```
1  //====================================================================;
2  //
3  //  CARDMOD.H
4  //
5  //  Abstract:
6  //      This is the header file commonly used for card modules.
7  //
8  //  This source code is only intended as a supplement to existing Microsoft
9  //  documentation.
10 //
11 //  THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
12 //  KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
13 //  IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
14 //  PURPOSE.
15 //
16 //  Copyright (C) Microsoft Corporation.  All Rights Reserved.
17 //
18 //====================================================================;
19 #ifndef __CARDMOD__H__
20 #define __CARDMOD__H__
21
22 #include <windows.h>
23 #include <wincrypt.h>
24 #pragma warning(push)
25 #pragma warning(disable:4201)
26 // Disable error C4201 in public header
27 //  nonstandard extension used : nameless struct/union
28 #include <winscard.h>
29 #pragma warning(pop)
30 #include <specstrings.h>
31
32 // This value should be passed to
33 //
34 //  SCardSetCardTypeProviderName
35 //  SCardGetCardTypeProviderName
36 //
37 // in order to query and set the Card Specific Module to be used
38 // for a given card.
39 #define SCARD_PROVIDER_CARD_MODULE 0x80000001
40
41 typedef struct _CARD_DATA CARD_DATA, *PCARD_DATA;
42
43 //
44 // This define can be used as a return value for queries involving
45 // card data that may be impossible to determine on a given card
46 // OS, such as the number of available card storage bytes.
47 //
48 #define CARD_DATA_VALUE_UNKNOWN                     ((DWORD) -1)
49
```

```
50 //
51 // Well Known Logical Names
52 //
53
54 //
55 // Logical Directory Names
56 //
57
58 // Second-level logical directories
59
60 #define szBASE_CSP_DIR                          "mscp"
61
62 #define szINTERMEDIATE_CERTS_DIR                "mscerts"
63
64 //
65 // Logical File Names
66 //
67 // When requesting (or otherwise referring to) any logical file, the full path
68 // must be used, including when referring to well known files.  For example,
69 // to request the wszCONTAINER_MAP_FILE, the provided name will be
70 // "/mscp/cmapfile".
71 //
72
73 // Well known logical files under Microsoft
74 #define szCACHE_FILE                            "cardcf"
75
76 #define szCARD_IDENTIFIER_FILE                  "cardid"
77
78 // Well known logical files under CSP
79 #define szCONTAINER_MAP_FILE                    "cmapfile"
80 #define szROOT_STORE_FILE                       "msroots"
81
82 //
83 // Well known logical files under User Certs
84 //
85 // The following prefixes are appended with the container index of the
86 // associated key.  For example, the certificate associated with the
87 // Key Exchange key in container index 2 will have the name:
88 //  "/mscp/kxc2"
89 //
90 #define szUSER_SIGNATURE_CERT_PREFIX            "ksc"
91 #define szUSER_KEYEXCHANGE_CERT_PREFIX          "kxc"
92 #define szUSER_SIGNATURE_PRIVATE_KEY_PREFIX     "kss"
93 #define szUSER_SIGNATURE_PUBLIC_KEY_PREFIX      "ksp"
94 #define szUSER_KEYEXCHANGE_PRIVATE_KEY_PREFIX   "kxs"
95 #define szUSER_KEYEXCHANGE_PUBLIC_KEY_PREFIX    "kxp"
96
97 //
98 // Logical Card User Names
```

```
 99 //
100 #define wszCARD_USER_EVERYONE                       L"anonymous"
101 #define wszCARD_USER_USER                           L"user"
102 #define wszCARD_USER_ADMIN                          L"admin"
103
104 // new ecc key specs
105
106 #define AT_ECDSA_P256       3
107 #define AT_ECDSA_P384       4
108 #define AT_ECDSA_P521       5
109 #define AT_ECDHE_P256       6
110 #define AT_ECDHE_P384       7
111 #define AT_ECDHE_P521       8
112
113 //
114 // Type: CARD_CACHE_FILE_FORMAT
115 //
116 // This struct is used as the file format of the cache file,
117 // as stored on the card.
118 //
119
120 #define CARD_CACHE_FILE_CURRENT_VERSION         1
121
122 typedef struct _CARD_CACHE_FILE_FORMAT
123 {
124     BYTE bVersion;
125     BYTE bPinsFreshness;
126
127     WORD wContainersFreshness;
128     WORD wFilesFreshness;
129 } CARD_CACHE_FILE_FORMAT, *PCARD_CACHE_FILE_FORMAT;
130
131 //
132 // Type: CONTAINER_MAP_RECORD
133 //
134 // This structure describes the format of the Base CSP's container map file,
135 // stored on the card.  This is well-known logical file wszCONTAINER_MAP_FILE.
136 // The file consists of zero or more of these records.
137 //
138 #define MAX_CONTAINER_NAME_LEN                  39
139
140 // This flag is set in the CONTAINER_MAP_RECORD bFlags member if the
141 // corresponding container is valid and currently exists on the card.
142 // If the container is deleted, its bFlags field must be cleared.
143 #define CONTAINER_MAP_VALID_CONTAINER           1
144
145 // This flag is set in the CONTAINER_MAP_RECORD bFlags
146 // member if the corresponding container is the default container on the card.
147 #define CONTAINER_MAP_DEFAULT_CONTAINER         2
```

```
148
149 typedef struct _CONTAINER_MAP_RECORD
150 {
151     WCHAR wszGuid [MAX_CONTAINER_NAME_LEN + 1];
152     BYTE bFlags;
153     BYTE bReserved;
154     WORD wSigKeySizeBits;
155     WORD wKeyExchangeKeySizeBits;
156 } CONTAINER_MAP_RECORD, *PCONTAINER_MAP_RECORD;
157
158 //
159 // Converts a card filename string from unicode to ansi
160 //
161 DWORD WINAPI I_CardConvertFileNameToAnsi(
162     IN      PCARD_DATA pCardData,
163     __in    LPWSTR wszUnicodeName,
164     __out   LPSTR *ppszAnsiName);
165
166 // Logical Directory Access Conditions
167 typedef enum
168 {
169     InvalidDirAc = 0,
170
171     // User Read, Write
172     UserCreateDeleteDirAc,
173
174     // Admin Write
175     AdminCreateDeleteDirAc
176
177 } CARD_DIRECTORY_ACCESS_CONDITION;
178
179 // Logical File Access Conditions
180 typedef enum
181 {
182     // Invalid value, chosed to cooincide with common initialization
183     // of memory
184     InvalidAc = 0,
185
186     // Everyone     Read
187     // User         Read, Write
188     //
189     // Example:  A user certificate file.
190     EveryoneReadUserWriteAc,
191
192     // Everyone     None
193     // User         Write, Execute
194     //
195     // Example:  A private key file.
196     UserWriteExecuteAc,
```

```
197
198     // Everyone     Read
199     // Admin        Read, Write
200     //
201     // Example:  The Card Identifier file.
202     EveryoneReadAdminWriteAc,
203
204     // Explicit value to set when it is desired to say that
205     // it is unknown
206     UnknownAc,
207
208     // Everyone No Access
209     // User Read Write
210     //
211     // Example:  A password wallet file.
212
213     UserReadWriteAc,
214     // Everyone/User No Access
215     // Admin Read Write
216     //
217     // Example:  Administration data.
