```
/*++

Copyright (c) 1996  Microsoft Corporation

Module Name:

    WinSCard

Abstract:

    This header file provides the definitions and symbols necessary for an
    Application or Smart Card Service Provider to access the Smartcard
    Subsystem.

Environment:

    Win32

Notes:

--*/

#ifndef _WINSCARD_H_
#define _WINSCARD_H_

#if defined (_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

#include <wtypes.h>
#include <winioctl.h>
#include "winsmcrd.h"
#ifndef SCARD_S_SUCCESS
#include "SCardErr.h"
#endif
#include <winapifamily.h>

#ifdef __cplusplus
extern "C" {
#endif

#pragma region Desktop Family
#if WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP)

#ifndef _LPCBYTE_DEFINED
#define _LPCBYTE_DEFINED
typedef const BYTE *LPCBYTE;
#endif
#ifndef _LPCVOID_DEFINED
```

```
50  #define _LPCVOID_DEFINED
51  typedef const VOID *LPCVOID;
52  #endif
53
54  #ifndef WINSCARDAPI
55  #define WINSCARDAPI
56  #endif
57  #ifndef WINSCARDDATA
58  #define WINSCARDDATA __declspec(dllimport)
59  #endif
60
61  /* In clr:pure we cannot mark data export with dllimport.
62   * We should add small functions which returns the value of
63   * the global.
64   */
65  #if !defined(_M_CEE_PURE)
66  WINSCARDDATA extern const SCARD_IO_REQUEST
67      g_rgSCardT0Pci,
68      g_rgSCardT1Pci,
69      g_rgSCardRawPci;
70  #define SCARD_PCI_T0  (&g_rgSCardT0Pci)
71  #define SCARD_PCI_T1  (&g_rgSCardT1Pci)
72  #define SCARD_PCI_RAW (&g_rgSCardRawPci)
73  #endif
74
75  //
76  ////////////////////////////////////////////////////////////////////////////
      //
77  //
78  //  Service Manager Access Services
79  //
80  //      The following services are used to manage user and terminal contexts
      for
81  //      Smart Cards.
82  //
83
84  typedef ULONG_PTR SCARDCONTEXT;
85  typedef SCARDCONTEXT *PSCARDCONTEXT, *LPSCARDCONTEXT;
86
87  typedef ULONG_PTR SCARDHANDLE;
88  typedef SCARDHANDLE *PSCARDHANDLE, *LPSCARDHANDLE;
89
90  #define SCARD_AUTOALLOCATE (DWORD)(-1)
91
92  #define SCARD_SCOPE_USER     0  // The context is a user context, and any
93                                  // database operations are performed within
                        the
94                                  // domain of the user.
95  #define SCARD_SCOPE_TERMINAL 1  // The context is that of the current
```

```
    terminal,
 96                                 // and any database operations are performed
 97                                 // within the domain of that terminal.  (The
 98                                 // calling application must have appropriate
 99                                 // access permissions for any database
                    actions.)
100 #define SCARD_SCOPE_SYSTEM    2 // The context is the system context, and any
101                                 // database operations are performed within
                    the
102                                 // domain of the system.  (The calling
103                                 // application must have appropriate access
104                                 // permissions for any database actions.)
105
106 extern WINSCARDAPI LONG WINAPI
107 SCardEstablishContext(
108     _In_  DWORD dwScope,
109     _Reserved_  LPCVOID pvReserved1,
110     _Reserved_  LPCVOID pvReserved2,
111     _Out_ LPSCARDCONTEXT phContext);
112
113 extern WINSCARDAPI LONG WINAPI
114 SCardReleaseContext(
115     _In_      SCARDCONTEXT hContext);
116
117 extern WINSCARDAPI LONG WINAPI
118 SCardIsValidContext(
119     _In_      SCARDCONTEXT hContext);
120
121
122 //
123 ////////////////////////////////////////////////////////////////////////////
      //
124 //
125 //  Smart Card Database Management Services
126 //
127 //      The following services provide for managing the Smart Card Database.
128 //
129
130 #define SCARD_ALL_READERS       TEXT("SCard$AllReaders\000")
131 #define SCARD_DEFAULT_READERS   TEXT("SCard$DefaultReaders\000")
132 #define SCARD_LOCAL_READERS     TEXT("SCard$LocalReaders\000")
133 #define SCARD_SYSTEM_READERS    TEXT("SCard$SystemReaders\000")
134
135 #define SCARD_PROVIDER_PRIMARY  1   // Primary Provider Id
136 #define SCARD_PROVIDER_CSP      2   // Crypto Service Provider Id
137 #define SCARD_PROVIDER_KSP      3   // Key Storage Provider Id
138
139
140 //
```

```
141 // Database Reader routines
142 //
143
144 extern WINSCARDAPI LONG WINAPI
145 SCardListReaderGroupsA(
146     _In_    SCARDCONTEXT hContext,
147     _Out_writes_opt_(*pcchGroups) _Post_ _NullNull_terminated_  LPSTR
      mszGroups,
148     _Inout_ LPDWORD pcchGroups);
149 extern WINSCARDAPI LONG WINAPI
150 SCardListReaderGroupsW(
151     _In_    SCARDCONTEXT hContext,
152     _Out_writes_opt_(*pcchGroups) _Post_ _NullNull_terminated_  LPWSTR
      mszGroups,
153     _Inout_ LPDWORD pcchGroups);
