```
//===============================================================;
//
//  CARDMOD.H
//
//  Abstract:
//      This is the header file commonly used for card modules.
//
//  This source code is only intended as a supplement to existing Microsoft
//  documentation.
//
//  THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
//  KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
//  IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
//  PURPOSE.
//
//  Copyright (C) Microsoft Corporation.  All Rights Reserved.
//
//===============================================================;
#ifndef __CARDMOD__H__
#define __CARDMOD__H__

#include <windows.h>
#include <wincrypt.h>
#pragma warning(push)
#pragma warning(disable:4201)
// Disable error C4201 in public header
//  nonstandard extension used : nameless struct/union
#include <winscard.h>
#pragma warning(pop)
#include <specstrings.h>

// This value should be passed to
//
//  SCardSetCardTypeProviderName
//  SCardGetCardTypeProviderName
//
// in order to query and set the Card Specific Module to be used
// for a given card.
#define SCARD_PROVIDER_CARD_MODULE 0x80000001

typedef struct _CARD_DATA CARD_DATA, *PCARD_DATA;

//
// This define can be used as a return value for queries involving
// card data that may be impossible to determine on a given card
// OS, such as the number of available card storage bytes.
//
#define CARD_DATA_VALUE_UNKNOWN                     ((DWORD) -1)

```

```
//
// Well Known Logical Names
//

//
// Logical Directory Names
//

// Second-level logical directories

#define szBASE_CSP_DIR                              "mscp"

#define szINTERMEDIATE_CERTS_DIR                    "mscerts"

//
// Logical File Names
//
// When requesting (or otherwise referring to) any logical file, the full path
// must be used, including when referring to well known files.  For example,
// to request the wszCONTAINER_MAP_FILE, the provided name will be
// "/mscp/cmapfile".
//

// Well known logical files under Microsoft
#define szCACHE_FILE                                "cardcf"

#define szCARD_IDENTIFIER_FILE                      "cardid"

// Well known logical files under CSP
#define szCONTAINER_MAP_FILE                        "cmapfile"
#define szROOT_STORE_FILE                           "msroots"

//
// Well known logical files under User Certs
//
// The following prefixes are appended with the container index of the
// associated key.  For example, the certificate associated with the
// Key Exchange key in container index 2 will have the name:
//  "/mscp/kxc2"
//
#define szUSER_SIGNATURE_CERT_PREFIX                "ksc"
#define szUSER_KEYEXCHANGE_CERT_PREFIX              "kxc"
#define szUSER_SIGNATURE_PRIVATE_KEY_PREFIX         "kss"
#define szUSER_SIGNATURE_PUBLIC_KEY_PREFIX          "ksp"
#define szUSER_KEYEXCHANGE_PRIVATE_KEY_PREFIX       "kxs"
#define szUSER_KEYEXCHANGE_PUBLIC_KEY_PREFIX        "kxp"

//
// Logical Card User Names
```

```
//
#define wszCARD_USER_EVERYONE                           L"anonymous"
#define wszCARD_USER_USER                               L"user"
#define wszCARD_USER_ADMIN                              L"admin"

// new ecc key specs

#define AT_ECDSA_P256      3
#define AT_ECDSA_P384      4
#define AT_ECDSA_P521      5
#define AT_ECDHE_P256      6
#define AT_ECDHE_P384      7
#define AT_ECDHE_P521      8

//
// Type: CARD_CACHE_FILE_FORMAT
//
// This struct is used as the file format of the cache file,
// as stored on the card.
//

#define CARD_CACHE_FILE_CURRENT_VERSION         1

typedef struct _CARD_CACHE_FILE_FORMAT
{
    BYTE bVersion;
    BYTE bPinsFreshness;

    WORD wContainersFreshness;
    WORD wFilesFreshness;
} CARD_CACHE_FILE_FORMAT, *PCARD_CACHE_FILE_FORMAT;

//
// Type: CONTAINER_MAP_RECORD
//
// This structure describes the format of the Base CSP's container map file,
// stored on the card.  This is well-known logical file wszCONTAINER_MAP_FILE.
// The file consists of zero or more of these records.
//
#define MAX_CONTAINER_NAME_LEN                  39

// This flag is set in the CONTAINER_MAP_RECORD bFlags member if the
// corresponding container is valid and currently exists on the card.
// If the container is deleted, its bFlags field must be cleared.
#define CONTAINER_MAP_VALID_CONTAINER           1

// This flag is set in the CONTAINER_MAP_RECORD bFlags
// member if the corresponding container is the default container on the card.
#define CONTAINER_MAP_DEFAULT_CONTAINER         2
```

```

typedef struct _CONTAINER_MAP_RECORD
{
    WCHAR wszGuid [MAX_CONTAINER_NAME_LEN + 1];
    BYTE bFlags;
    BYTE bReserved;
    WORD wSigKeySizeBits;
    WORD wKeyExchangeKeySizeBits;
} CONTAINER_MAP_RECORD, *PCONTAINER_MAP_RECORD;

//
// Converts a card filename string from unicode to ansi
//
DWORD WINAPI I_CardConvertFileNameToAnsi(
    IN      PCARD_DATA pCardData,
    __in    LPWSTR wszUnicodeName,
    __out   LPSTR *ppszAnsiName);

// Logical Directory Access Conditions
typedef enum
{
    InvalidDirAc = 0,

    // User Read, Write
    UserCreateDeleteDirAc,

    // Admin Write
    AdminCreateDeleteDirAc

} CARD_DIRECTORY_ACCESS_CONDITION;

// Logical File Access Conditions
typedef enum
{
    // Invalid value, chosed to cooincide with common initialization
    // of memory
    InvalidAc = 0,

    // Everyone     Read
    // User         Read, Write
    //
    // Example:  A user certificate file.
    EveryoneReadUserWriteAc,

    // Everyone     None
    // User         Write, Execute
    //
    // Example:  A private key file.
    UserWriteExecuteAc,

    // Everyone     Read
    // Admin        Read, Write
    //
    // Example:  The Card Identifier file.
    EveryoneReadAdminWriteAc,

    // Explicit value to set when it is desired to say that
    // it is unknown
    UnknownAc,

    // Everyone No Access
    // User Read Write
    //
    // Example:  A password wallet file.

    UserReadWriteAc,
    // Everyone/User No Access
    // Admin Read Write
    //
    // Example:  Administration data.

    AdminReadWriteAc
} CARD_FILE_ACCESS_CONDITION;

//
// Function: CardAcquireContext
//
// Purpose: Initialize the CARD_DATA structure which will be used by
//          the CSP to interact with a specific card.