218
219     AdminReadWriteAc
220 } CARD_FILE_ACCESS_CONDITION;
221
222 //
223 // Function: CardAcquireContext
224 //
225 // Purpose: Initialize the CARD_DATA structure which will be used by
226 //          the CSP to interact with a specific card.
227 //
228 typedef DWORD (WINAPI *PFN_CARD_ACQUIRE_CONTEXT)(
229     IN      PCARD_DATA  pCardData,
230     __in    DWORD       dwFlags);
231
232 DWORD
233 WINAPI
234 CardAcquireContext(
235     IN      PCARD_DATA  pCardData,
236     __in    DWORD       dwFlags);
237
238 //
239 // Function: CardDeleteContext
240 //
241 // Purpose: Free resources consumed by the CARD_DATA structure.
242 //
243 typedef DWORD (WINAPI *PFN_CARD_DELETE_CONTEXT)(
244     __inout     PCARD_DATA  pCardData);
245
```

```
246 DWORD
247 WINAPI
248 CardDeleteContext(
249     __inout     PCARD_DATA  pCardData);
250
251 //
252 // Function: CardQueryCapabilities
253 //
254 // Purpose: Query the card module for specific functionality
255 //          provided by this card.
256 //
257 #define CARD_CAPABILITIES_CURRENT_VERSION 1
258
259 typedef struct _CARD_CAPABILITIES
260 {
261     IN OUT DWORD            dwVersion;
262     IN     BOOL             fCertificateCompression;
263     IN     BOOL             fKeyGen;
264 } CARD_CAPABILITIES, *PCARD_CAPABILITIES;
265
266 typedef DWORD (WINAPI *PFN_CARD_QUERY_CAPABILITIES)(
267     __in      PCARD_DATA          pCardData,
268     __in      PCARD_CAPABILITIES  pCardCapabilities);
269
270 DWORD
271 WINAPI
272 CardQueryCapabilities(
273     __in      PCARD_DATA          pCardData,
274     __in      PCARD_CAPABILITIES  pCardCapabilities);
275
276 //
277 // Function: CardDeleteContainer
278 //
279 // Purpose: Delete the specified key container.
280 //
281 typedef DWORD (WINAPI *PFN_CARD_DELETE_CONTAINER)(
282     __in      PCARD_DATA  pCardData,
283     __in      BYTE        bContainerIndex,
284     __in      DWORD       dwReserved);
285
286 DWORD
287 WINAPI
288 CardDeleteContainer(
289     __in      PCARD_DATA  pCardData,
290     __in      BYTE        bContainerIndex,
291     __in      DWORD       dwReserved);
292
293 //
294 // Function: CardCreateContainer
```

```
295 //
296
297 #define CARD_CREATE_CONTAINER_KEY_GEN           1
298 #define CARD_CREATE_CONTAINER_KEY_IMPORT        2
299
300 typedef DWORD (WINAPI *PFN_CARD_CREATE_CONTAINER)(
301     __in      PCARD_DATA  pCardData,
302     __in      BYTE        bContainerIndex,
303     __in      DWORD       dwFlags,
304     __in      DWORD       dwKeySpec,
305     __in      DWORD       dwKeySize,
306     __in      PBYTE       pbKeyData);
307
308 DWORD
309 WINAPI
310 CardCreateContainer(
311     __in      PCARD_DATA  pCardData,
312     __in      BYTE        bContainerIndex,
313     __in      DWORD       dwFlags,
314     __in      DWORD       dwKeySpec,
315     __in      DWORD       dwKeySize,
316     __in      PBYTE       pbKeyData);
317
318 //
319 // Function: CardGetContainerInfo
320 //
321 // Purpose: Query for all public information available about
322 //          the named key container.  This includes the Signature
323 //          and Key Exchange type public keys, if they exist.
324 //
325 //          The pbSigPublicKey and pbKeyExPublicKey buffers contain the
326 //          Signature and Key Exchange public keys, respectively, if they
327 //          exist.  The format of these buffers is a Crypto
328 //          API PUBLICKEYBLOB -
329 //
330 //              BLOBHEADER
331 //              RSAPUBKEY
332 //              modulus
333 //
334 //          In the case of ECC public keys, the pbSigPublicKey will contain
335 //          the ECDSA key and pbKeyExPublicKey will contain the ECDH key if
336 //          they exist. ECC key structure -
337 //
338 //              BCRYPT_ECCKEY_BLOB
339 //              X coord (big endian)
340 //              Y coord (big endian)
341 //
342 #define CONTAINER_INFO_CURRENT_VERSION 1
343
```

```
344 typedef struct _CONTAINER_INFO
345 {
346     IN OUT DWORD dwVersion;
347     IN     DWORD dwReserved;
348
349     OUT    DWORD cbSigPublicKey;
350     OUT    PBYTE pbSigPublicKey;
351
352     OUT    DWORD cbKeyExPublicKey;
353     OUT    PBYTE pbKeyExPublicKey;
354 } CONTAINER_INFO, *PCONTAINER_INFO;
355
356 typedef DWORD (WINAPI *PFN_CARD_GET_CONTAINER_INFO)(
357     __in      PCARD_DATA  pCardData,
358     __in      BYTE        bContainerIndex,
359     __in      DWORD       dwFlags,
360     __in      PCONTAINER_INFO pContainerInfo);
361
362 DWORD
363 WINAPI
364 CardGetContainerInfo(
365     __in      PCARD_DATA  pCardData,
366     __in      BYTE        bContainerIndex,
367     __in      DWORD       dwFlags,
368     __in      PCONTAINER_INFO pContainerInfo);
369
370 //
371 // Function: CardAuthenticatePin
372 //
373 typedef DWORD (WINAPI *PFN_CARD_AUTHENTICATE_PIN)(
374     __in                  PCARD_DATA  pCardData,
375     __in                  LPWSTR      pwszUserId,
376     __in_bcount(cbPin)    PBYTE       pbPin,
377     __in                  DWORD       cbPin,
378     __out_opt             PDWORD pcAttemptsRemaining);
379
380
381 DWORD
382 WINAPI
383 CardAuthenticatePin(
384     __in                  PCARD_DATA  pCardData,
385     __in                  LPWSTR      pwszUserId,
386     __in_bcount(cbPin)    PBYTE       pbPin,
387     __in                  DWORD       cbPin,
388     __out_opt             PDWORD pcAttemptsRemaining);
389
390 //
391 // Function: CardGetChallenge
392 //
```

```
393 typedef DWORD (WINAPI *PFN_CARD_GET_CHALLENGE)(
394     __in                                PCARD_DATA  pCardData,
395     __out_bcount(*pcbChallengeData)     PBYTE       *ppbChallengeData,
396     __out                               PDWORD      pcbChallengeData);
397
398 DWORD
399 WINAPI
400 CardGetChallenge(
401     __in                                  PCARD_DATA pCardData,
402     __deref_out_bcount(*pcbChallengeData) PBYTE     *ppbChallengeData,
403     __out                                 PDWORD     pcbChallengeData);
404
405 //
406 // Function: CardAuthenticateChallenge
407 //
408 typedef DWORD (WINAPI *PFN_CARD_AUTHENTICATE_CHALLENGE)(
409     __in                            PCARD_DATA  pCardData,
410     __in_bcount(cbResponseData)     PBYTE       pbResponseData,
411     __in                            DWORD       cbResponseData,
412     __out_opt                       PDWORD pcAttemptsRemaining);
413
414 DWORD
415 WINAPI
416 CardAuthenticateChallenge(
417     __in                            PCARD_DATA  pCardData,