154 #ifdef UNICODE
155 #define SCardListReaderGroups  SCardListReaderGroupsW
156 #else
157 #define SCardListReaderGroups  SCardListReaderGroupsA
158 #endif // !UNICODE
159
160 _Success_(return == SCARD_S_SUCCESS)
161 extern WINSCARDAPI LONG WINAPI
162 SCardListReadersA(
163     _In_     SCARDCONTEXT hContext,
164     _In_opt_ LPCSTR mszGroups,
165     _When_(_Old_(*pcchReaders) == SCARD_AUTOALLOCATE, _At_((LPSTR *)
      mszReaders, _Outptr_result_buffer_maybenull_(*pcchReaders) _At_(*_Curr_,
       _Post_z_ _Post_ _NullNull_terminated_)))
166     _When_(_Old_(*pcchReaders) != SCARD_AUTOALLOCATE, _Out_writes_opt_
      (*pcchReaders) _Post_ _NullNull_terminated_)
167             LPSTR mszReaders,
168     _Inout_  LPDWORD pcchReaders);
169 _Success_(return == SCARD_S_SUCCESS)
170 extern WINSCARDAPI LONG WINAPI
171 SCardListReadersW(
172     _In_     SCARDCONTEXT hContext,
173     _In_opt_ LPCWSTR mszGroups,
174     _When_(_Old_(*pcchReaders) == SCARD_AUTOALLOCATE, _At_((LPWSTR *)
      mszReaders, _Outptr_result_buffer_maybenull_(*pcchReaders) _At_(*_Curr_,
       _Post_z_ _Post_ _NullNull_terminated_)))
175     _When_(_Old_(*pcchReaders) != SCARD_AUTOALLOCATE, _Out_writes_opt_
      (*pcchReaders) _Post_ _NullNull_terminated_)
176             LPWSTR mszReaders,
177     _Inout_  LPDWORD pcchReaders);
178 #ifdef UNICODE
179 #define SCardListReaders  SCardListReadersW
180 #else
181 #define SCardListReaders  SCardListReadersA
```

```
182 #endif // !UNICODE
183
184 _Success_(return == SCARD_S_SUCCESS)
185 extern WINSCARDAPI LONG WINAPI
186 SCardListCardsA(
187     _In_      SCARDCONTEXT hContext,
188     _In_opt_  LPCBYTE pbAtr,
189     _In_reads_opt_(cguidInterfaceCount)  LPCGUID rgquidInterfaces,
190     _In_      DWORD cguidInterfaceCount,
191     _When_(_Old_(*pcchCards) == SCARD_AUTOALLOCATE, _At_((LPSTR *)mszCards,
        _Outptr_result_buffer_maybenull_(*pcchCards) _At_(*_Curr_, _Post_z_)))
192     _When_(_Old_(*pcchCards) != SCARD_AUTOALLOCATE, _Out_writes_opt_z_
        (*pcchCards))
193             CHAR *mszCards,
194     _Inout_   LPDWORD pcchCards);
195 _Success_(return == SCARD_S_SUCCESS)
196 extern WINSCARDAPI LONG WINAPI
197 SCardListCardsW(
198     _In_      SCARDCONTEXT hContext,
199     _In_opt_  LPCBYTE pbAtr,
200     _In_reads_opt_(cguidInterfaceCount)  LPCGUID rgquidInterfaces,
201     _In_      DWORD cguidInterfaceCount,
202     _When_(_Old_(*pcchCards) == SCARD_AUTOALLOCATE, _At_((LPWSTR *)mszCards,
        _Outptr_result_buffer_maybenull_(*pcchCards) _At_(*_Curr_, _Post_z_)))
203     _When_(_Old_(*pcchCards) != SCARD_AUTOALLOCATE, _Out_writes_opt_z_
        (*pcchCards))
204             WCHAR *mszCards,
205     _Inout_   LPDWORD pcchCards);
206 #ifdef UNICODE
207 #define SCardListCards  SCardListCardsW
208 #else
209 #define SCardListCards  SCardListCardsA
210 #endif // !UNICODE
211 //
212 // NOTE:    The routine SCardListCards name differs from the PC/SC definition.
213 //          It should be:
214 //
215 //              extern WINSCARDAPI LONG WINAPI
216 //              SCardListCardTypes(
217 //                  _In_      SCARDCONTEXT hContext,
218 //                  _In_opt_  LPCBYTE pbAtr,
219 //                  _In_opt_  LPCGUID rgquidInterfaces,
220 //                  _In_      DWORD cguidInterfaceCount,
221 //                  _Out_opt_ LPTSTR mszCards,
222 //                  _Inout_   LPDWORD pcchCards);
223 //
224 //          Here's a work-around MACRO:
225 #define SCardListCardTypes SCardListCards
226
```

```
227 extern WINSCARDAPI LONG WINAPI
228 SCardListInterfacesA(
229     _In_     SCARDCONTEXT hContext,
230     _In_     LPCSTR szCard,
231     _Out_    LPGUID pguidInterfaces,
232     _Inout_  LPDWORD pcguidInterfaces);
233 extern WINSCARDAPI LONG WINAPI
234 SCardListInterfacesW(
235     _In_     SCARDCONTEXT hContext,
236     _In_     LPCWSTR szCard,
237     _Out_    LPGUID pguidInterfaces,
238     _Inout_  LPDWORD pcguidInterfaces);
239 #ifdef UNICODE
240 #define SCardListInterfaces  SCardListInterfacesW
241 #else
242 #define SCardListInterfaces  SCardListInterfacesA
243 #endif // !UNICODE
244
245 extern WINSCARDAPI LONG WINAPI
246 SCardGetProviderIdA(
247     _In_     SCARDCONTEXT hContext,
248     _In_     LPCSTR szCard,
249     _Out_    LPGUID pguidProviderId);
250 extern WINSCARDAPI LONG WINAPI
251 SCardGetProviderIdW(
252     _In_     SCARDCONTEXT hContext,
253     _In_     LPCWSTR szCard,
254     _Out_    LPGUID pguidProviderId);
255 #ifdef UNICODE
256 #define SCardGetProviderId  SCardGetProviderIdW
257 #else
258 #define SCardGetProviderId  SCardGetProviderIdA
259 #endif // !UNICODE
260 //
261 // NOTE:    The routine SCardGetProviderId in this implementation uses GUIDs.
262 //          The PC/SC definition uses BYTEs.