//
typedef DWORD (WINAPI *PFN_CARD_ACQUIRE_CONTEXT)(
    IN      PCARD_DATA  pCardData,
    __in    DWORD       dwFlags);

DWORD
WINAPI
CardAcquireContext(
    IN      PCARD_DATA  pCardData,
    __in    DWORD       dwFlags);

//
// Function: CardDeleteContext
//
// Purpose: Free resources consumed by the CARD_DATA structure.
//
typedef DWORD (WINAPI *PFN_CARD_DELETE_CONTEXT)(
    __inout     PCARD_DATA  pCardData);

```

```
DWORD
WINAPI
CardDeleteContext(
    __inout     PCARD_DATA  pCardData);

//
// Function: CardQueryCapabilities
//
// Purpose: Query the card module for specific functionality
//          provided by this card.
//
#define CARD_CAPABILITIES_CURRENT_VERSION 1

typedef struct _CARD_CAPABILITIES
{
    IN OUT DWORD            dwVersion;
    IN     BOOL             fCertificateCompression;
    IN     BOOL             fKeyGen;
} CARD_CAPABILITIES, *PCARD_CAPABILITIES;

typedef DWORD (WINAPI *PFN_CARD_QUERY_CAPABILITIES)(
    __in      PCARD_DATA          pCardData,
    __in      PCARD_CAPABILITIES  pCardCapabilities);

DWORD
WINAPI
CardQueryCapabilities(
    __in      PCARD_DATA          pCardData,
    __in      PCARD_CAPABILITIES  pCardCapabilities);

//
// Function: CardDeleteContainer
//
// Purpose: Delete the specified key container.
//
typedef DWORD (WINAPI *PFN_CARD_DELETE_CONTAINER)(
    __in      PCARD_DATA  pCardData,
    __in      BYTE        bContainerIndex,
    __in      DWORD       dwReserved);

DWORD
WINAPI
CardDeleteContainer(
    __in      PCARD_DATA  pCardData,
    __in      BYTE        bContainerIndex,
    __in      DWORD       dwReserved);

//
// Function: CardCreateContainer
```

```
//

#define CARD_CREATE_CONTAINER_KEY_GEN           1
#define CARD_CREATE_CONTAINER_KEY_IMPORT        2

typedef DWORD (WINAPI *PFN_CARD_CREATE_CONTAINER)(
    __in      PCARD_DATA  pCardData,
    __in      BYTE        bContainerIndex,
    __in      DWORD       dwFlags,
    __in      DWORD       dwKeySpec,
    __in      DWORD       dwKeySize,
    __in      PBYTE       pbKeyData);

DWORD
WINAPI
CardCreateContainer(
    __in      PCARD_DATA  pCardData,
    __in      BYTE        bContainerIndex,
    __in      DWORD       dwFlags,
    __in      DWORD       dwKeySpec,
    __in      DWORD       dwKeySize,
    __in      PBYTE       pbKeyData);

//
// Function: CardGetContainerInfo
//
// Purpose: Query for all public information available about
//          the named key container.  This includes the Signature
//          and Key Exchange type public keys, if they exist.
//
//          The pbSigPublicKey and pbKeyExPublicKey buffers contain the
//          Signature and Key Exchange public keys, respectively, if they
//          exist.  The format of these buffers is a Crypto
//          API PUBLICKEYBLOB -
//
//              BLOBHEADER
//              RSAPUBKEY
//              modulus
//
//          In the case of ECC public keys, the pbSigPublicKey will contain
//          the ECDSA key and pbKeyExPublicKey will contain the ECDH key if
//          they exist. ECC key structure -
//
//              BCRYPT_ECCKEY_BLOB
//              X coord (big endian)
//              Y coord (big endian)
//
#define CONTAINER_INFO_CURRENT_VERSION 1

```

```
typedef struct _CONTAINER_INFO
{
    IN OUT DWORD dwVersion;
    IN     DWORD dwReserved;

    OUT    DWORD cbSigPublicKey;
    OUT    PBYTE pbSigPublicKey;

    OUT    DWORD cbKeyExPublicKey;
    OUT    PBYTE pbKeyExPublicKey;
} CONTAINER_INFO, *PCONTAINER_INFO;

typedef DWORD (WINAPI *PFN_CARD_GET_CONTAINER_INFO)(
    __in      PCARD_DATA  pCardData,
    __in      BYTE        bContainerIndex,
    __in      DWORD       dwFlags,
    __in      PCONTAINER_INFO pContainerInfo);

DWORD
WINAPI
CardGetContainerInfo(
    __in      PCARD_DATA  pCardData,
    __in      BYTE        bContainerIndex,
    __in      DWORD       dwFlags,
    __in      PCONTAINER_INFO pContainerInfo);

//
// Function: CardAuthenticatePin
//
typedef DWORD (WINAPI *PFN_CARD_AUTHENTICATE_PIN)(
    __in                    PCARD_DATA  pCardData,
    __in                    LPWSTR      pwszUserId,
    __in_bcount(cbPin)      PBYTE       pbPin,
    __in                    DWORD       cbPin,
    __out_opt               PDWORD pcAttemptsRemaining);


DWORD
WINAPI
CardAuthenticatePin(
    __in                    PCARD_DATA  pCardData,
    __in                    LPWSTR      pwszUserId,
    __in_bcount(cbPin)      PBYTE       pbPin,
    __in                    DWORD       cbPin,
    __out_opt               PDWORD pcAttemptsRemaining);

//
// Function: CardGetChallenge
//
```

```
typedef DWORD (WINAPI *PFN_CARD_GET_CHALLENGE)(
    __in                                PCARD_DATA  pCardData,
    __out_bcount(*pcbChallengeData)     PBYTE       *ppbChallengeData,
    __out                               PDWORD      pcbChallengeData);

DWORD
WINAPI
CardGetChallenge(
    __in                                  PCARD_DATA pCardData,
    __deref_out_bcount(*pcbChallengeData) PBYTE      *ppbChallengeData,
    __out                                 PDWORD      pcbChallengeData);

//
// Function: CardAuthenticateChallenge
//
typedef DWORD (WINAPI *PFN_CARD_AUTHENTICATE_CHALLENGE)(
    __in                             PCARD_DATA  pCardData,
    __in_bcount(cbResponseData)      PBYTE       pbResponseData,
    __in                             DWORD       cbResponseData,
    __out_opt                        PDWORD pcAttemptsRemaining);

DWORD
WINAPI
CardAuthenticateChallenge(
    __in                             PCARD_DATA  pCardData,
    __in_bcount(cbResponseData)      PBYTE       pbResponseData,
    __in                             DWORD       cbResponseData,