418     __in_bcount(cbResponseData)     PBYTE       pbResponseData,
419     __in                            DWORD       cbResponseData,
420     __out_opt                       PDWORD pcAttemptsRemaining);
421
422 //
423 // Function: CardUnblockPin
424 //
425 #define CARD_AUTHENTICATE_PIN_CHALLENGE_RESPONSE                1
426 #define CARD_AUTHENTICATE_PIN_PIN                               2
427
428 typedef DWORD (WINAPI *PFN_CARD_UNBLOCK_PIN)(
429     __in                              PCARD_DATA  pCardData,
430     __in                              LPWSTR      pwszUserId,
431     __in_bcount(cbAuthenticationData) PBYTE       pbAuthenticationData,
432     __in                              DWORD       cbAuthenticationData,
433     __in_bcount(cbNewPinData)         PBYTE       pbNewPinData,
434     __in                              DWORD       cbNewPinData,
435     __in                              DWORD       cRetryCount,
436     __in                              DWORD       dwFlags);
437
438 DWORD
439 WINAPI
440 CardUnblockPin(
441     __in                              PCARD_DATA  pCardData,
```

```
442     __in                                LPWSTR      pwszUserId,
443     __in_bcount(cbAuthenticationData)   PBYTE       pbAuthenticationData,
444     __in                                DWORD       cbAuthenticationData,
445     __in_bcount(cbNewPinData)           PBYTE       pbNewPinData,
446     __in                                DWORD       cbNewPinData,
447     __in                                DWORD       cRetryCount,
448     __in                                DWORD       dwFlags);
449
450 //
451 // Function: CardChangeAuthenticator
452 //
453 typedef DWORD (WINAPI *PFN_CARD_CHANGE_AUTHENTICATOR)(
454     __in                                PCARD_DATA  pCardData,
455     __in                                LPWSTR      pwszUserId,
456     __in_bcount(cbCurrentAuthenticator) PBYTE       pbCurrentAuthenticator,
457     __in                                DWORD       cbCurrentAuthenticator,
458     __in_bcount(cbNewAuthenticator)     PBYTE       pbNewAuthenticator,
459     __in                                DWORD       cbNewAuthenticator,
460     __in                                DWORD       cRetryCount,
461     __in                                DWORD       dwFlags,
462     __out_opt                           PDWORD pcAttemptsRemaining);
463
464 DWORD
465 WINAPI
466 CardChangeAuthenticator(
467     __in                                PCARD_DATA  pCardData,
468     __in                                LPWSTR      pwszUserId,
469     __in_bcount(cbCurrentAuthenticator) PBYTE       pbCurrentAuthenticator,
470     __in                                DWORD       cbCurrentAuthenticator,
471     __in_bcount(cbNewAuthenticator)     PBYTE       pbNewAuthenticator,
472     __in                                DWORD       cbNewAuthenticator,
473     __in                                DWORD       cRetryCount,
474     __in                                DWORD       dwFlags,
475     __out_opt                           PDWORD pcAttemptsRemaining);
476
477 //
478 // Function: CardDeauthenticate
479 //
480 // Purpose: De-authenticate the specified logical user name on the card.
481 //
482 // This is an optional API.  If implemented, this API is used instead
483 // of SCARD_RESET_CARD by the Base CSP.  An example scenario is leaving
484 // a transaction in which the card has been authenticated (a Pin has been
485 // successfully presented).
486 //
487 // The pwszUserId parameter will point to a valid well-known User Name (see
488 // above).
489 //
490 // The dwFlags parameter is currently unused and will always be zero.
```

```
491 //
492 // Card modules that choose to not implement this API must set the CARD_DATA
493 // pfnCardDeauthenticate pointer to NULL.
494 //
495 typedef DWORD (WINAPI *PFN_CARD_DEAUTHENTICATE)(
496     __in      PCARD_DATA  pCardData,
497     __in      LPWSTR      pwszUserId,
498     __in      DWORD       dwFlags);
499
500 DWORD
501 WINAPI
502 CardDeauthenticate(
503     __in    PCARD_DATA  pCardData,
504     __in    LPWSTR      pwszUserId,
505     __in    DWORD       dwFlags);
506
507 // Directory Control Group
508
509 //
510 // Function: CardCreateDirectory
511 //
512 // Purpose: Register the specified application name on the card, and apply the
513 //          provided access condition.
514 //
515 // Return Value:
516 //          ERROR_FILE_EXISTS - directory already exists
517 //
518 typedef DWORD (WINAPI *PFN_CARD_CREATE_DIRECTORY)(
519     __in    PCARD_DATA  pCardData,
520     __in    LPSTR       pszDirectoryName,
521     __in    CARD_DIRECTORY_ACCESS_CONDITION AccessCondition);
522
523 DWORD
524 WINAPI
525 CardCreateDirectory(
526     __in    PCARD_DATA  pCardData,
527     __in    LPSTR       pszDirectoryName,
528     __in    CARD_DIRECTORY_ACCESS_CONDITION AccessCondition);
529
530 //
531 // Function: CardDeleteDirectory
532 //
533 // Purpose: Unregister the specified application from the card.
534 //
535 // Return Value:
536 //          SCARD_E_DIR_NOT_FOUND - directory does not exist
537 //          ERROR_DIR_NOT_EMPTY - the directory is not empty
538 //
539 typedef DWORD (WINAPI *PFN_CARD_DELETE_DIRECTORY)(
```

```
540      __in    PCARD_DATA  pCardData,
541      __in    LPSTR       pszDirectoryName);
542
543  DWORD
544  WINAPI
545  CardDeleteDirectory(
546      __in    PCARD_DATA  pCardData,
547      __in    LPSTR       pszDirectoryName);
548
549  // File Control Group
550
551  //
552  // Function: CardCreateFile
553  //
554  typedef DWORD (WINAPI *PFN_CARD_CREATE_FILE)(
555      __in    PCARD_DATA  pCardData,
556      __in    LPSTR       pszDirectoryName,
557      __in    LPSTR       pszFileName,
558      __in    DWORD       cbInitialCreationSize,
559      __in    CARD_FILE_ACCESS_CONDITION AccessCondition);
560
561  DWORD
562  WINAPI
563  CardCreateFile(
564      __in    PCARD_DATA  pCardData,
565      __in    LPSTR       pszDirectoryName,
566      __in    LPSTR       pszFileName,
567      __in    DWORD       cbInitialCreationSize,
568      __in    CARD_FILE_ACCESS_CONDITION AccessCondition);
569
570  //
571  // Function: CardReadFile
572  //
573  // Purpose: Read the specified file from the card.
574  //
575  //          The pbData parameter should be allocated
576  //          by the card module and freed by the CSP.  The card module
577  //          must set the cbData parameter to the size of the returned buffer.