263 //
264
265 _Success_(return == SCARD_S_SUCCESS)
266 extern WINSCARDAPI LONG WINAPI
267 SCardGetCardTypeProviderNameA(
268     _In_      SCARDCONTEXT hContext,
269     _In_      LPCSTR szCardName,
270     _In_      DWORD dwProviderId,
271     _When_(_Old_(*pcchProvider) == SCARD_AUTOALLOCATE, _At_((LPSTR *)
        szProvider, _Outptr_result_buffer_all_(*pcchProvider)))
272     _When_(_Old_(*pcchProvider) != SCARD_AUTOALLOCATE, _Out_writes_to_
        (*pcchProvider, *pcchProvider) _Post_z_)
273             CHAR *szProvider,
```

```
274     _Inout_   LPDWORD pcchProvider);
275 _Success_(return == SCARD_S_SUCCESS)
276 extern WINSCARDAPI LONG WINAPI
277 SCardGetCardTypeProviderNameW(
278     _In_      SCARDCONTEXT hContext,
279     _In_      LPCWSTR szCardName,
280     _In_      DWORD dwProviderId,
281     _When_(_Old_(*pcchProvider) == SCARD_AUTOALLOCATE, _At_((LPWSTR *)
          szProvider, _Outptr_result_buffer_all_(*pcchProvider)))
282     _When_(_Old_(*pcchProvider) != SCARD_AUTOALLOCATE, _Out_writes_to_
          (*pcchProvider, *pcchProvider) _Post_z_)
283             WCHAR *szProvider,
284     _Inout_   LPDWORD pcchProvider);
285 #ifdef UNICODE
286 #define SCardGetCardTypeProviderName  SCardGetCardTypeProviderNameW
287 #else
288 #define SCardGetCardTypeProviderName  SCardGetCardTypeProviderNameA
289 #endif // !UNICODE
290 //
291 // NOTE:    This routine is an extension to the PC/SC definitions.
292 //
293
294
295 //
296 // Database Writer routines
297 //
298
299 extern WINSCARDAPI LONG WINAPI
300 SCardIntroduceReaderGroupA(
301     _In_ SCARDCONTEXT hContext,
302     _In_ LPCSTR szGroupName);
303 extern WINSCARDAPI LONG WINAPI
304 SCardIntroduceReaderGroupW(
305     _In_ SCARDCONTEXT hContext,
306     _In_ LPCWSTR szGroupName);
307 #ifdef UNICODE
308 #define SCardIntroduceReaderGroup  SCardIntroduceReaderGroupW
309 #else
310 #define SCardIntroduceReaderGroup  SCardIntroduceReaderGroupA
311 #endif // !UNICODE
312
313 extern WINSCARDAPI LONG WINAPI
314 SCardForgetReaderGroupA(
315     _In_ SCARDCONTEXT hContext,
316     _In_ LPCSTR szGroupName);
317 extern WINSCARDAPI LONG WINAPI
318 SCardForgetReaderGroupW(
319     _In_ SCARDCONTEXT hContext,
320     _In_ LPCWSTR szGroupName);
```

```
321 #ifdef UNICODE
322 #define SCardForgetReaderGroup  SCardForgetReaderGroupW
323 #else
324 #define SCardForgetReaderGroup  SCardForgetReaderGroupA
325 #endif // !UNICODE
326
327 extern WINSCARDAPI LONG WINAPI
328 SCardIntroduceReaderA(
329     _In_ SCARDCONTEXT hContext,
330     _In_ LPCSTR szReaderName,
331     _In_ LPCSTR szDeviceName);
332 extern WINSCARDAPI LONG WINAPI
333 SCardIntroduceReaderW(
334     _In_ SCARDCONTEXT hContext,
335     _In_ LPCWSTR szReaderName,
336     _In_ LPCWSTR szDeviceName);
337 #ifdef UNICODE
338 #define SCardIntroduceReader  SCardIntroduceReaderW
339 #else
340 #define SCardIntroduceReader  SCardIntroduceReaderA
341 #endif // !UNICODE
342
343 extern WINSCARDAPI LONG WINAPI
344 SCardForgetReaderA(
345     _In_ SCARDCONTEXT hContext,
346     _In_ LPCSTR szReaderName);
347 extern WINSCARDAPI LONG WINAPI
348 SCardForgetReaderW(
349     _In_ SCARDCONTEXT hContext,
350     _In_ LPCWSTR szReaderName);
351 #ifdef UNICODE
352 #define SCardForgetReader  SCardForgetReaderW
353 #else
354 #define SCardForgetReader  SCardForgetReaderA
355 #endif // !UNICODE
356
357 extern WINSCARDAPI LONG WINAPI
358 SCardAddReaderToGroupA(
359     _In_ SCARDCONTEXT hContext,
360     _In_ LPCSTR szReaderName,
361     _In_ LPCSTR szGroupName);
362 extern WINSCARDAPI LONG WINAPI
363 SCardAddReaderToGroupW(
364     _In_ SCARDCONTEXT hContext,
365     _In_ LPCWSTR szReaderName,
366     _In_ LPCWSTR szGroupName);
367 #ifdef UNICODE
368 #define SCardAddReaderToGroup  SCardAddReaderToGroupW
369 #else
```

```
370 #define SCardAddReaderToGroup  SCardAddReaderToGroupA
371 #endif // !UNICODE
372
373 extern WINSCARDAPI LONG WINAPI
374 SCardRemoveReaderFromGroupA(
375     _In_ SCARDCONTEXT hContext,
376     _In_ LPCSTR szReaderName,
377     _In_ LPCSTR szGroupName);
378 extern WINSCARDAPI LONG WINAPI
379 SCardRemoveReaderFromGroupW(
380     _In_ SCARDCONTEXT hContext,
381     _In_ LPCWSTR szReaderName,
382     _In_ LPCWSTR szGroupName);
383 #ifdef UNICODE
384 #define SCardRemoveReaderFromGroup  SCardRemoveReaderFromGroupW
385 #else
386 #define SCardRemoveReaderFromGroup  SCardRemoveReaderFromGroupA
387 #endif // !UNICODE
388
389 extern WINSCARDAPI LONG WINAPI
390 SCardIntroduceCardTypeA(
391     _In_     SCARDCONTEXT hContext,
392     _In_     LPCSTR szCardName,
393     _In_opt_ LPCGUID pguidPrimaryProvider,