    __out_opt                        PDWORD pcAttemptsRemaining);

//
// Function: CardUnblockPin
//
#define CARD_AUTHENTICATE_PIN_CHALLENGE_RESPONSE                 1
#define CARD_AUTHENTICATE_PIN_PIN                                2

typedef DWORD (WINAPI *PFN_CARD_UNBLOCK_PIN)(
    __in                               PCARD_DATA  pCardData,
    __in                               LPWSTR      pwszUserId,
    __in_bcount(cbAuthenticationData)  PBYTE       pbAuthenticationData,
    __in                               DWORD       cbAuthenticationData,
    __in_bcount(cbNewPinData)          PBYTE       pbNewPinData,
    __in                               DWORD       cbNewPinData,
    __in                               DWORD       cRetryCount,
    __in                               DWORD       dwFlags);

DWORD
WINAPI
CardUnblockPin(
    __in                               PCARD_DATA  pCardData,
    __in                                LPWSTR      pwszUserId,
    __in_bcount(cbAuthenticationData)  PBYTE       pbAuthenticationData,
    __in                                DWORD       cbAuthenticationData,
    __in_bcount(cbNewPinData)          PBYTE       pbNewPinData,
    __in                                DWORD       cbNewPinData,
    __in                                DWORD       cRetryCount,
    __in                                DWORD       dwFlags);

//
// Function: CardChangeAuthenticator
//
typedef DWORD (WINAPI *PFN_CARD_CHANGE_AUTHENTICATOR)(
    __in                                  PCARD_DATA  pCardData,
    __in                                  LPWSTR      pwszUserId,
    __in_bcount(cbCurrentAuthenticator)  PBYTE       pbCurrentAuthenticator,
    __in                                  DWORD       cbCurrentAuthenticator,
    __in_bcount(cbNewAuthenticator)      PBYTE       pbNewAuthenticator,
    __in                                  DWORD       cbNewAuthenticator,
    __in                                  DWORD       cRetryCount,
    __in                                  DWORD       dwFlags,
    __out_opt                             PDWORD pcAttemptsRemaining);

DWORD
WINAPI
CardChangeAuthenticator(
    __in                                  PCARD_DATA  pCardData,
    __in                                  LPWSTR      pwszUserId,
    __in_bcount(cbCurrentAuthenticator)  PBYTE       pbCurrentAuthenticator,
    __in                                  DWORD       cbCurrentAuthenticator,
    __in_bcount(cbNewAuthenticator)      PBYTE       pbNewAuthenticator,
    __in                                  DWORD       cbNewAuthenticator,
    __in                                  DWORD       cRetryCount,
    __in                                  DWORD       dwFlags,
    __out_opt                             PDWORD pcAttemptsRemaining);

//
// Function: CardDeauthenticate
//
// Purpose: De-authenticate the specified logical user name on the card.
//
// This is an optional API.  If implemented, this API is used instead
// of SCARD_RESET_CARD by the Base CSP.  An example scenario is leaving
// a transaction in which the card has been authenticated (a Pin has been
// successfully presented).
//
// The pwszUserId parameter will point to a valid well-known User Name (see
// above).
//
// The dwFlags parameter is currently unused and will always be zero.
```

```
//
// Card modules that choose to not implement this API must set the CARD_DATA
// pfnCardDeauthenticate pointer to NULL.
//
typedef DWORD (WINAPI *PFN_CARD_DEAUTHENTICATE)(
    __in      PCARD_DATA  pCardData,
    __in      LPWSTR      pwszUserId,
    __in      DWORD       dwFlags);

DWORD
WINAPI
CardDeauthenticate(
    __in    PCARD_DATA  pCardData,
    __in    LPWSTR      pwszUserId,
    __in    DWORD       dwFlags);

// Directory Control Group

//
// Function: CardCreateDirectory
//
// Purpose: Register the specified application name on the card, and apply the
//          provided access condition.
//
// Return Value:
//          ERROR_FILE_EXISTS - directory already exists
//
typedef DWORD (WINAPI *PFN_CARD_CREATE_DIRECTORY)(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName,
    __in    CARD_DIRECTORY_ACCESS_CONDITION AccessCondition);

DWORD
WINAPI
CardCreateDirectory(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName,
    __in    CARD_DIRECTORY_ACCESS_CONDITION AccessCondition);

//
// Function: CardDeleteDirectory
//
// Purpose: Unregister the specified application from the card.
//
// Return Value:
//          SCARD_E_DIR_NOT_FOUND - directory does not exist
//          ERROR_DIR_NOT_EMPTY - the directory is not empty
//
typedef DWORD (WINAPI *PFN_CARD_DELETE_DIRECTORY)(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName);

DWORD
WINAPI
CardDeleteDirectory(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName);

// File Control Group

//
// Function: CardCreateFile
//
typedef DWORD (WINAPI *PFN_CARD_CREATE_FILE)(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName,
    __in    LPSTR       pszFileName,
    __in    DWORD       cbInitialCreationSize,
    __in    CARD_FILE_ACCESS_CONDITION AccessCondition);

DWORD
WINAPI
CardCreateFile(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName,
    __in    LPSTR       pszFileName,
    __in    DWORD       cbInitialCreationSize,
    __in    CARD_FILE_ACCESS_CONDITION AccessCondition);

//
// Function: CardReadFile
//
// Purpose: Read the specified file from the card.
//
//          The pbData parameter should be allocated
//          by the card module and freed by the CSP.  The card module
//          must set the cbData parameter to the size of the returned buffer.