578  //
579  typedef DWORD (WINAPI *PFN_CARD_READ_FILE)(
580      __in                        PCARD_DATA  pCardData,
581      __in                        LPSTR       pszDirectoryName,
582      __in                        LPSTR       pszFileName,
583      __in                        DWORD       dwFlags,
584      __out_bcount(*pcbData)      PBYTE      *ppbData,
585      __out                       PDWORD      pcbData);
586
587  DWORD
588  WINAPI
```

```
589 CardReadFile(
590     __in                            PCARD_DATA  pCardData,
591     __in                            LPSTR       pszDirectoryName,
592     __in                            LPSTR       pszFileName,
593     __in                            DWORD       dwFlags,
594     __deref_out_bcount(*pcbData)    PBYTE      *ppbData,
595     __out                           PDWORD      pcbData);
596
597 //
598 // Function: CardWriteFile
599 //
600 typedef DWORD (WINAPI *PFN_CARD_WRITE_FILE)(
601     __in                    PCARD_DATA  pCardData,
602     __in                    LPSTR       pszDirectoryName,
603     __in                    LPSTR       pszFileName,
604     __in                    DWORD       dwFlags,
605     __in_bcount(cbData)     PBYTE       pbData,
606     __in                    DWORD       cbData);
607
608 DWORD
609 WINAPI
610 CardWriteFile(
611     __in                    PCARD_DATA  pCardData,
612     __in                    LPSTR       pszDirectoryName,
613     __in                    LPSTR       pszFileName,
614     __in                    DWORD       dwFlags,
615     __in_bcount(cbData)     PBYTE       pbData,
616     __in                    DWORD       cbData);
617
618 //
619 // Function: CardDeleteFile
620 //
621 typedef DWORD (WINAPI *PFN_CARD_DELETE_FILE)(
622     __in    PCARD_DATA  pCardData,
623     __in    LPSTR       pszDirectoryName,
624     __in    LPSTR       pszFileName,
625     __in    DWORD       dwFlags);
626
627 DWORD
628 WINAPI
629 CardDeleteFile(
630     __in    PCARD_DATA  pCardData,
631     __in    LPSTR       pszDirectoryName,
632     __in    LPSTR       pszFileName,
633     __in    DWORD       dwFlags);
634
635 //
636 // Function: CardEnumFiles
637 //
```

```
638 // Purpose: Return a multi-string list of the general files
639 //          present on this card.  The multi-string is allocated
640 //          by the card module and must be freed by the CSP.
641 //
642 //  The caller must provide a logical file directory name in the
643 //  pmwszFileNames parameter (see Logical Directory Names, above).
644 //  The logical directory name indicates which group of files will be
645 //  enumerated.
646 //
647 //  The logical directory name is expected to be a static string, so the
648 //  the card module will not free it.  The card module
649 //  will allocate a new buffer in *pmwszFileNames to store the multi-string
650 //  list of enumerated files using pCardData->pfnCspAlloc.
651 //
652 //  If the function fails for any reason, *pmwszFileNames is set to NULL.
653 //
654 typedef DWORD (WINAPI *PFN_CARD_ENUM_FILES)(
655     __in      PCARD_DATA  pCardData,
656     __in      LPSTR       pszDirectoryName,
657     __out_ecount(*pdwcbFileName)
658               LPSTR      *pmszFileNames,
659     __out     LPDWORD     pdwcbFileName,
660     __in      DWORD       dwFlags);
661
662 DWORD
663 WINAPI
664 CardEnumFiles(
665     __in      PCARD_DATA  pCardData,
666     __in      LPSTR       pszDirectoryName,
667     __out_ecount(*pdwcbFileName)
668               LPSTR      *pmszFileNames,
669     __out     LPDWORD     pdwcbFileName,
670     __in      DWORD       dwFlags);
671
672 //
673 // Function: CardGetFileInfo
674 //
675 #define CARD_FILE_INFO_CURRENT_VERSION 1
676
677 typedef struct _CARD_FILE_INFO
678 {
679     IN OUT DWORD dwVersion;
680     OUT    DWORD cbFileSize;
681     OUT    CARD_FILE_ACCESS_CONDITION AccessCondition;
682 } CARD_FILE_INFO, *PCARD_FILE_INFO;
683
684 typedef DWORD (WINAPI *PFN_CARD_GET_FILE_INFO)(
685     __in         PCARD_DATA  pCardData,
686     __in         LPSTR      pszDirectoryName,
```

```
687     __in         LPSTR      pszFileName,
688     __in         PCARD_FILE_INFO pCardFileInfo);
689
690 DWORD
691 WINAPI
692 CardGetFileInfo(
693     __in         PCARD_DATA  pCardData,
694     __in         LPSTR      pszDirectoryName,
695     __in         LPSTR      pszFileName,
696     __in         PCARD_FILE_INFO pCardFileInfo);
697
698 //
699 // Function: CardQueryFreeSpace
700 //
701 #define CARD_FREE_SPACE_INFO_CURRENT_VERSION 1
702
703 typedef struct _CARD_FREE_SPACE_INFO
704 {
705     IN OUT DWORD dwVersion;
706     OUT    DWORD dwBytesAvailable;
707     OUT    DWORD dwKeyContainersAvailable;
708     OUT    DWORD dwMaxKeyContainers;
709
710 } CARD_FREE_SPACE_INFO, *PCARD_FREE_SPACE_INFO;
711
712 typedef DWORD (WINAPI *PFN_CARD_QUERY_FREE_SPACE)(
713     __in      PCARD_DATA  pCardData,
714     __in      DWORD       dwFlags,
715     __in      PCARD_FREE_SPACE_INFO pCardFreeSpaceInfo);
716
717 DWORD
718 WINAPI
719 CardQueryFreeSpace(
720     __in      PCARD_DATA  pCardData,
721     __in      DWORD       dwFlags,
722     __in      PCARD_FREE_SPACE_INFO pCardFreeSpaceInfo);
723
724 //
725 // Function: CardQueryKeySizes
726 //
727 #define CARD_KEY_SIZES_CURRENT_VERSION 1
728
729 typedef struct _CARD_KEY_SIZES
730 {
731     IN OUT  DWORD dwVersion;
732
733     OUT     DWORD dwMinimumBitlen;
734     OUT     DWORD dwDefaultBitlen;
735     OUT     DWORD dwMaximumBitlen;
```

```
736     OUT     DWORD dwIncrementalBitlen;
737
738 } CARD_KEY_SIZES, *PCARD_KEY_SIZES;
739
740 typedef DWORD (WINAPI *PFN_CARD_QUERY_KEY_SIZES)(
741     __in      PCARD_DATA  pCardData,
742     __in      DWORD       dwKeySpec,
743     __in      DWORD       dwFlags,
744     __in      PCARD_KEY_SIZES pKeySizes);
745
746 DWORD
747 WINAPI
748 CardQueryKeySizes(
749     __in      PCARD_DATA  pCardData,
750     __in      DWORD       dwKeySpec,
751     __in      DWORD       dwFlags,
752     __in      PCARD_KEY_SIZES pKeySizes);
753
754 //
755 // Function: CardRSADecrypt
756 //
757 // Purpose: Perform a private key decryption on the supplied data.  The
758 //          card module should assume that pbData is the length of the
759 //          key modulus.
760 //
761 #define CARD_RSA_KEY_DECRYPT_INFO_CURRENT_VERSION 1
762
763 typedef struct _CARD_RSA_DECRYPT_INFO
764 {
765     __in                        DWORD dwVersion;
766     __in                        BYTE bContainerIndex;
767
768     // For RSA operations, this should be AT_SIGNATURE or AT_KEYEXCHANGE.
769     __in                        DWORD dwKeySpec;
770
771     // This is the buffer and length that the caller expects to be decrypted.
772     // For RSA operations, cbData is redundant since the length of the buffer
773     // should always be equal to the length of the key modulus.