394     _In_opt_ LPCGUID rgguidInterfaces,
395     _In_     DWORD dwInterfaceCount,
396     _In_     LPCBYTE pbAtr,
397     _In_     LPCBYTE pbAtrMask,
398     _In_     DWORD cbAtrLen);
399 extern WINSCARDAPI LONG WINAPI
400 SCardIntroduceCardTypeW(
401     _In_     SCARDCONTEXT hContext,
402     _In_     LPCWSTR szCardName,
403     _In_opt_ LPCGUID pguidPrimaryProvider,
404     _In_opt_ LPCGUID rgguidInterfaces,
405     _In_     DWORD dwInterfaceCount,
406     _In_     LPCBYTE pbAtr,
407     _In_     LPCBYTE pbAtrMask,
408     _In_     DWORD cbAtrLen);
409 #ifdef UNICODE
410 #define SCardIntroduceCardType  SCardIntroduceCardTypeW
411 #else
412 #define SCardIntroduceCardType  SCardIntroduceCardTypeA
413 #endif // !UNICODE
414 //
415 // NOTE:    The routine SCardIntroduceCardType's parameters' order differs
      from
416 //          the PC/SC definition.  It should be:
417 //
```

```
418 //              extern WINSCARDAPI LONG WINAPI
419 //              SCardIntroduceCardType(
420 //                  _In_     SCARDCONTEXT hContext,
421 //                  _In_     LPCTSTR szCardName,
422 //                  _In_     LPCBYTE pbAtr,
423 //                  _In_     LPCBYTE pbAtrMask,
424 //                  _In_     DWORD cbAtrLen,
425 //                  _In_opt_ LPCGUID pguidPrimaryProvider,
426 //                  _In_opt_ LPCGUID rgguidInterfaces,
427 //                  _In_     DWORD dwInterfaceCount);
428 //
429 //          Here's a work-around MACRO:
430 #define PCSCardIntroduceCardType(hContext, szCardName, pbAtr, pbAtrMask,       ↵
      cbAtrLen, pguidPrimaryProvider, rgguidInterfaces, dwInterfaceCount) \
431             SCardIntroduceCardType(hContext, szCardName, pguidPrimaryProvider,  ↵
                rgguidInterfaces, dwInterfaceCount, pbAtr, pbAtrMask, cbAtrLen)
432
433 extern WINSCARDAPI LONG WINAPI
434 SCardSetCardTypeProviderNameA(
435     _In_ SCARDCONTEXT hContext,
436     _In_ LPCSTR szCardName,
437     _In_ DWORD dwProviderId,
438     _In_ LPCSTR szProvider);
439 extern WINSCARDAPI LONG WINAPI
440 SCardSetCardTypeProviderNameW(
441     _In_ SCARDCONTEXT hContext,
442     _In_ LPCWSTR szCardName,
443     _In_ DWORD dwProviderId,
444     _In_ LPCWSTR szProvider);
445 #ifdef UNICODE
446 #define SCardSetCardTypeProviderName  SCardSetCardTypeProviderNameW
447 #else
448 #define SCardSetCardTypeProviderName  SCardSetCardTypeProviderNameA
449 #endif // !UNICODE
450 //
451 // NOTE:    This routine is an extention to the PC/SC specifications.
452 //
453
454 extern WINSCARDAPI LONG WINAPI
455 SCardForgetCardTypeA(
456     _In_ SCARDCONTEXT hContext,
457     _In_ LPCSTR szCardName);
458 extern WINSCARDAPI LONG WINAPI
459 SCardForgetCardTypeW(
460     _In_ SCARDCONTEXT hContext,
461     _In_ LPCWSTR szCardName);
462 #ifdef UNICODE
463 #define SCardForgetCardType  SCardForgetCardTypeW
464 #else
```

```
465 #define SCardForgetCardType  SCardForgetCardTypeA
466 #endif // !UNICODE
467
468
469 //
470 ////////////////////////////////////////////////////////////////////////////
    //
471 //
472 //  Service Manager Support Routines
473 //
474 //      The following services are supplied to simplify the use of the Service
475 //      Manager API.
476 //
477
478 extern WINSCARDAPI LONG WINAPI
479 SCardFreeMemory(
480     _In_ SCARDCONTEXT hContext,
481     _In_ LPCVOID pvMem);
482
483 #if (NTDDI_VERSION >= NTDDI_WINXP)
484 extern WINSCARDAPI HANDLE WINAPI
485 SCardAccessStartedEvent(void);
486
487 extern WINSCARDAPI void WINAPI
488 SCardReleaseStartedEvent(void);
489 #endif // (NTDDI_VERSION >= NTDDI_WINXP)
490
491 //
492 ////////////////////////////////////////////////////////////////////////////
    //
493 //
494 //  Reader Services
495 //
496 //      The following services supply means for tracking cards within readers.
497 //
498
499 typedef struct {
500     LPCSTR      szReader;       // reader name
501     LPVOID      pvUserData;     // user defined data
502     DWORD       dwCurrentState; // current state of reader at time of call
503     DWORD       dwEventState;   // state of reader after state change
504     DWORD       cbAtr;          // Number of bytes in the returned ATR.
505     BYTE        rgbAtr[36];     // Atr of inserted card, (extra alignment
        bytes)
506 } SCARD_READERSTATEA, *PSCARD_READERSTATEA, *LPSCARD_READERSTATEA;
507 typedef struct {
508     LPCWSTR     szReader;       // reader name
509     LPVOID      pvUserData;     // user defined data
510     DWORD       dwCurrentState; // current state of reader at time of call
```