//
typedef DWORD (WINAPI *PFN_CARD_READ_FILE)(
    __in                        PCARD_DATA  pCardData,
    __in                        LPSTR       pszDirectoryName,
    __in                        LPSTR       pszFileName,
    __in                        DWORD       dwFlags,
    __out_bcount(*pcbData)      PBYTE      *ppbData,
    __out                       PDWORD      pcbData);

DWORD
WINAPI
```

```
CardReadFile(
    __in                            PCARD_DATA  pCardData,
    __in                            LPSTR       pszDirectoryName,
    __in                            LPSTR       pszFileName,
    __in                            DWORD       dwFlags,
    __deref_out_bcount(*pcbData)    PBYTE      *ppbData,
    __out                           PDWORD      pcbData);

//
// Function: CardWriteFile
//
typedef DWORD (WINAPI *PFN_CARD_WRITE_FILE)(
    __in                    PCARD_DATA  pCardData,
    __in                    LPSTR       pszDirectoryName,
    __in                    LPSTR       pszFileName,
    __in                    DWORD       dwFlags,
    __in_bcount(cbData)     PBYTE       pbData,
    __in                    DWORD       cbData);

DWORD
WINAPI
CardWriteFile(
    __in                    PCARD_DATA  pCardData,
    __in                    LPSTR       pszDirectoryName,
    __in                    LPSTR       pszFileName,
    __in                    DWORD       dwFlags,
    __in_bcount(cbData)     PBYTE       pbData,
    __in                    DWORD       cbData);

//
// Function: CardDeleteFile
//
typedef DWORD (WINAPI *PFN_CARD_DELETE_FILE)(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName,
    __in    LPSTR       pszFileName,
    __in    DWORD       dwFlags);

DWORD
WINAPI
CardDeleteFile(
    __in    PCARD_DATA  pCardData,
    __in    LPSTR       pszDirectoryName,
    __in    LPSTR       pszFileName,
    __in    DWORD       dwFlags);

//
// Function: CardEnumFiles
//
```

```
// Purpose: Return a multi-string list of the general files
//          present on this card.  The multi-string is allocated
//          by the card module and must be freed by the CSP.
//
//  The caller must provide a logical file directory name in the
//  pmwszFileNames parameter (see Logical Directory Names, above).
//  The logical directory name indicates which group of files will be
//  enumerated.
//
//  The logical directory name is expected to be a static string, so the
//  the card module will not free it.  The card module
//  will allocate a new buffer in *pmwszFileNames to store the multi-string
//  list of enumerated files using pCardData->pfnCspAlloc.
//
//  If the function fails for any reason, *pmwszFileNames is set to NULL.
//
typedef DWORD (WINAPI *PFN_CARD_ENUM_FILES)(
    __in      PCARD_DATA  pCardData,
    __in      LPSTR       pszDirectoryName,
    __out_ecount(*pdwcbFileName)
              LPSTR      *pmszFileNames,
    __out     LPDWORD     pdwcbFileName,
    __in      DWORD       dwFlags);

DWORD
WINAPI
CardEnumFiles(
    __in      PCARD_DATA  pCardData,
    __in      LPSTR       pszDirectoryName,
    __out_ecount(*pdwcbFileName)
              LPSTR      *pmszFileNames,
    __out     LPDWORD     pdwcbFileName,
    __in      DWORD       dwFlags);

//
// Function: CardGetFileInfo
//
#define CARD_FILE_INFO_CURRENT_VERSION 1

typedef struct _CARD_FILE_INFO
{
    IN OUT DWORD dwVersion;
    OUT    DWORD cbFileSize;
    OUT    CARD_FILE_ACCESS_CONDITION AccessCondition;
} CARD_FILE_INFO, *PCARD_FILE_INFO;

typedef DWORD (WINAPI *PFN_CARD_GET_FILE_INFO)(
    __in         PCARD_DATA  pCardData,
    __in         LPSTR      pszDirectoryName,
    __in          LPSTR      pszFileName,
    __in          PCARD_FILE_INFO pCardFileInfo);

DWORD
WINAPI
CardGetFileInfo(
    __in          PCARD_DATA  pCardData,
    __in          LPSTR      pszDirectoryName,
    __in          LPSTR      pszFileName,
    __in          PCARD_FILE_INFO pCardFileInfo);

//
// Function: CardQueryFreeSpace
//
#define CARD_FREE_SPACE_INFO_CURRENT_VERSION 1

typedef struct _CARD_FREE_SPACE_INFO
{
    IN OUT DWORD dwVersion;
    OUT    DWORD dwBytesAvailable;
    OUT    DWORD dwKeyContainersAvailable;
    OUT    DWORD dwMaxKeyContainers;

} CARD_FREE_SPACE_INFO, *PCARD_FREE_SPACE_INFO;

typedef DWORD (WINAPI *PFN_CARD_QUERY_FREE_SPACE)(
    __in      PCARD_DATA  pCardData,
    __in      DWORD       dwFlags,
    __in      PCARD_FREE_SPACE_INFO pCardFreeSpaceInfo);

DWORD
WINAPI
CardQueryFreeSpace(
    __in      PCARD_DATA  pCardData,
    __in      DWORD       dwFlags,
    __in      PCARD_FREE_SPACE_INFO pCardFreeSpaceInfo);

//
// Function: CardQueryKeySizes
//
#define CARD_KEY_SIZES_CURRENT_VERSION 1

typedef struct _CARD_KEY_SIZES
{
    IN OUT  DWORD dwVersion;

    OUT     DWORD dwMinimumBitlen;
    OUT     DWORD dwDefaultBitlen;
    OUT     DWORD dwMaximumBitlen;
    OUT     DWORD dwIncrementalBitlen;

} CARD_KEY_SIZES, *PCARD_KEY_SIZES;

typedef DWORD (WINAPI *PFN_CARD_QUERY_KEY_SIZES)(
    __in      PCARD_DATA  pCardData,
    __in      DWORD       dwKeySpec,
    __in      DWORD       dwFlags,
    __in      PCARD_KEY_SIZES pKeySizes);

DWORD
WINAPI
CardQueryKeySizes(
    __in      PCARD_DATA  pCardData,
    __in      DWORD       dwKeySpec,
    __in      DWORD       dwFlags,
    __in      PCARD_KEY_SIZES pKeySizes);

//
// Function: CardRSADecrypt
//
// Purpose: Perform a private key decryption on the supplied data.  The
//          card module should assume that pbData is the length of the
//          key modulus.
//
#define CARD_RSA_KEY_DECRYPT_INFO_CURRENT_VERSION 1

typedef struct _CARD_RSA_DECRYPT_INFO
{
    __in                        DWORD dwVersion;
    __in                        BYTE bContainerIndex;

    // For RSA operations, this should be AT_SIGNATURE or AT_KEYEXCHANGE.
    __in                        DWORD dwKeySpec;

    // This is the buffer and length that the caller expects to be decrypted.
    // For RSA operations, cbData is redundant since the length of the buffer
    // should always be equal to the length of the key modulus.