774     __out_bcount(cbData)        PBYTE pbData;
775     __out                       DWORD cbData;
776
777 } CARD_RSA_DECRYPT_INFO, *PCARD_RSA_DECRYPT_INFO;
778
779 typedef DWORD (WINAPI *PFN_CARD_RSA_DECRYPT)(
780     __in        PCARD_DATA              pCardData,
781     __inout     PCARD_RSA_DECRYPT_INFO  pInfo);
782
783 DWORD
784 WINAPI
```

```
785 CardRSADecrypt(
786     __in        PCARD_DATA              pCardData,
787     __inout     PCARD_RSA_DECRYPT_INFO  pInfo);
788
789 #define CARD_PADDING_INFO_PRESENT 0x40000000
790 #define CARD_BUFFER_SIZE_ONLY     0x20000000
791 #define CARD_PADDING_NONE         0x00000001
792 #define CARD_PADDING_PKCS1        0x00000002
793 #define CARD_PADDING_PSS          0x00000004
794
795 // CARD_SIGNING_INFO_BASIC_VERSION is provided for thos applications
796 // do not intend to support passing in the pPaddingInfo structure
797 #define CARD_SIGNING_INFO_BASIC_VERSION 1
798
799 //
800 // Function: CardSignData
801 //
802 // Purpose: Sign inupt data using a specified key
803 //
804 #define CARD_SIGNING_INFO_CURRENT_VERSION 2
805 typedef struct _CARD_SIGNING_INFO
806 {
807     IN DWORD  dwVersion;
808
809     IN BYTE   bContainerIndex;
810
811     // See dwKeySpec constants
812     IN DWORD  dwKeySpec;
813
814     // If CARD_BUFFER_SIZE_ONLY flag is present then the card
815     // module should return only the size of the resulting
816     // key in cbSignedData
817     IN DWORD  dwSigningFlags;
818
819     // If the aiHashAlg is non zero, then it specifies the algorithm
820     // to use when padding the data using PKCS
821     IN ALG_ID aiHashAlg;
822
823     // This is the buffer and length that the caller expects to be signed.
824     // Signed version is allocated a buffer and put in cb/pbSignedData.  That ↵
        should
825     // be freed using PFN_CSP_FREE callback.
826     IN PBYTE  pbData;
827     IN DWORD  cbData;
828
829     OUT PBYTE  pbSignedData;
830     OUT DWORD  cbSignedData;
831
832     // The following parameters are new in version 2 of the
```

```
833     // CARD_SIGNING_INFO structure.
834     // If CARD_PADDING_INFO_PRESENT is set in dwSigningFlags then
835     // pPaddingInfo will point to the BCRYPT_PADDING_INFO structure
836     // defined by dwPaddingType.  Currently supported values are
837     // CARD_PADDING_PKCS1, CARD_PADDING_PSS and CARD_PADDING_NONE
838     IN LPVOID pPaddingInfo;
839     IN DWORD  dwPaddingType;
840 } CARD_SIGNING_INFO, *PCARD_SIGNING_INFO;
841
842 typedef DWORD (WINAPI *PFN_CARD_SIGN_DATA)(
843     __in      PCARD_DATA          pCardData,
844     __in      PCARD_SIGNING_INFO  pInfo);
845
846 DWORD
847 WINAPI
848 CardSignData(
849     __in      PCARD_DATA          pCardData,
850     __in      PCARD_SIGNING_INFO  pInfo);
851
852 //
853 // Type: CARD_DH_AGREEMENT_INFO
854 //
855 // CARD_DH_AGREEMENT_INFO version 1 is no longer supported and should
856 // not be implemented
857 //
858
859 #define CARD_DH_AGREEMENT_INFO_VERSION 2
860
861 typedef struct _CARD_DH_AGREEMENT_INFO
862 {
863     IN  DWORD dwVersion;
864     IN  BYTE  bContainerIndex;
865     IN  DWORD dwFlags;
866     IN  DWORD dwPublicKey;
867     IN  PBYTE pbPublicKey;
868     IN  PBYTE pbReserved;
869     IN  DWORD cbReserved;
870
871     OUT BYTE bSecretAgreementIndex;
872 } CARD_DH_AGREEMENT_INFO, *PCARD_DH_AGREEMENT_INFO;
873
874 //
875 // Function:  CardConstructDHAgreement
876 //
877 // Purpose: compute a DH secret agreement from a ECDH key on the card
878 // and the public portion of another ECDH key
879 //
880
881 typedef DWORD (WINAPI *PFN_CARD_CONSTRUCT_DH_AGREEMENT)(
```

```
882     __in     PCARD_DATA pCardData,
883     __in     PCARD_DH_AGREEMENT_INFO pAgreementInfo);
884
885 DWORD WINAPI CardConstructDHAgreement(
886     __in     PCARD_DATA pCardData,
887     __in     PCARD_DH_AGREEMENT_INFO pAgreementInfo);
888
889 //
890 // Type: CARD_DERIVE_KEY_INFO
891 //
892 #define CARD_DERIVE_KEY_VERSION 1
893
894 typedef struct _CARD_DERIVE_KEY
895 {
896     IN  DWORD             dwVersion;
897
898     // If CARD_BUFFER_SIZE_ONLY is passed then the card module
899     // should return only the size of the resulting key in
900     // cbDerivedKey
901     IN  DWORD             dwFlags;
902     IN  LPWSTR            pwszKDF;
903     IN  BYTE              bSecretAgreementIndex;
904
905     IN  PVOID             pParameterList;
906
907     OUT PBYTE             pbDerivedKey;
908     OUT DWORD             cbDerivedKey;
909
910 } CARD_DERIVE_KEY, *PCARD_DERIVE_KEY;
911
912 //
913 // Function:  CardDeriveKey
914 //
915 // Purpose: Generate a dervived session key using a generated agreed
916 // secret and various other parameters.
917 //
918
919 typedef DWORD (WINAPI *PFN_CARD_DERIVE_KEY)(
920     __in     PCARD_DATA pCardData,
921     __in     PCARD_DERIVE_KEY pAgreementInfo);
922
923 DWORD WINAPI CardDeriveKey(
924     __in     PCARD_DATA pCardData,
925     __in     PCARD_DERIVE_KEY pAgreementInfo);
926
927 //
928 // Function:  CardDestroyAgreement
929 //
930 // Purpose: Force a deletion of the DH agreed secret.
```

```
931 //
932
933 typedef DWORD (WINAPI *PFN_CARD_DESTROY_DH_AGREEMENT)(
934     __in PCARD_DATA pCardData,
935     __in BYTE       bSecretAgreementIndex,
936     __in DWORD      dwFlags);
937
938 DWORD WINAPI CardDestroyDHAgreement(
939     __in PCARD_DATA pCardData,
940     __in BYTE       bSecretAgreementIndex,
941     __in DWORD      dwFlags);
942
943 //
944 // Function:  CspGetDHAgreement
945 //
946 // Purpose: The CARD_DERIVE_KEY structure contains a list of parameters
947 // (pParameterList) which might contain a reference to one or more addition
948 // agreed secrets (KDF_NCRYPT_SECRET_HANDLE).  This callback is provided by
949 // the caller of CardDeriveKey and will translate the parameter into the
950 // on card agreed secret handle.
951 //
952
953 typedef DWORD (WINAPI *PFN_CSP_GET_DH_AGREEMENT)(
954     IN  PCARD_DATA          pCardData,
955     IN  PVOID               hSecretAgreement,
956     OUT BYTE*               pbSecretAgreementIndex,
957     IN  DWORD               dwFlags);
958
959 DWORD WINAPI CspGetDHAgreement(
960     __in  PCARD_DATA          pCardData,
961     __in  PVOID               hSecretAgreement,
962     __out BYTE*               pbSecretAgreementIndex,
963     __in  DWORD               dwFlags);
964
965 //
966 // Memory Management Routines
967 //
968 // These routines are supplied to the card module
969 // by the calling CSP.