```
511     DWORD       dwEventState;   // state of reader after state change
512     DWORD       cbAtr;          // Number of bytes in the returned ATR.
513     BYTE        rgbAtr[36];     // Atr of inserted card, (extra alignment
          bytes)
514 } SCARD_READERSTATEW, *PSCARD_READERSTATEW, *LPSCARD_READERSTATEW;
515 #ifdef UNICODE
516 typedef SCARD_READERSTATEW SCARD_READERSTATE;
517 typedef PSCARD_READERSTATEW PSCARD_READERSTATE;
518 typedef LPSCARD_READERSTATEW LPSCARD_READERSTATE;
519 #else
520 typedef SCARD_READERSTATEA SCARD_READERSTATE;
521 typedef PSCARD_READERSTATEA PSCARD_READERSTATE;
522 typedef LPSCARD_READERSTATEA LPSCARD_READERSTATE;
523 #endif // UNICODE
524
525 // Backwards compatibility macros
526 #define SCARD_READERSTATE_A SCARD_READERSTATEA
527 #define SCARD_READERSTATE_W SCARD_READERSTATEW
528 #define PSCARD_READERSTATE_A PSCARD_READERSTATEA
529 #define PSCARD_READERSTATE_W PSCARD_READERSTATEW
530 #define LPSCARD_READERSTATE_A LPSCARD_READERSTATEA
531 #define LPSCARD_READERSTATE_W LPSCARD_READERSTATEW
532
533 #define SCARD_STATE_UNAWARE     0x00000000  // The application is unaware of
      the
534                                             // current state, and would like
                    to
535                                             // know.  The use of this value
536                                             // results in an immediate return
537                                             // from state transition
                    monitoring
538                                             // services.  This is represented
                    by
539                                             // all bits set to zero.
540 #define SCARD_STATE_IGNORE      0x00000001  // The application requested that
541                                             // this reader be ignored.  No
                    other
542                                             // bits will be set.
543 #define SCARD_STATE_CHANGED     0x00000002  // This implies that there is a
544                                             // difference between the state
545                                             // believed by the application,
                    and
546                                             // the state known by the Service
547                                             // Manager.  When this bit is set,
548                                             // the application may assume a
549                                             // significant state change has
550                                             // occurred on this reader.
551 #define SCARD_STATE_UNKNOWN     0x00000004  // This implies that the given
552                                             // reader name is not recognized
```

```
                    by
553                                         // the Service Manager.  If this  ↵
                    bit
554                                         // is set, then                   ↵
                    SCARD_STATE_CHANGED
555                                         // and SCARD_STATE_IGNORE will    ↵
                    also
556                                         // be set.
557 #define SCARD_STATE_UNAVAILABLE 0x00000008  // This implies that the actual
558                                         // state of this reader is not
559                                         // available.  If this bit is set,
560                                         // then all the following bits are
561                                         // clear.
562 #define SCARD_STATE_EMPTY       0x00000010  // This implies that there is not
563                                         // card in the reader.  If this   ↵
                    bit
564                                         // is set, all the following bits
565                                         // will be clear.
566 #define SCARD_STATE_PRESENT     0x00000020  // This implies that there is a   ↵
      card
567                                         // in the reader.
568 #define SCARD_STATE_ATRMATCH    0x00000040  // This implies that there is a   ↵
      card
569                                         // in the reader with an ATR
570                                         // matching one of the target     ↵
                    cards.
571                                         // If this bit is set,
572                                         // SCARD_STATE_PRESENT will also  ↵
                    be
573                                         // set.  This bit is only returned
574                                         // on the SCardLocateCard()       ↵
                    service.
575 #define SCARD_STATE_EXCLUSIVE   0x00000080  // This implies that the card in  ↵
      the
576                                         // reader is allocated for        ↵
                    exclusive
577                                         // use by another application.  If
578                                         // this bit is set,
579                                         // SCARD_STATE_PRESENT will also  ↵
                    be
580                                         // set.
581 #define SCARD_STATE_INUSE       0x00000100  // This implies that the card in  ↵
      the
582                                         // reader is in use by one or more
583                                         // other applications, but may be
584                                         // connected to in shared mode.   ↵
                    If
585                                         // this bit is set,
586                                         // SCARD_STATE_PRESENT will also  ↵
```

```
                    be
587                                             // set.
588 #define SCARD_STATE_MUTE        0x00000200  // This implies that the card in ↵
      the
589                                             // reader is unresponsive or not
590                                             // supported by the reader or
591                                             // software.
592 #define SCARD_STATE_UNPOWERED   0x00000400  // This implies that the card in ↵
      the
593                                             // reader has not been powered up.
594
595 extern WINSCARDAPI LONG WINAPI
596 SCardLocateCardsA(
597     _In_    SCARDCONTEXT hContext,
598     _In_    LPCSTR mszCards,
599     _Inout_ LPSCARD_READERSTATEA rgReaderStates,
600     _In_    DWORD cReaders);
601 extern WINSCARDAPI LONG WINAPI
602 SCardLocateCardsW(
603     _In_    SCARDCONTEXT hContext,
604     _In_    LPCWSTR mszCards,
605     _Inout_ LPSCARD_READERSTATEW rgReaderStates,
606     _In_    DWORD cReaders);
607 #ifdef UNICODE
608 #define SCardLocateCards  SCardLocateCardsW
609 #else
610 #define SCardLocateCards  SCardLocateCardsA
611 #endif // !UNICODE
612
613 #if (NTDDI_VERSION >= NTDDI_WINXP)