    __out_bcount(cbData)        PBYTE pbData;
    __out                       DWORD cbData;

} CARD_RSA_DECRYPT_INFO, *PCARD_RSA_DECRYPT_INFO;

typedef DWORD (WINAPI *PFN_CARD_RSA_DECRYPT)(
    __in        PCARD_DATA              pCardData,
    __inout     PCARD_RSA_DECRYPT_INFO  pInfo);

DWORD
WINAPI
```

```
CardRSADecrypt(
    __in        PCARD_DATA              pCardData,
    __inout     PCARD_RSA_DECRYPT_INFO  pInfo);

#define CARD_PADDING_INFO_PRESENT 0x40000000
#define CARD_BUFFER_SIZE_ONLY     0x20000000
#define CARD_PADDING_NONE         0x00000001
#define CARD_PADDING_PKCS1        0x00000002
#define CARD_PADDING_PSS          0x00000004

// CARD_SIGNING_INFO_BASIC_VERSION is provided for thos applications
// do not intend to support passing in the pPaddingInfo structure
#define CARD_SIGNING_INFO_BASIC_VERSION 1

//
// Function: CardSignData
//
// Purpose: Sign inupt data using a specified key
//
#define CARD_SIGNING_INFO_CURRENT_VERSION 2
typedef struct _CARD_SIGNING_INFO
{
    IN DWORD  dwVersion;

    IN BYTE   bContainerIndex;

    // See dwKeySpec constants
    IN DWORD  dwKeySpec;

    // If CARD_BUFFER_SIZE_ONLY flag is present then the card
    // module should return only the size of the resulting
    // key in cbSignedData
    IN DWORD  dwSigningFlags;

    // If the aiHashAlg is non zero, then it specifies the algorithm
    // to use when padding the data using PKCS
    IN ALG_ID aiHashAlg;

    // This is the buffer and length that the caller expects to be signed.
    // Signed version is allocated a buffer and put in cb/pbSignedData.  That 
      should
    // be freed using PFN_CSP_FREE callback.
    IN PBYTE  pbData;
    IN DWORD  cbData;

    OUT PBYTE  pbSignedData;
    OUT DWORD  cbSignedData;

    // The following parameters are new in version 2 of the
    // CARD_SIGNING_INFO structure.
    // If CARD_PADDING_INFO_PRESENT is set in dwSigningFlags then
    // pPaddingInfo will point to the BCRYPT_PADDING_INFO structure
    // defined by dwPaddingType.  Currently supported values are
    // CARD_PADDING_PKCS1, CARD_PADDING_PSS and CARD_PADDING_NONE
    IN LPVOID pPaddingInfo;
    IN DWORD  dwPaddingType;
} CARD_SIGNING_INFO, *PCARD_SIGNING_INFO;

typedef DWORD (WINAPI *PFN_CARD_SIGN_DATA)(
    __in      PCARD_DATA          pCardData,
    __in      PCARD_SIGNING_INFO  pInfo);

DWORD
WINAPI
CardSignData(
    __in      PCARD_DATA          pCardData,
    __in      PCARD_SIGNING_INFO  pInfo);

//
// Type: CARD_DH_AGREEMENT_INFO
//
// CARD_DH_AGREEMENT_INFO version 1 is no longer supported and should
// not be implemented
//

#define CARD_DH_AGREEMENT_INFO_VERSION 2

typedef struct _CARD_DH_AGREEMENT_INFO
{
    IN  DWORD dwVersion;
    IN  BYTE  bContainerIndex;
    IN  DWORD dwFlags;
    IN  DWORD dwPublicKey;
    IN  PBYTE pbPublicKey;
    IN  PBYTE pbReserved;
    IN  DWORD cbReserved;

    OUT BYTE bSecretAgreementIndex;
} CARD_DH_AGREEMENT_INFO, *PCARD_DH_AGREEMENT_INFO;

//
// Function:  CardConstructDHAgreement
//
// Purpose: compute a DH secret agreement from a ECDH key on the card
// and the public portion of another ECDH key
//

typedef DWORD (WINAPI *PFN_CARD_CONSTRUCT_DH_AGREEMENT)(
    __in     PCARD_DATA pCardData,
    __in     PCARD_DH_AGREEMENT_INFO pAgreementInfo);

DWORD WINAPI CardConstructDHAgreement(
    __in     PCARD_DATA pCardData,
    __in     PCARD_DH_AGREEMENT_INFO pAgreementInfo);

//
// Type: CARD_DERIVE_KEY_INFO
//
#define CARD_DERIVE_KEY_VERSION 1

typedef struct _CARD_DERIVE_KEY
{
    IN  DWORD              dwVersion;

    // If CARD_BUFFER_SIZE_ONLY is passed then the card module
    // should return only the size of the resulting key in
    // cbDerivedKey
    IN  DWORD              dwFlags;
    IN  LPWSTR             pwszKDF;
    IN  BYTE               bSecretAgreementIndex;

    IN  PVOID              pParameterList;

    OUT PBYTE              pbDerivedKey;
    OUT DWORD              cbDerivedKey;

} CARD_DERIVE_KEY, *PCARD_DERIVE_KEY;

//
// Function:  CardDeriveKey
//
// Purpose: Generate a dervived session key using a generated agreed
// secret and various other parameters.
//

typedef DWORD (WINAPI *PFN_CARD_DERIVE_KEY)(
    __in     PCARD_DATA pCardData,
    __in     PCARD_DERIVE_KEY pAgreementInfo);

DWORD WINAPI CardDeriveKey(
    __in     PCARD_DATA pCardData,
    __in     PCARD_DERIVE_KEY pAgreementInfo);

//
// Function:  CardDestroyAgreement
//
// Purpose: Force a deletion of the DH agreed secret.
```

```
//

typedef DWORD (WINAPI *PFN_CARD_DESTROY_DH_AGREEMENT)(
    __in PCARD_DATA pCardData,
    __in BYTE       bSecretAgreementIndex,
    __in DWORD      dwFlags);

DWORD WINAPI CardDestroyDHAgreement(
    __in PCARD_DATA pCardData,
    __in BYTE       bSecretAgreementIndex,
    __in DWORD      dwFlags);

//
// Function:  CspGetDHAgreement
//
// Purpose: The CARD_DERIVE_KEY structure contains a list of parameters
// (pParameterList) which might contain a reference to one or more addition
// agreed secrets (KDF_NCRYPT_SECRET_HANDLE).  This callback is provided by
// the caller of CardDeriveKey and will translate the parameter into the
// on card agreed secret handle.
//

typedef DWORD (WINAPI *PFN_CSP_GET_DH_AGREEMENT)(
    IN  PCARD_DATA          pCardData,
    IN  PVOID               hSecretAgreement,
    OUT BYTE*               pbSecretAgreementIndex,
    IN  DWORD               dwFlags);

DWORD WINAPI CspGetDHAgreement(
    __in  PCARD_DATA          pCardData,
    __in  PVOID               hSecretAgreement,
    __out BYTE*               pbSecretAgreementIndex,
    __in  DWORD               dwFlags);

//
// Memory Management Routines
//
// These routines are supplied to the card module
// by the calling CSP.