970 //
971
972 //
973 // Function: PFN_CSP_ALLOC
974 //
975 typedef LPVOID (WINAPI *PFN_CSP_ALLOC)(
976     IN      SIZE_T      Size);
977
978 //
979 // Function: PFN_CSP_REALLOC
```

```
 980 //
 981 typedef LPVOID (WINAPI *PFN_CSP_REALLOC)(
 982     IN      LPVOID      Address,
 983     IN      SIZE_T      Size);
 984
 985 //
 986 // Function: PFN_CSP_FREE
 987 //
 988 // Note: Data allocated for the CSP by the card module must
 989 //       be freed by the CSP.
 990 //
 991 typedef void (WINAPI *PFN_CSP_FREE)(
 992     IN      LPVOID      Address);
 993
 994 //
 995 // Function: PFN_CSP_CACHE_ADD_FILE
 996 //
 997 // A copy of the pbData parameter is added to the cache.
 998 //
 999 typedef DWORD (WINAPI *PFN_CSP_CACHE_ADD_FILE)(
1000     IN      PVOID       pvCacheContext,
1001     IN      LPWSTR      wszTag,
1002     IN      DWORD       dwFlags,
1003     IN      PBYTE       pbData,
1004     IN      DWORD       cbData);
1005
1006 //
1007 // Function: PFN_CSP_CACHE_LOOKUP_FILE
1008 //
1009 // If the cache lookup is successful,
1010 // the caller must free the *ppbData pointer with pfnCspFree.
1011 //
1012 typedef DWORD (WINAPI *PFN_CSP_CACHE_LOOKUP_FILE)(
1013     IN      PVOID       pvCacheContext,
1014     IN      LPWSTR      wszTag,
1015     IN      DWORD       dwFlags,
1016     IN      PBYTE      *ppbData,
1017     IN      PDWORD      pcbData);
1018
1019 //
1020 // Function: PFN_CSP_CACHE_DELETE_FILE
1021 //
1022 // Deletes the specified item from the cache.
1023 //
1024 typedef DWORD (WINAPI *PFN_CSP_CACHE_DELETE_FILE)(
1025     IN      PVOID       pvCacheContext,
1026     IN      LPWSTR      wszTag,
1027     IN      DWORD       dwFlags);
1028
```

```
1029 //
1030 // Function: PFN_CSP_PAD_DATA
1031 //
1032 // Deletes Callback to pad buffer for cyrpto operation.  Used when
1033 // the card does not provide this.
1034 //
1035 typedef DWORD (WINAPI *PFN_CSP_PAD_DATA)(
1036     IN      PCARD_SIGNING_INFO pSigningInfo,
1037     IN      DWORD              cbMaxWidth,
1038     OUT     DWORD*             pcbPaddedBuffer,
1039     OUT     PBYTE*             ppbPaddedBuffer);
1040
1041 // ****************
1042 // PIN SUPPORT
1043 // ****************
1044
1045 //
1046 // There are 8 PINs currently defined in version 6. PIN values 0, 1 and 2 are
1047 // reserved for backwards compatibility, whereas PIN values 3-7 can be used
1048 // as additional PINs to protect key containers.
1049 //
1050
1051 typedef     DWORD                       PIN_ID, *PPIN_ID;
1052 typedef     DWORD                       PIN_SET, *PPIN_SET;
1053
1054 #define     MAX_PINS                    8
1055
1056 #define     ROLE_EVERYONE               0
1057 #define     ROLE_USER                   1
1058 #define     ROLE_ADMIN                  2
1059
1060 #define     PIN_SET_ALL_ROLES           0xFF
1061 #define     CREATE_PIN_SET(PinId)       (1 << PinId)
1062 #define     SET_PIN(PinSet, PinId)      PinSet |= CREATE_PIN_SET(PinId)
1063 #define     IS_PIN_SET(PinSet, PinId)   (0 != (PinSet & CREATE_PIN_SET
      (PinId)))
1064 #define     CLEAR_PIN(PinSet, PinId)    PinSet &= ~CREATE_PIN_SET(PinId)
1065
1066 #define     PIN_CHANGE_FLAG_UNBLOCK     0x01
1067 #define     PIN_CHANGE_FLAG_CHANGEPIN   0x02
1068
1069 #define     CP_CACHE_MODE_GLOBAL_CACHE  1
1070 #define     CP_CACHE_MODE_SESSION_ONLY  2
1071 #define     CP_CACHE_MODE_NO_CACHE      3
1072
1073 #define     CARD_AUTHENTICATE_GENERATE_SESSION_PIN      0x10000000
1074 #define     CARD_AUTHENTICATE_SESSION_PIN               0x20000000
1075
1076 #define     CARD_PIN_STRENGTH_PLAINTEXT                 0x1
```

```
1077  #define     CARD_PIN_STRENGTH_SESSION_PIN              0x2
1078
1079  #define     CARD_PIN_SILENT_CONTEXT                    0x00000040
1080
1081  typedef enum
1082  {
1083      AlphaNumericPinType = 0,            // Regular PIN
1084      ExternalPinType,                    // Biometric PIN
1085      ChallengeResponsePinType,           // Challenge/Response PIN
1086      EmptyPinType                        // No PIN
1087  } SECRET_TYPE;
1088
1089  typedef enum
1090  {
1091      AuthenticationPin,                  // Authentication PIN
1092      DigitalSignaturePin,                // Digital Signature PIN
1093      EncryptionPin,                      // Encryption PIN
1094      NonRepudiationPin,                  // Non Repudiation PIN
1095      AdministratorPin,                   // Administrator PIN
1096      PrimaryCardPin                      // Primary Card PIN
1097  } SECRET_PURPOSE;
1098
1099  typedef enum
1100  {
1101      PinCacheNormal = 0,
1102      PinCacheTimed,
1103      PinCacheNone
1104  } PIN_CACHE_POLICY_TYPE;
1105
1106  #define      PIN_CACHE_POLICY_CURRENT_VERSION     6
1107
1108  typedef struct _PIN_CACHE_POLICY
1109  {
1110      IN OUT  DWORD                               dwVersion;
1111      OUT     PIN_CACHE_POLICY_TYPE               PinCachePolicyType;
1112      OUT     DWORD                               dwPinCachePolicyInfo;
1113  } PIN_CACHE_POLICY, *PPIN_CACHE_POLICY;
1114
1115  #define      PIN_INFO_CURRENT_VERSION            6
1116
1117  #define      PIN_INFO_REQUIRE_SECURE_ENTRY       1
1118
1119  typedef struct _PIN_INFO
1120  {
1121      IN OUT  DWORD                               dwVersion;
1122      OUT     SECRET_TYPE                         PinType;
1123      OUT     SECRET_PURPOSE                      PinPurpose;
1124      OUT     PIN_SET                             dwChangePermission;
1125      OUT     PIN_SET                             dwUnblockPermission;
```

```
1126      OUT     PIN_CACHE_POLICY                        PinCachePolicy;
1127      OUT     DWORD                                   dwFlags;
1128  } PIN_INFO, *PPIN_INFO;
1129
1130  typedef DWORD (WINAPI *PFN_CARD_GET_CHALLENGE_EX)(
1131      __in                                  PCARD_DATA  pCardData,
1132      __in                                  PIN_ID      PinId,
1133      __out_bcount(*pcbChallengeData)       PBYTE       *ppbChallengeData,
1134      __out                                 PDWORD      pcbChallengeData,
1135      __in                                  DWORD       dwFlags);
1136
1137  DWORD
1138  WINAPI
1139  CardGetChallengeEx(
1140      __in                                    PCARD_DATA pCardData,
1141      __in                                    PIN_ID     PinId,
1142      __deref_out_bcount(*pcbChallengeData) PBYTE     *ppbChallengeData,
1143      __out                                   PDWORD     pcbChallengeData,
1144      __in                                    DWORD      dwFlags);
1145
1146  typedef DWORD (WINAPI *PFN_CARD_AUTHENTICATE_EX)(
1147      __in   PCARD_DATA                               pCardData,
1148      __in   PIN_ID                                   PinId,
1149      __in   DWORD                                    dwFlags,
1150      __in_bcount(cbPinData)   PBYTE                  pbPinData,
1151      __in   DWORD                                    cbPinData,
1152      __deref_out_bcount_opt(*pcbSessionPin) PBYTE  *ppbSessionPin,
1153      __out_opt PDWORD                                pcbSessionPin,
1154      __out_opt PDWORD                                pcAttemptsRemaining);
1155
1156  DWORD
1157  WINAPI
1158  CardAuthenticateEx(
1159      __in   PCARD_DATA                               pCardData,
1160      __in   PIN_ID                                   PinId,
1161      __in   DWORD                                    dwFlags,
1162      __in   PBYTE                                    pbPinData,
1163      __in   DWORD                                    cbPinData,
1164      __deref_out_bcount_opt(*pcbSessionPin) PBYTE  *ppbSessionPin,
1165      __out_opt PDWORD                                pcbSessionPin,
1166      __out_opt PDWORD                                pcAttemptsRemaining);
1167
1168  typedef DWORD (WINAPI *PFN_CARD_CHANGE_AUTHENTICATOR_EX)(
