page

SCardControl

Compatibility	
Winscard	Yes
pcsc-lite	Yes
SpringCard	Limited

La fonction **SCardControl** vous donne un contrôle direct sur le lecteur (même lorsqu'il n'y a pas de carte à l'intérieur).

13. Synopsis

```
LONG SCardControl(SCARDHANDLE hCard,
                  DWORD dwControlCode,
                  LPCVOID pbSendBuffer,
                  DWORD cbSendLength,
                  LPVOID pbRecvBuffer,
                  DWORD pcbRecvLength,
                  LPDWORD lpBytesReturned);
```

14. Paramètres

hCard	in	Card context as returned by SCardConnect
dwControlCode	in	*Control code for the operation to be performed*
pbSendBuffer	in	Data required to perform the operation. This parameter can be NULL if the operation takes no data.
cbSendLength	in	Length of the data
pbRecvBuffer	out	*Response from the reader. This parameter can be NULL if the operation returns nothing.*
pcbRecvLength	in	*Max. length of pbRecvBuffer*
lpBytesReturned	out	*Actual length of response in pbRecvBuffer*

1. Remarque 1

La liste de toutes les commandes disponibles, ainsi que leurs codes de contrôle et propriétaires (ex: spécifique vendeur et le plus souvent spécifique produit).
Toujours ce référer à la documentation fournie par le vendeur du lecteur pour connaître la liste des opérations supportées (ainsi que les spécifications sur les paramètres et les réponses).

1. Spécifique SpringCard API

Le paramètre dwControlCode est ignoré, et pbSendBuffer est transférée “as is” à la puce GemCore qui est à l'intérieur des lecteurs SpringCard-CF ou SpringCard-WAP.
Merci de vous référer à la documentation Gemalto GemCore dans ce cas.

2.

3. Remarque 2

Selon l'implémentation, appeler le SCardControl peut nécessiter que la connexion hCard a été créée avec soit la valeur SCARD_SHARE_DIRECT ou la valeur SCARD_SHARE_EXCLUSIVE pour le paramètre SCardConnect ou SCardReconnect dwShareMode.

All information in this document is subject to the standard disclaimer stated on last page

All information in this document is subject to the standard disclaimer stated on last page

3. Liste des constantes PC/SC

3.1. Codes erreur

Value	Meaning
SCARD_S_SUCCESS	No error was encountered.
SCARD_E_BAD_SEEK	An error occurred in setting the smart card file object pointer.
SCARD_E_CANCELLED	The action was canceled by an SCardCancel request.
SCARD_E_CANT_DISPOSE	The system could not dispose of the media in the requested manner.
SCARD_E_CARD_UNSUPPORTED	The smart card does not meet minimal requirements for support.
SCARD_E_CERTIFICATE_UNAVAILABLE	The requested certificate could not be obtained.
SCARD_E_COMM_DATA_LOST	A communications error with the smart card has been detected.
SCARD_E_DIR_NOT_FOUND	The specified directory does not exist in the smart card.
SCARD_E_DUPLICATE_READER	The reader driver did not produce a unique reader name.
SCARD_E_FILE_NOT_FOUND	The specified file does not exist in the smart card.
SCARD_E_ICC_CREATEORDER	The requested order of object creation is not supported.
SCARD_E_ICC_INSTALLATION	No primary provider can be found for the smart card.
SCARD_E_INSUFFICIENT_BUFFER	The data buffer for returned data is too small for the returned data.
SCARD_E_INVALID_ATR	An ATR string obtained from the registry is not a valid ATR string.
SCARD_E_INVALID_CHV	The supplied PIN is incorrect.
SCARD_E_INVALID_HANDLE	The supplied handle was not valid.
SCARD_E_INVALID_PARAMETER	One or more of the supplied parameters could not be properly interpreted.
SCARD_E_INVALID_TARGET	Registry startup information is missing or not valid.
SCARD_E_INVALID_VALUE	One or more of the supplied parameter values could not be properly interpreted.
SCARD_E_NO_ACCESS	Access is denied to the file.

All information in this document is subject to the standard disclaimer stated on last page

Value	Meaning
SCARD_E_NO_DIR	The supplied path does not represent a smart card directory.
SCARD_E_NO_FILE	The supplied path does not represent a smart card file.
SCARD_E_NO_MEMORY	Not enough memory available to complete this command.
SCARD_E_NO_READERS_AVAILABLE	No smart card reader is available.
SCARD_E_NO_SERVICE	The smart card resource manager is not running.
SCARD_E_NO_SMARTCARD	The operation requires a smart card, but no smart card is currently in the device.
SCARD_E_NO_SUCH_CERTIFICATE	The requested certificate does not exist.
SCARD_E_NOT_READY	The reader or card is not ready to accept commands.
SCARD_E_NOT_TRANSACTED	An attempt was made to end a nonexistent transaction.
SCARD_E_PCI_TOO_SMALL	The PCI receive buffer was too small.
SCARD_E_PROTO_MISMATCH	The requested protocols are incompatible with the protocol currently in use with the card.
SCARD_E_READER_UNAVAILABLE	The specified reader is not currently available for use.
SCARD_E_READER_UNSUPPORTED	The reader driver does not meet minimal requirements for support.
SCARD_E_SERVICE_STOPPED	The smart card resource manager has shut down.
SCARD_E_SHARING_VIOLATION	The smart card cannot be accessed because of other outstanding connections.
SCARD_E_SYSTEM_CANCELLED	The action was canceled by the system, presumably to log off or shut down.
SCARD_E_TIMEOUT	The user-specified time-out value has expired.
SCARD_E_UNEXPECTED	An unexpected card error has occurred.
SCARD_E_UNKNOWN_CARD	The specified smart card name is not recognized.
SCARD_E_UNKNOWN_READER	The specified reader name is not recognized.
SCARD_E_UNKNOWN_RES_MNG	An unrecognized error code was returned from a layered component.
SCARD_E_UNSUPPORTED_FEATURE	This smart card does not support the requested feature.
SCARD_E_WRITE_TOO_MANY	An attempt was made to write more data than would fit in the target object.
SCARD_F_COMM_ERROR	An internal communications error has been detected.
SCARD_F_INTERNAL_ERROR	An internal consistency check failed.