614 typedef struct _SCARD_ATRMASK {
615     DWORD       cbAtr;          // Number of bytes in the ATR and the mask.
616     BYTE        rgbAtr[36];     // Atr of card (extra alignment bytes)
617     BYTE        rgbMask[36];    // Mask for the Atr (extra alignment bytes)
618 } SCARD_ATRMASK, *PSCARD_ATRMASK, *LPSCARD_ATRMASK;
619
620
621 extern WINSCARDAPI LONG WINAPI
622 SCardLocateCardsByATRA(
623     _In_    SCARDCONTEXT hContext,
624     _In_    LPSCARD_ATRMASK rgAtrMasks,
625     _In_    DWORD cAtrs,
626     _Inout_ LPSCARD_READERSTATEA rgReaderStates,
627     _In_    DWORD cReaders);
628 extern WINSCARDAPI LONG WINAPI
629 SCardLocateCardsByATRW(
630     _In_    SCARDCONTEXT hContext,
631     _In_    LPSCARD_ATRMASK rgAtrMasks,
632     _In_    DWORD cAtrs,
```

```
633     _Inout_ LPSCARD_READERSTATEW rgReaderStates,
634     _In_    DWORD cReaders);
635 #ifdef UNICODE
636 #define SCardLocateCardsByATR  SCardLocateCardsByATRW
637 #else
638 #define SCardLocateCardsByATR  SCardLocateCardsByATRA
639 #endif // !UNICODE
640 #endif // (NTDDI_VERSION >= NTDDI_WINXP)
641
642 extern WINSCARDAPI LONG WINAPI
643 SCardGetStatusChangeA(
644     _In_    SCARDCONTEXT hContext,
645     _In_    DWORD dwTimeout,
646     _Inout_ LPSCARD_READERSTATEA rgReaderStates,
647     _In_    DWORD cReaders);
648 extern WINSCARDAPI LONG WINAPI
649 SCardGetStatusChangeW(
650     _In_    SCARDCONTEXT hContext,
651     _In_    DWORD dwTimeout,
652     _Inout_ LPSCARD_READERSTATEW rgReaderStates,
653     _In_    DWORD cReaders);
654 #ifdef UNICODE
655 #define SCardGetStatusChange  SCardGetStatusChangeW
656 #else
657 #define SCardGetStatusChange  SCardGetStatusChangeA
658 #endif // !UNICODE
659
660 extern WINSCARDAPI LONG WINAPI
661 SCardCancel(
662     _In_    SCARDCONTEXT hContext);
663
664
665 //
666 ////////////////////////////////////////////////////////////////////////////
      //
667 //
668 //  Card/Reader Communication Services
669 //
670 //      The following services provide means for communication with the card.
671 //
672
673 #define SCARD_SHARE_EXCLUSIVE 1 // This application is not willing to share
      this
674                                 // card with other applications.
675 #define SCARD_SHARE_SHARED    2 // This application is willing to share this
676                                 // card with other applications.
677 #define SCARD_SHARE_DIRECT    3 // This application demands direct control of
678                                 // the reader, so it is not available to other
679                                 // applications.
```

```
680
681 #define SCARD_LEAVE_CARD      0 // Don't do anything special on close
682 #define SCARD_RESET_CARD      1 // Reset the card on close
683 #define SCARD_UNPOWER_CARD    2 // Power down the card on close
684 #define SCARD_EJECT_CARD      3 // Eject the card on close
685
686 extern WINSCARDAPI LONG WINAPI
687 SCardConnectA(
688     _In_    SCARDCONTEXT hContext,
689     _In_    LPCSTR szReader,
690     _In_    DWORD dwShareMode,
691     _In_    DWORD dwPreferredProtocols,
692     _Out_   LPSCARDHANDLE phCard,
693     _Out_   LPDWORD pdwActiveProtocol);
694 extern WINSCARDAPI LONG WINAPI
695 SCardConnectW(
696     _In_    SCARDCONTEXT hContext,
697     _In_    LPCWSTR szReader,
698     _In_    DWORD dwShareMode,
699     _In_    DWORD dwPreferredProtocols,
700     _Out_   LPSCARDHANDLE phCard,
701     _Out_   LPDWORD pdwActiveProtocol);
702 #ifdef UNICODE
703 #define SCardConnect  SCardConnectW
704 #else
705 #define SCardConnect  SCardConnectA
706 #endif // !UNICODE
707
708 extern WINSCARDAPI LONG WINAPI
709 SCardReconnect(
710     _In_      SCARDHANDLE hCard,
711     _In_      DWORD dwShareMode,
712     _In_      DWORD dwPreferredProtocols,
713     _In_      DWORD dwInitialization,
714     _Out_opt_ LPDWORD pdwActiveProtocol);
715
716 extern WINSCARDAPI LONG WINAPI
717 SCardDisconnect(
718     _In_    SCARDHANDLE hCard,
719     _In_    DWORD dwDisposition);
720
721 extern WINSCARDAPI LONG WINAPI
722 SCardBeginTransaction(
723     _In_    SCARDHANDLE hCard);
724
725 extern WINSCARDAPI LONG WINAPI
726 SCardEndTransaction(
727     _In_    SCARDHANDLE hCard,
728     _In_    DWORD dwDisposition);
```

```
729
730 extern WINSCARDAPI LONG WINAPI
731 SCardCancelTransaction(
732     _In_    SCARDHANDLE hCard);
733 //
734 // NOTE:    This call corresponds to the PC/SC SCARDCOMM::Cancel routine,
735 //          terminating a blocked SCardBeginTransaction service.
736 //
737
738
739 extern WINSCARDAPI LONG WINAPI
740 SCardState(
741     _In_    SCARDHANDLE hCard,
742     _Out_   LPDWORD pdwState,
743     _Out_   LPDWORD pdwProtocol,
744     _Out_writes_bytes_(*pcbAtrLen)   LPBYTE pbAtr,
745     _Inout_ LPDWORD pcbAtrLen);
746 //
747 // NOTE:    SCardState is an obsolete routine.  PC/SC has replaced it with
748 //          SCardStatus.