1169      __in   PCARD_DATA                               pCardData,
1170      __in   DWORD                                    dwFlags,
1171      __in   PIN_ID                                   dwAuthenticatingPinId,
1172      __in_bcount(cbAuthenticatingPinData) PBYTE    pbAuthenticatingPinData,
1173      __in   DWORD                                    cbAuthenticatingPinData,
1174      __in   PIN_ID                                   dwTargetPinId,
```

```
1175      __in_bcount(cbTargetData)            PBYTE    pbTargetData,
1176      __in    DWORD                                 cbTargetData,
1177      __in    DWORD                                 cRetryCount,
1178      __out_opt PDWORD                              pcAttemptsRemaining);
1179
1180  DWORD WINAPI CardChangeAuthenticatorEx(
1181      __in    PCARD_DATA                            pCardData,
1182      __in    DWORD                                 dwFlags,
1183      __in    PIN_ID                                dwAuthenticatingPinId,
1184      __in_bcount(cbAuthenticatingPinData) PBYTE    pbAuthenticatingPinData,
1185      __in    DWORD                                 cbAuthenticatingPinData,
1186      __in    PIN_ID                                dwTargetPinId,
1187      __in_bcount(cbTargetData)            PBYTE    pbTargetData,
1188      __in    DWORD                                 cbTargetData,
1189      __in    DWORD                                 cRetryCount,
1190      __out_opt PDWORD                              pcAttemptsRemaining);
1191
1192  typedef DWORD (WINAPI *PFN_CARD_DEAUTHENTICATE_EX)(
1193      __in    PCARD_DATA                            pCardData,
1194      __in    PIN_SET                               PinId,
1195      __in    DWORD                                 dwFlags);
1196
1197  DWORD WINAPI CardDeauthenticateEx(
1198      __in    PCARD_DATA                            pCardData,
1199      __in    PIN_SET                               PinId,
1200      __in    DWORD                                 dwFlags);
1201
1202  // ******************
1203  // Container Porperties
1204  // ******************
1205
1206  #define CCP_CONTAINER_INFO             L"Container Info" // Read only
1207  #define CCP_PIN_IDENTIFIER             L"PIN Identifier"
1208
1209  typedef DWORD (WINAPI *PFN_CARD_GET_CONTAINER_PROPERTY)(
1210      __in    PCARD_DATA                                pCardData,
1211      __in    BYTE                                      bContainerIndex,
1212      __in    LPCWSTR                                   wszProperty,
1213      __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE  pbData,
1214      __in    DWORD                                     cbData,
1215      __out   PDWORD                                    pdwDataLen,
1216      __in    DWORD                                     dwFlags);
1217
1218  DWORD WINAPI CardGetContainerProperty(
1219      __in    PCARD_DATA                                pCardData,
1220      __in    BYTE                                      bContainerIndex,
1221      __in    LPCWSTR                                   wszProperty,
1222      __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE  pbData,
1223      __in    DWORD                                     cbData,
```

```
1224      __out   PDWORD                                  pdwDataLen,
1225      __in    DWORD                                   dwFlags);
1226
1227  typedef DWORD (WINAPI *PFN_CARD_SET_CONTAINER_PROPERTY)(
1228      __in    PCARD_DATA                              pCardData,
1229      __in    BYTE                                    bContainerIndex,
1230      __in    LPCWSTR                                 wszProperty,
1231      __in_bcount(cbDataLen)  PBYTE                   pbData,
1232      __in    DWORD                                   cbDataLen,
1233      __in    DWORD                                   dwFlags);
1234
1235  DWORD WINAPI CardSetContainerProperty(
1236      __in    PCARD_DATA                              pCardData,
1237      __in    BYTE                                    bContainerIndex,
1238      __in    LPCWSTR                                 wszProperty,
1239      __in_bcount(cbDataLen)  PBYTE                   pbData,
1240      __in    DWORD                                   cbDataLen,
1241      __in    DWORD                                   dwFlags);
1242
1243  // ********************
1244  // Card Properties
1245  // ********************
1246
1247  #define CP_CARD_FREE_SPACE              L"Free Space"               // Read
        only
1248  #define CP_CARD_CAPABILITIES            L"Capabilities"             // Read
        only
1249  #define CP_CARD_KEYSIZES                L"Key Sizes"                // Read
        only
1250
1251  #define CP_CARD_READ_ONLY               L"Read Only Mode"
1252  #define CP_CARD_CACHE_MODE              L"Cache Mode"
1253  #define CP_SUPPORTS_WIN_X509_ENROLLMENT L"Supports Windows x.509 Enrollment"
1254
1255  #define CP_CARD_GUID                    L"Card Identifier"
1256  #define CP_CARD_SERIAL_NO               L"Card Serial Number"
1257
1258  #define CP_CARD_PIN_INFO                L"PIN Information"
1259  #define CP_CARD_LIST_PINS               L"PIN List"                 // Read
        only
1260  #define CP_CARD_AUTHENTICATED_STATE     L"Authenticated State"      // Read
        only
1261
1262  #define CP_CARD_PIN_STRENGTH_VERIFY     L"PIN Strength Verify"      // Read
        only
1263  #define CP_CARD_PIN_STRENGTH_CHANGE     L"PIN Strength Change"      // Read
        only
1264  #define CP_CARD_PIN_STRENGTH_UNBLOCK    L"PIN Strength Unblock"     // Read
        only
```

```
1265
1266 #define CP_PARENT_WINDOW                L"Parent Window"            // Write
       only
1267 #define CP_PIN_CONTEXT_STRING           L"PIN Context String"       // Write
       only
1268
1269 typedef DWORD (WINAPI *PFN_CARD_GET_PROPERTY)(
1270     __in    PCARD_DATA                                  pCardData,
1271     __in    LPCWSTR                                     wszProperty,
1272     __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE    pbData,
1273     __in    DWORD                                       cbData,
1274     __out PDWORD                                        pdwDataLen,
1275     __in    DWORD                                       dwFlags);
1276
1277 DWORD WINAPI CardGetProperty(
1278     __in    PCARD_DATA                                  pCardData,
1279     __in    LPCWSTR                                     wszProperty,
1280     __out_bcount_part_opt(cbData, *pdwDataLen) PBYTE    pbData,
1281     __in    DWORD                                       cbData,
1282     __out   PDWORD                                      pdwDataLen,
1283     __in    DWORD                                       dwFlags);
1284
1285 typedef DWORD (WINAPI *PFN_CARD_SET_PROPERTY)(
1286     __in    PCARD_DATA                                  pCardData,
1287     __in    LPCWSTR                                     wszProperty,
1288     __in_bcount(cbDataLen)  PBYTE                       pbData,
1289     __in    DWORD                                       cbDataLen,
1290     __in    DWORD                                       dwFlags);
1291
1292 DWORD WINAPI CardSetProperty(
1293     __in    PCARD_DATA                                  pCardData,
1294     __in    LPCWSTR                                     wszProperty,
1295     __in_bcount(cbDataLen)  PBYTE                       pbData,
1296     __in    DWORD                                       cbDataLen,
1297     __in    DWORD                                       dwFlags);
1298
1299 //
1300 // Type: CARD_DATA
1301 //
1302
1303 #define CARD_DATA_VERSION_SIX   6
1304
1305 // This version supports new features such as a designed
1306 // CardSecretAgreement and key derivation functions.  Also
1307 // added is the PKCS#1 2.1 (PSS) padding format.