//

//
// Function: PFN_CSP_ALLOC
//
typedef LPVOID (WINAPI *PFN_CSP_ALLOC)(
    IN      SIZE_T      Size);

//
// Function: PFN_CSP_REALLOC
```

```
//
typedef LPVOID (WINAPI *PFN_CSP_REALLOC)(
    IN      LPVOID      Address,
    IN      SIZE_T      Size);

//
// Function: PFN_CSP_FREE
//
// Note: Data allocated for the CSP by the card module must
//       be freed by the CSP.
//
typedef void (WINAPI *PFN_CSP_FREE)(
    IN      LPVOID      Address);

//
// Function: PFN_CSP_CACHE_ADD_FILE
//
// A copy of the pbData parameter is added to the cache.
//
typedef DWORD (WINAPI *PFN_CSP_CACHE_ADD_FILE)(
    IN      PVOID       pvCacheContext,
    IN      LPWSTR      wszTag,
    IN      DWORD       dwFlags,
    IN      PBYTE       pbData,
    IN      DWORD       cbData);

//
// Function: PFN_CSP_CACHE_LOOKUP_FILE
//
// If the cache lookup is successful,
// the caller must free the *ppbData pointer with pfnCspFree.
//
typedef DWORD (WINAPI *PFN_CSP_CACHE_LOOKUP_FILE)(
    IN      PVOID       pvCacheContext,
    IN      LPWSTR      wszTag,
    IN      DWORD       dwFlags,
    IN      PBYTE      *ppbData,
    IN      PDWORD      pcbData);

//
// Function: PFN_CSP_CACHE_DELETE_FILE
//
// Deletes the specified item from the cache.
//
typedef DWORD (WINAPI *PFN_CSP_CACHE_DELETE_FILE)(
    IN      PVOID       pvCacheContext,
    IN      LPWSTR      wszTag,
    IN      DWORD       dwFlags);

```

```
//
// Function: PFN_CSP_PAD_DATA
//
// Deletes Callback to pad buffer for cyrpto operation.  Used when
// the card does not provide this.
//
typedef DWORD (WINAPI *PFN_CSP_PAD_DATA)(
    IN      PCARD_SIGNING_INFO pSigningInfo,
    IN      DWORD              cbMaxWidth,
    OUT     DWORD*             pcbPaddedBuffer,
    OUT     PBYTE*             ppbPaddedBuffer);

// ****************
// PIN SUPPORT
// ****************

//
// There are 8 PINs currently defined in version 6. PIN values 0, 1 and 2 are
// reserved for backwards compatibility, whereas PIN values 3-7 can be used
// as additional PINs to protect key containers.
//

typedef     DWORD                       PIN_ID, *PPIN_ID;
typedef     DWORD                       PIN_SET, *PPIN_SET;

#define     MAX_PINS                    8

#define     ROLE_EVERYONE               0
#define     ROLE_USER                   1
#define     ROLE_ADMIN                  2

#define     PIN_SET_ALL_ROLES           0xFF
#define     CREATE_PIN_SET(PinId)       (1 << PinId)
#define     SET_PIN(PinSet, PinId)      PinSet |= CREATE_PIN_SET(PinId)
#define     IS_PIN_SET(PinSet, PinId)   (0 != (PinSet & CREATE_PIN_SET
  (PinId)))
#define     CLEAR_PIN(PinSet, PinId)    PinSet &= ~CREATE_PIN_SET(PinId)

#define     PIN_CHANGE_FLAG_UNBLOCK     0x01
#define     PIN_CHANGE_FLAG_CHANGEPIN   0x02

#define     CP_CACHE_MODE_GLOBAL_CACHE  1
#define     CP_CACHE_MODE_SESSION_ONLY  2
#define     CP_CACHE_MODE_NO_CACHE      3

#define     CARD_AUTHENTICATE_GENERATE_SESSION_PIN      0x10000000
#define     CARD_AUTHENTICATE_SESSION_PIN               0x20000000

#define     CARD_PIN_STRENGTH_PLAINTEXT                 0x1
```

```
#define     CARD_PIN_STRENGTH_SESSION_PIN                 0x2

#define     CARD_PIN_SILENT_CONTEXT                       0x00000040

typedef enum
{
    AlphaNumericPinType = 0,            // Regular PIN
    ExternalPinType,                    // Biometric PIN
    ChallengeResponsePinType,           // Challenge/Response PIN
    EmptyPinType                        // No PIN
} SECRET_TYPE;

typedef enum
{
    AuthenticationPin,                  // Authentication PIN
    DigitalSignaturePin,                // Digital Signature PIN
    EncryptionPin,                      // Encryption PIN
    NonRepudiationPin,                  // Non Repudiation PIN
    AdministratorPin,                   // Administrator PIN
    PrimaryCardPin                      // Primary Card PIN
} SECRET_PURPOSE;

typedef enum
{
    PinCacheNormal = 0,
    PinCacheTimed,
    PinCacheNone
} PIN_CACHE_POLICY_TYPE;

#define      PIN_CACHE_POLICY_CURRENT_VERSION     6

typedef struct _PIN_CACHE_POLICY
{
    IN OUT  DWORD                                   dwVersion;
    OUT     PIN_CACHE_POLICY_TYPE                   PinCachePolicyType;
    OUT     DWORD                                   dwPinCachePolicyInfo;
} PIN_CACHE_POLICY, *PPIN_CACHE_POLICY;

#define      PIN_INFO_CURRENT_VERSION             6

#define      PIN_INFO_REQUIRE_SECURE_ENTRY        1

typedef struct _PIN_INFO
{
    IN OUT  DWORD                                   dwVersion;
    OUT     SECRET_TYPE                             PinType;
    OUT     SECRET_PURPOSE                          PinPurpose;
    OUT     PIN_SET                                 dwChangePermission;
    OUT     PIN_SET                                 dwUnblockPermission;
    OUT     PIN_CACHE_POLICY                    PinCachePolicy;
    OUT     DWORD                               dwFlags;
} PIN_INFO, *PPIN_INFO;

typedef DWORD (WINAPI *PFN_CARD_GET_CHALLENGE_EX)(
    __in                                PCARD_DATA  pCardData,
    __in                                PIN_ID      PinId,
    __out_bcount(*pcbChallengeData)     PBYTE       *ppbChallengeData,
    __out                               PDWORD      pcbChallengeData,
    __in                                DWORD       dwFlags);

DWORD
WINAPI
CardGetChallengeEx(
    __in                                  PCARD_DATA pCardData,
    __in                                  PIN_ID      PinId,
    __deref_out_bcount(*pcbChallengeData) PBYTE      *ppbChallengeData,
    __out                                 PDWORD      pcbChallengeData,
    __in                                  DWORD       dwFlags);

typedef DWORD (WINAPI *PFN_CARD_AUTHENTICATE_EX)(
    __in    PCARD_DATA                                  pCardData,
    __in    PIN_ID                                      PinId,
    __in    DWORD                                       dwFlags,
    __in_bcount(cbPinData)    PBYTE                     pbPinData,
    __in    DWORD                                       cbPinData,
    __deref_out_bcount_opt(*pcbSessionPin) PBYTE        *ppbSessionPin,
    __out_opt PDWORD                                    pcbSessionPin,
    __out_opt PDWORD                                    pcAttemptsRemaining);

DWORD
WINAPI
CardAuthenticateEx(
    __in    PCARD_DATA                                  pCardData,
    __in    PIN_ID                                      PinId,
    __in    DWORD                                       dwFlags,
    __in    PBYTE                                       pbPinData,
    __in    DWORD                                       cbPinData,
    __deref_out_bcount_opt(*pcbSessionPin) PBYTE        *ppbSessionPin,
    __out_opt PDWORD                                    pcbSessionPin,
    __out_opt PDWORD                                    pcAttemptsRemaining);

typedef DWORD (WINAPI *PFN_CARD_CHANGE_AUTHENTICATOR_EX)(
    __in    PCARD_DATA                                  pCardData,
    __in    DWORD                                       dwFlags,