749 //
750
751 extern WINSCARDAPI LONG WINAPI
752 SCardStatusA(
753     _In_        SCARDHANDLE hCard,
754     _When_(_Old_(*pcchReaderLen) == SCARD_AUTOALLOCATE, _At_((LPSTR *)
      mszReaderNames, _Outptr_result_buffer_maybenull_(*pcchReaderLen) _At_
      (*_Curr_, _Post_z_ _Post_ _NullNull_terminated_)))
755     _When_(_Old_(*pcchReaderLen) != SCARD_AUTOALLOCATE, _Out_writes_opt_
      (*pcchReaderLen) _Post_ _NullNull_terminated_)
756                 LPSTR mszReaderNames,
757     _Inout_opt_ LPDWORD pcchReaderLen,
758     _Out_opt_   LPDWORD pdwState,
759     _Out_opt_   LPDWORD pdwProtocol,
760     _When_(_Old_(*pcbAtrLen) == SCARD_AUTOALLOCATE, _At_((LPBYTE *)pbAtr,
      _Outptr_result_buffer_maybenull_(*pcbAtrLen) _At_(*_Curr_, _Post_
      _NullNull_terminated_)))
761     _When_(_Old_(*pcbAtrLen) != SCARD_AUTOALLOCATE, _Out_writes_opt_
      (*pcbAtrLen) _Post_ _NullNull_terminated_)
762                 LPBYTE pbAtr,
763     _Inout_opt_ LPDWORD pcbAtrLen);
764 extern WINSCARDAPI LONG WINAPI
765 SCardStatusW(
766     _In_        SCARDHANDLE hCard,
767     _When_(_Old_(*pcchReaderLen) == SCARD_AUTOALLOCATE, _At_((LPWSTR *)
      mszReaderNames, _Outptr_result_buffer_maybenull_(*pcchReaderLen) _At_
      (*_Curr_, _Post_z_ _Post_ _NullNull_terminated_)))
768     _When_(_Old_(*pcchReaderLen) != SCARD_AUTOALLOCATE, _Out_writes_opt_
      (*pcchReaderLen) _Post_ _NullNull_terminated_)
```

```
769                 LPWSTR mszReaderNames,
770     _Inout_opt_ LPDWORD pcchReaderLen,
771     _Out_opt_   LPDWORD pdwState,
772     _Out_opt_   LPDWORD pdwProtocol,
773     _When_(_Old_(*pcbAtrLen) == SCARD_AUTOALLOCATE, _At_((LPBYTE *)pbAtr,
          _Outptr_result_buffer_maybenull_(*pcbAtrLen) _At_(*_Curr_, _Post_
          _NullNull_terminated_)))
774     _When_(_Old_(*pcbAtrLen) != SCARD_AUTOALLOCATE, _Out_writes_opt_
          (*pcbAtrLen) _Post_ _NullNull_terminated_)
775                 LPBYTE pbAtr,
776     _Inout_opt_ LPDWORD pcbAtrLen);
777 #ifdef UNICODE
778 #define SCardStatus  SCardStatusW
779 #else
780 #define SCardStatus  SCardStatusA
781 #endif // !UNICODE
782
783 extern WINSCARDAPI LONG WINAPI
784 SCardTransmit(
785     _In_        SCARDHANDLE hCard,
786     _In_        LPCSCARD_IO_REQUEST pioSendPci,
787     _In_reads_bytes_(cbSendLength) LPCBYTE pbSendBuffer,
788     _In_        DWORD cbSendLength,
789     _Inout_opt_ LPSCARD_IO_REQUEST pioRecvPci,
790     _Out_writes_bytes_(*pcbRecvLength) LPBYTE pbRecvBuffer,
791     _Inout_     LPDWORD pcbRecvLength);
792
793 #if (NTDDI_VERSION >= NTDDI_VISTA)
794 extern WINSCARDAPI LONG WINAPI
795 SCardGetTransmitCount(
796     _In_ SCARDHANDLE hCard,
797     _Out_ LPDWORD pcTransmitCount);
798 #endif // (NTDDI_VERSION >= NTDDI_VISTA)
799
800 //
801 ////////////////////////////////////////////////////////////////////////////
      //
802 //
803 //  Reader Control Routines
804 //
805 //      The following services provide for direct, low-level manipulation of
      the
806 //      reader by the calling application allowing it control over the
807 //      attributes of the communications with the card.
808 //
809
810 extern WINSCARDAPI LONG WINAPI
811 SCardControl(
812     _In_    SCARDHANDLE hCard,
```

```
813     _In_    DWORD dwControlCode,
814     _In_reads_bytes_(cbInBufferSize) LPCVOID lpInBuffer,
815     _In_    DWORD cbInBufferSize,
816     _Out_writes_bytes_(cbOutBufferSize) LPVOID lpOutBuffer,
817     _In_    DWORD cbOutBufferSize,
818     _Out_   LPDWORD lpBytesReturned);
819
820 extern WINSCARDAPI LONG WINAPI
821 SCardGetAttrib(
822     _In_    SCARDHANDLE hCard,
823     _In_    DWORD dwAttrId,
824     _Out_writes_bytes_opt_(*pcbAttrLen) LPBYTE pbAttr,
825     _Inout_ LPDWORD pcbAttrLen);
826 //
827 // NOTE:    The routine SCardGetAttrib's name differs from the PC/SC
      definition.
828 //          It should be:
829 //
830 //              extern WINSCARDAPI LONG WINAPI
831 //              SCardGetReaderCapabilities(
832 //                  _In_    SCARDHANDLE hCard,
833 //                  _In_    DWORD dwTag,
834 //                  _Out_   LPBYTE pbAttr,
835 //                  _Inout_ LPDWORD pcbAttrLen);
836 //
837 //          Here's a work-around MACRO:
838 #define SCardGetReaderCapabilities SCardGetAttrib
839
840 extern WINSCARDAPI LONG WINAPI
841 SCardSetAttrib(
842     _In_ SCARDHANDLE hCard,
843     _In_ DWORD dwAttrId,
844     _In_reads_bytes_(cbAttrLen) LPCBYTE pbAttr,
845     _In_ DWORD cbAttrLen);
846 //
847 // NOTE:    The routine SCardSetAttrib's name differs from the PC/SC
      definition.
848 //          It should be:
849 //
850 //              extern WINSCARDAPI LONG WINAPI
851 //              SCardSetReaderCapabilities(
852 //                  _In_    SCARDHANDLE hCard,
853 //                  _In_    DWORD dwTag,
854 //                  _In_    LPCBYTE pbAttr,
855 //                  _In_    DWORD cbAttrLen);
856 //
857 //          Here's a work-around MACRO:
858 #define SCardSetReaderCapabilities SCardSetAttrib
859
```

```
860
861 //
862 ////////////////////////////////////////////////////////////////////////////
      //
863 //
864 //  Smart Card Dialog definitions
865 //
866 //      The following section contains structures and  exported function
867 //      declarations for the Smart Card Common Dialog dialog.