1308 #define CARD_DATA_VERSION_FIVE  5
1309
1310 // This is the minimum version currently supported.  Those
1311 // applications that require basic RSA crypto functionality
```

```
1312 // and file operations should use this version
1313 #define CARD_DATA_VERSION_FOUR  4
1314
1315 // For those apps, that want the maximum version available, use
1316 // CARD_DATA_CURRENT_VERSION.  Otherwise applications should
1317 // target a specific version that includes the functionality
1318 // that they require.
1319 #define CARD_DATA_CURRENT_VERSION CARD_DATA_VERSION_SIX
1320
1321 typedef struct _CARD_DATA
1322 {
1323     // These members must be initialized by the CSP/KSP before
1324     // calling CardAcquireContext.
1325
1326     DWORD                               dwVersion;
1327
1328     PBYTE                               pbAtr;
1329     DWORD                               cbAtr;
1330     LPWSTR                              pwszCardName;
1331
1332     PFN_CSP_ALLOC                       pfnCspAlloc;
1333     PFN_CSP_REALLOC                     pfnCspReAlloc;
1334     PFN_CSP_FREE                        pfnCspFree;
1335
1336     PFN_CSP_CACHE_ADD_FILE              pfnCspCacheAddFile;
1337     PFN_CSP_CACHE_LOOKUP_FILE           pfnCspCacheLookupFile;
1338     PFN_CSP_CACHE_DELETE_FILE           pfnCspCacheDeleteFile;
1339     PVOID                               pvCacheContext;
1340
1341     PFN_CSP_PAD_DATA                    pfnCspPadData;
1342
1343     SCARDCONTEXT                        hSCardCtx;
1344     SCARDHANDLE                         hScard;
1345
1346     // pointer to vendor specific information
1347
1348     PVOID                               pvVendorSpecific;
1349
1350     // These members are initialized by the card module
1351
1352     PFN_CARD_DELETE_CONTEXT             pfnCardDeleteContext;
1353     PFN_CARD_QUERY_CAPABILITIES         pfnCardQueryCapabilities;
1354     PFN_CARD_DELETE_CONTAINER           pfnCardDeleteContainer;
1355     PFN_CARD_CREATE_CONTAINER           pfnCardCreateContainer;
1356     PFN_CARD_GET_CONTAINER_INFO         pfnCardGetContainerInfo;
1357     PFN_CARD_AUTHENTICATE_PIN           pfnCardAuthenticatePin;
1358     PFN_CARD_GET_CHALLENGE              pfnCardGetChallenge;
1359     PFN_CARD_AUTHENTICATE_CHALLENGE     pfnCardAuthenticateChallenge;
1360     PFN_CARD_UNBLOCK_PIN                pfnCardUnblockPin;
```

```
1361      PFN_CARD_CHANGE_AUTHENTICATOR       pfnCardChangeAuthenticator;
1362      PFN_CARD_DEAUTHENTICATE             pfnCardDeauthenticate;
1363      PFN_CARD_CREATE_DIRECTORY           pfnCardCreateDirectory;
1364      PFN_CARD_DELETE_DIRECTORY           pfnCardDeleteDirectory;
1365      LPVOID                              pvUnused3;
1366      LPVOID                              pvUnused4;
1367      PFN_CARD_CREATE_FILE                pfnCardCreateFile;
1368      PFN_CARD_READ_FILE                  pfnCardReadFile;
1369      PFN_CARD_WRITE_FILE                 pfnCardWriteFile;
1370      PFN_CARD_DELETE_FILE                pfnCardDeleteFile;
1371      PFN_CARD_ENUM_FILES                 pfnCardEnumFiles;
1372      PFN_CARD_GET_FILE_INFO              pfnCardGetFileInfo;
1373      PFN_CARD_QUERY_FREE_SPACE           pfnCardQueryFreeSpace;
1374      PFN_CARD_QUERY_KEY_SIZES            pfnCardQueryKeySizes;
1375
1376      PFN_CARD_SIGN_DATA                  pfnCardSignData;
1377      PFN_CARD_RSA_DECRYPT                pfnCardRSADecrypt;
1378      PFN_CARD_CONSTRUCT_DH_AGREEMENT     pfnCardConstructDHAgreement;
1379
1380      // New functions in version five.
1381      PFN_CARD_DERIVE_KEY                 pfnCardDeriveKey;
1382      PFN_CARD_DESTROY_DH_AGREEMENT       pfnCardDestroyDHAgreement;
1383      PFN_CSP_GET_DH_AGREEMENT            pfnCspGetDHAgreement;
1384
1385      // version 6 additions below here
1386      PFN_CARD_GET_CHALLENGE_EX           pfnCardGetChallengeEx;
1387      PFN_CARD_AUTHENTICATE_EX            pfnCardAuthenticateEx;
1388      PFN_CARD_CHANGE_AUTHENTICATOR_EX    pfnCardChangeAuthenticatorEx;
1389      PFN_CARD_DEAUTHENTICATE_EX          pfnCardDeauthenticateEx;
1390      PFN_CARD_GET_CONTAINER_PROPERTY     pfnCardGetContainerProperty;
1391      PFN_CARD_SET_CONTAINER_PROPERTY     pfnCardSetContainerProperty;
1392      PFN_CARD_GET_PROPERTY               pfnCardGetProperty;
1393      PFN_CARD_SET_PROPERTY               pfnCardSetProperty;
1394
1395  } CARD_DATA, *PCARD_DATA;
1396
1397  #endif
1398
1399
```