    __in    PIN_ID                                      dwAuthenticatingPinId,
    __in_bcount(cbAuthenticatingPinData) PBYTE          pbAuthenticatingPinData,
    __in    DWORD                                       cbAuthenticatingPinData,
    __in    PIN_ID                                      dwTargetPinId,
    __in_bcount(cbTargetData)            PBYTE    pbTargetData,
    __in    DWORD                                 cbTargetData,
    __in    DWORD                                 cRetryCount,
    __out_opt PDWORD                              pcAttemptsRemaining);

DWORD WINAPI CardChangeAuthenticatorEx(
    __in    PCARD_DATA                            pCardData,
    __in    DWORD                                 dwFlags,
    __in    PIN_ID                                dwAuthenticatingPinId,
    __in_bcount(cbAuthenticatingPinData) PBYTE    pbAuthenticatingPinData,
    __in    DWORD                                 cbAuthenticatingPinData,
    __in    PIN_ID                                dwTargetPinId,
    __in_bcount(cbTargetData)            PBYTE    pbTargetData,
    __in    DWORD                                 cbTargetData,
    __in    DWORD                                 cRetryCount,
    __out_opt PDWORD                              pcAttemptsRemaining);

typedef DWORD (WINAPI *PFN_CARD_DEAUTHENTICATE_EX)(
    __in    PCARD_DATA                            pCardData,
    __in    PIN_SET                               PinId,
    __in    DWORD                                 dwFlags);

DWORD WINAPI CardDeauthenticateEx(
    __in    PCARD_DATA                            pCardData,
    __in    PIN_SET                               PinId,
    __in    DWORD                                 dwFlags);

// *******************
// Container Porperties
// *******************

#define CCP_CONTAINER_INFO               L"Container Info" // Read only
#define CCP_PIN_IDENTIFIER               L"PIN Identifier"

typedef DWORD (WINAPI *PFN_CARD_GET_CONTAINER_PROPERTY)(
    __in    PCARD_DATA                                    pCardData,
    __in    BYTE                                          bContainerIndex,
    __in    LPCWSTR                                       wszProperty,
    __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE      pbData,
    __in    DWORD                                         cbData,
    __out   PDWORD                                        pdwDataLen,
    __in    DWORD                                         dwFlags);

DWORD WINAPI CardGetContainerProperty(
    __in    PCARD_DATA                                    pCardData,
    __in    BYTE                                          bContainerIndex,
    __in    LPCWSTR                                       wszProperty,
    __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE      pbData,
    __in    DWORD                                         cbData,
```

```c
    __out  PDWORD                                   pdwDataLen,
    __in   DWORD                                    dwFlags);

typedef DWORD (WINAPI *PFN_CARD_SET_CONTAINER_PROPERTY)(
    __in   PCARD_DATA                               pCardData,
    __in   BYTE                                     bContainerIndex,
    __in   LPCWSTR                                  wszProperty,
    __in_bcount(cbDataLen)  PBYTE                   pbData,
    __in   DWORD                                    cbDataLen,
    __in   DWORD                                    dwFlags);

DWORD WINAPI CardSetContainerProperty(
    __in   PCARD_DATA                               pCardData,
    __in   BYTE                                     bContainerIndex,
    __in   LPCWSTR                                  wszProperty,
    __in_bcount(cbDataLen)  PBYTE                   pbData,
    __in   DWORD                                    cbDataLen,
    __in   DWORD                                    dwFlags);

// ******************
// Card Properties
// ******************

#define CP_CARD_FREE_SPACE              L"Free Space"               // Read only
#define CP_CARD_CAPABILITIES            L"Capabilities"             // Read only
#define CP_CARD_KEYSIZES                L"Key Sizes"                // Read only

#define CP_CARD_READ_ONLY               L"Read Only Mode"
#define CP_CARD_CACHE_MODE              L"Cache Mode"
#define CP_SUPPORTS_WIN_X509_ENROLLMENT L"Supports Windows x.509 Enrollment"

#define CP_CARD_GUID                    L"Card Identifier"
#define CP_CARD_SERIAL_NO               L"Card Serial Number"

#define CP_CARD_PIN_INFO                L"PIN Information"
#define CP_CARD_LIST_PINS               L"PIN List"                 // Read only
#define CP_CARD_AUTHENTICATED_STATE     L"Authenticated State"      // Read only

#define CP_CARD_PIN_STRENGTH_VERIFY     L"PIN Strength Verify"      // Read only
#define CP_CARD_PIN_STRENGTH_CHANGE     L"PIN Strength Change"      // Read only
#define CP_CARD_PIN_STRENGTH_UNBLOCK    L"PIN Strength Unblock"     // Read only
```

```c

#define CP_PARENT_WINDOW                L"Parent Window"           // Write only
#define CP_PIN_CONTEXT_STRING           L"PIN Context String"      // Write only

typedef DWORD (WINAPI *PFN_CARD_GET_PROPERTY)(
    __in    PCARD_DATA                                  pCardData,
    __in    LPCWSTR                                     wszProperty,
    __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE   pbData,
    __in    DWORD                                       cbData,
    __out PDWORD                                        pdwDataLen,
    __in    DWORD                                       dwFlags);

DWORD WINAPI CardGetProperty(
    __in    PCARD_DATA                                  pCardData,
    __in    LPCWSTR                                     wszProperty,
    __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE   pbData,
    __in    DWORD                                       cbData,
    __out   PDWORD                                      pdwDataLen,
    __in    DWORD                                       dwFlags);

typedef DWORD (WINAPI *PFN_CARD_SET_PROPERTY)(
    __in    PCARD_DATA                                  pCardData,
    __in    LPCWSTR                                     wszProperty,
    __in_bcount(cbDataLen)  PBYTE                       pbData,
    __in    DWORD                                       cbDataLen,
    __in    DWORD                                       dwFlags);

DWORD WINAPI CardSetProperty(
    __in    PCARD_DATA                                  pCardData,
    __in    LPCWSTR                                     wszProperty,
    __in_bcount(cbDataLen)  PBYTE                       pbData,
    __in    DWORD                                       cbDataLen,
    __in    DWORD                                       dwFlags);

//
// Type: CARD_DATA
//

#define CARD_DATA_VERSION_SIX   6

// This version supports new features such as a designed
// CardSecretAgreement and key derivation functions.  Also
// added is the PKCS#1 2.1 (PSS) padding format.