All information in this document is subject to the standard disclaimer stated on last page

Value	Meaning
SCARD_F_UNKNOWN_ERROR	An internal error has been detected, but the source is unknown.
SCARD_F_WAITED_TOO_LONG	An internal consistency timer has expired.
SCARD_P_SHUTDOWN	The operation has been aborted to allow the server application to exit.
SCARD_S_SUCCESS	No error was encountered.
SCARD_W_CANCELLED_BY_USER	The action was canceled by the user.
SCARD_W_CHV_BLOCKED	The card cannot be accessed because the maximum number of PIN entry attempts has been reached.
SCARD_W_EOF	The end of the smart card file has been reached.
SCARD_W_REMOVED_CARD	The smart card has been removed, so further communication is not possible.
SCARD_W_RESET_CARD	The smart card has been reset, so any shared state information is not valid.
SCARD_W_SECURITY_VIOLATION	Access was denied because of a security violation.
SCARD_W_UNPOWERED_CARD	Power has been removed from the smart card, so that further communication is not possible.
SCARD_W_UNRESPONSIVE_CARD	The smart card is not responding to a reset.
SCARD_W_UNSUPPORTED_CARD	The reader cannot communicate with the card, due to ATR string configuration conflicts.
SCARD_W_WRONG_CHV	The card cannot be accessed because the wrong PIN was presented.

3.2. Portées du contexte

3.3. Modes de partage

3.4. Protocoles

3.5. Pavillon de disposition

All information in this document is subject to the standard disclaimer stated on last page

4. Utiliser PC/SC avec SpringCard wrapper pour .NET

All information in this document is subject to the standard disclaimer stated on last page

5.

INFORMATIONS LÉGALES

Avertissement

Ce document est mis à votre disposition à titre d'information uniquement, et ne doit pas être considéré comme une offre commerciale, un contrat ou autre forme d'engagement entre SPRINGCARD et vous. Aucune des informations fournies dans ce document ne peut remplacer l'obtention de données par vos propres moyens.

Les informations fournies dans ce document peuvent concerner ou faire référence à des produits et/ou des services qui ne sont pas disponibles dans votre pays.

Le présent document est fourni "tel quel" et ce sans aucune garantie, expresse ou implicite, dans les limites prévues par les lois applicables. Même si SPRINGCARD fait son possible pour fournir des informations fiables, nous ne pouvons nullement garantir que ce document soit exempt d'inexactitudes, d'erreurs et/ou d'omissions, ni que son contenu soit pertinent pour votre besoin spécifique, ni qu'il soit à jour. SPRINGCARD se réserve le droit de modifier les informations qu'il contient à tout moment et sans préavis.

SPRINGCARD ne garantit aucunement les résultats obtenus par la mise en œuvre des produits décrits dans ce document. SPRINGCARD ne peut être tenue pour responsable d'aucune conséquence ni d'aucun dommage direct ou indirect, d'aucune perte de profits, interruption de service, perte de données, défaut de fonctionnement, liés à l'utilisation ou à l'impossibilité d'utiliser les produits (matériel et/ou logiciel) décrits dans ce document.

Les produits décrits dans ce document ne sont pas conçus pour être utilisés avec ou dans des équipements médicaux, ni avec ou dans tout équipement où un dysfonctionnement peut provoquer des dommages corporels. Les clients de SPRINGCARD qui choisissent d'utiliser ou de vendre les produits pour ce type d'applications le font à leurs seuls risques, et s'engagent à indemniser SPRINGCARD de tout dommage résultant d'une mise en œuvre inappropriée des produits.

Information sur la marque

SPRINGCARD, le logo SPRINGCARD sont des marques déposées de SPRINGCARD SAS. Les autres noms de marques, noms de produits, ou marques déposées, appartiennent à leurs titulaires respectifs. Les informations contenues dans ce document peuvent faire l'objet de modifications sans préavis.

Information sur le Copyright

Les informations contenues dans ce document sont soit disponibles publiquement, soit constituent la propriété intellectuelle de SPRINGCARD et/ou ses fournisseurs ou partenaires.

Vous pouvez télécharger, lire, copier et imprimer ce document en respectant les conditions suivantes : (1) ce document ne peut être utilisé qu'à des fins privées, éducatives, et non commerciales, et (2) ce document ne doit en aucun cas être modifié ou renommé. Vous ne pouvez pas utiliser, télécharger, copier, imprimer, exposer, reproduire, publier, certifier, poster, transmettre ou distribuer ce document, en entier ou en partie, sans la permission écrite préalable de SPRINGCARD.

Copyright © SPRINGCARD SAS 2018, tous droits réservés.

Éditeur

SPRINGCARD SAS au capital de 227 000 €

RCS EVRY B 429 665 482

Parc Gutenberg, 13 voie La Cardon, 91120 Palaiseau – France

Contact

Pour plus d'informations et pour identifier votre revendeur ou distributeur, merci de consulter notre site Internet :

www.springcard.com