#define CARD_DATA_VERSION_FIVE  5

// This is the minimum version currently supported.  Those
// applications that require basic RSA crypto functionality
// and file operations should use this version
#define CARD_DATA_VERSION_FOUR  4

// For those apps, that want the maximum version available, use
// CARD_DATA_CURRENT_VERSION.  Otherwise applications should
// target a specific version that includes the functionality
// that they require.
#define CARD_DATA_CURRENT_VERSION CARD_DATA_VERSION_SIX

typedef struct _CARD_DATA
{
    // These members must be initialized by the CSP/KSP before
    // calling CardAcquireContext.

    DWORD                                   dwVersion;

    PBYTE                                   pbAtr;
    DWORD                                   cbAtr;
    LPWSTR                                  pwszCardName;

    PFN_CSP_ALLOC                           pfnCspAlloc;
    PFN_CSP_REALLOC                         pfnCspReAlloc;
    PFN_CSP_FREE                            pfnCspFree;

    PFN_CSP_CACHE_ADD_FILE                  pfnCspCacheAddFile;
    PFN_CSP_CACHE_LOOKUP_FILE               pfnCspCacheLookupFile;
    PFN_CSP_CACHE_DELETE_FILE               pfnCspCacheDeleteFile;
    PVOID                                   pvCacheContext;

    PFN_CSP_PAD_DATA                        pfnCspPadData;

    SCARDCONTEXT                            hSCardCtx;
    SCARDHANDLE                             hScard;

    // pointer to vendor specific information

    PVOID                                   pvVendorSpecific;

    // These members are initialized by the card module

    PFN_CARD_DELETE_CONTEXT                 pfnCardDeleteContext;
    PFN_CARD_QUERY_CAPABILITIES             pfnCardQueryCapabilities;
    PFN_CARD_DELETE_CONTAINER               pfnCardDeleteContainer;
    PFN_CARD_CREATE_CONTAINER               pfnCardCreateContainer;
    PFN_CARD_GET_CONTAINER_INFO             pfnCardGetContainerInfo;
    PFN_CARD_AUTHENTICATE_PIN               pfnCardAuthenticatePin;
    PFN_CARD_GET_CHALLENGE                  pfnCardGetChallenge;
    PFN_CARD_AUTHENTICATE_CHALLENGE         pfnCardAuthenticateChallenge;
    PFN_CARD_UNBLOCK_PIN                    pfnCardUnblockPin;
    PFN_CARD_CHANGE_AUTHENTICATOR        pfnCardChangeAuthenticator;
    PFN_CARD_DEAUTHENTICATE              pfnCardDeauthenticate;
    PFN_CARD_CREATE_DIRECTORY            pfnCardCreateDirectory;
    PFN_CARD_DELETE_DIRECTORY            pfnCardDeleteDirectory;
    LPVOID                               pvUnused3;
    LPVOID                               pvUnused4;
    PFN_CARD_CREATE_FILE                 pfnCardCreateFile;
    PFN_CARD_READ_FILE                   pfnCardReadFile;
    PFN_CARD_WRITE_FILE                  pfnCardWriteFile;
    PFN_CARD_DELETE_FILE                 pfnCardDeleteFile;
    PFN_CARD_ENUM_FILES                  pfnCardEnumFiles;
    PFN_CARD_GET_FILE_INFO               pfnCardGetFileInfo;
    PFN_CARD_QUERY_FREE_SPACE            pfnCardQueryFreeSpace;
    PFN_CARD_QUERY_KEY_SIZES             pfnCardQueryKeySizes;

    PFN_CARD_SIGN_DATA                   pfnCardSignData;
    PFN_CARD_RSA_DECRYPT                 pfnCardRSADecrypt;
    PFN_CARD_CONSTRUCT_DH_AGREEMENT      pfnCardConstructDHAgreement;

    // New functions in version five.
    PFN_CARD_DERIVE_KEY                  pfnCardDeriveKey;
    PFN_CARD_DESTROY_DH_AGREEMENT        pfnCardDestroyDHAgreement;
    PFN_CSP_GET_DH_AGREEMENT             pfnCspGetDHAgreement;

    // version 6 additions below here
    PFN_CARD_GET_CHALLENGE_EX            pfnCardGetChallengeEx;
    PFN_CARD_AUTHENTICATE_EX             pfnCardAuthenticateEx;
    PFN_CARD_CHANGE_AUTHENTICATOR_EX     pfnCardChangeAuthenticatorEx;
    PFN_CARD_DEAUTHENTICATE_EX           pfnCardDeauthenticateEx;
    PFN_CARD_GET_CONTAINER_PROPERTY      pfnCardGetContainerProperty;
    PFN_CARD_SET_CONTAINER_PROPERTY      pfnCardSetContainerProperty;
    PFN_CARD_GET_PROPERTY                pfnCardGetProperty;
    PFN_CARD_SET_PROPERTY                pfnCardSetProperty;

} CARD_DATA, *PCARD_DATA;

#endif


```