868 //
869
870 // Defined constants
871 // Flags
872 #define SC_DLG_MINIMAL_UI       0x01
873 #define SC_DLG_NO_UI            0x02
874 #define SC_DLG_FORCE_UI         0x04
875
876 #define SCERR_NOCARDNAME        0x4000
877 #define SCERR_NOGUIDS           0x8000
878
879 typedef SCARDHANDLE (WINAPI *LPOCNCONNPROCA) (_In_ SCARDCONTEXT, _In_ LPSTR,
      _In_ LPSTR, _In_ PVOID);
880 typedef SCARDHANDLE (WINAPI *LPOCNCONNPROCW) (_In_ SCARDCONTEXT, _In_ LPWSTR,
      _In_ LPWSTR, _In_ PVOID);
881 #ifdef UNICODE
882 #define LPOCNCONNPROC  LPOCNCONNPROCW
883 #else
884 #define LPOCNCONNPROC  LPOCNCONNPROCA
885 #endif // !UNICODE
886 typedef BOOL (WINAPI *LPOCNCHKPROC) (_In_ SCARDCONTEXT, _In_ SCARDHANDLE, _In_
      PVOID);
887 typedef void (WINAPI *LPOCNDSCPROC) (_In_ SCARDCONTEXT, _In_ SCARDHANDLE, _In_
      PVOID);
888
889
890 //
891 // OPENCARD_SEARCH_CRITERIA: In order to specify a user-extended search,
892 // lpfnCheck must not be NULL.  Moreover, the connection to be made to the
893 // card before performing the callback must be indicated by either providing
894 // lpfnConnect and lpfnDisconnect OR by setting dwShareMode.
895 // If both the connection callbacks and dwShareMode are non-NULL, the
      callbacks
896 // will be used.
897 //
898
899 typedef struct {
900     DWORD           dwStructSize;
901     LPSTR           lpstrGroupNames;        // OPTIONAL reader groups to
        include in
```

```
902     DWORD           nMaxGroupNames;         //          search.  NULL defaults
          to
903                                             //          SCard$DefaultReaders
904     LPCGUID         rgguidInterfaces;       // OPTIONAL requested interfaces
905     DWORD           cguidInterfaces;        //          supported by card's
          SSP
906     LPSTR           lpstrCardNames;         // OPTIONAL requested card names;
          all cards w/
907     DWORD           nMaxCardNames;          //          matching ATRs will be
          accepted
908     LPOCNCHKPROC    lpfnCheck;              // OPTIONAL if NULL no user check
          will be performed.
909     LPOCNCONNPROCA  lpfnConnect;            // OPTIONAL if lpfnConnect is
          provided,
910     LPOCNDSCPROC    lpfnDisconnect;         //          lpfnDisconnect must
          also be set.
911     LPVOID          pvUserData;             // OPTIONAL parameter to callbacks
912     DWORD           dwShareMode;            // OPTIONAL must be set if
          lpfnCheck is not null
913     DWORD           dwPreferredProtocols;   // OPTIONAL
914 } OPENCARD_SEARCH_CRITERIAA, *POPENCARD_SEARCH_CRITERIAA,
      *LPOPENCARD_SEARCH_CRITERIAA;
915 typedef struct {
916     DWORD           dwStructSize;
917     LPWSTR          lpstrGroupNames;        // OPTIONAL reader groups to
          include in
918     DWORD           nMaxGroupNames;         //          search.  NULL defaults
          to
919                                             //          SCard$DefaultReaders
920     LPCGUID         rgguidInterfaces;       // OPTIONAL requested interfaces
921     DWORD           cguidInterfaces;        //          supported by card's
          SSP
922     LPWSTR          lpstrCardNames;         // OPTIONAL requested card names;
          all cards w/
923     DWORD           nMaxCardNames;          //          matching ATRs will be
          accepted
924     LPOCNCHKPROC    lpfnCheck;              // OPTIONAL if NULL no user check
          will be performed.
925     LPOCNCONNPROCW  lpfnConnect;            // OPTIONAL if lpfnConnect is
          provided,
926     LPOCNDSCPROC    lpfnDisconnect;         //          lpfnDisconnect must
          also be set.
927     LPVOID          pvUserData;             // OPTIONAL parameter to callbacks
928     DWORD           dwShareMode;            // OPTIONAL must be set if
          lpfnCheck is not null
929     DWORD           dwPreferredProtocols;   // OPTIONAL
930 } OPENCARD_SEARCH_CRITERIAW, *POPENCARD_SEARCH_CRITERIAW,
      *LPOPENCARD_SEARCH_CRITERIAW;
931 #ifdef UNICODE
```

```
932 typedef OPENCARD_SEARCH_CRITERIAW OPENCARD_SEARCH_CRITERIA;
933 typedef POPENCARD_SEARCH_CRITERIAW POPENCARD_SEARCH_CRITERIA;
934 typedef LPOPENCARD_SEARCH_CRITERIAW LPOPENCARD_SEARCH_CRITERIA;
935 #else
936 typedef OPENCARD_SEARCH_CRITERIAA OPENCARD_SEARCH_CRITERIA;
937 typedef POPENCARD_SEARCH_CRITERIAA POPENCARD_SEARCH_CRITERIA;
938 typedef LPOPENCARD_SEARCH_CRITERIAA LPOPENCARD_SEARCH_CRITERIA;
939 #endif // UNICODE
940
941
942 //
943 // OPENCARDNAME_EX: used by SCardUIDlgSelectCard; replaces obsolete
      OPENCARDNAME
944 //
945
946 typedef struct {
947     DWORD           dwStructSize;           // REQUIRED
948     SCARDCONTEXT    hSCardContext;          // REQUIRED
949     HWND            hwndOwner;              // OPTIONAL
950     DWORD           dwFlags;                // OPTIONAL -- default is
        SC_DLG_MINIMAL_UI
951     LPCSTR          lpstrTitle;             // OPTIONAL
952     LPCSTR          lpstrSearchDesc;        // OPTIONAL (eg. "Please insert
        your <brandname> smart card.")
953     HICON           hIcon;                  // OPTIONAL 32x32 icon for your
        brand insignia
954     POPENCARD_SEARCH_CRITERIAA pOpenCardSearchCriteria; // OPTIONAL
955     LPOCNCONNPROCA  lpfnConnect;            // OPTIONAL - performed on
        successful selection
956     LPVOID          pvUserData;             // OPTIONAL parameter to
        lpfnConnect
957     DWORD           dwShareMode;            // OPTIONAL - if lpfnConnect is
        NULL, dwShareMode and
958     DWORD           dwPreferredProtocols;   // OPTIONAL dwPreferredProtocols
        will be used to
959                                             //          connect to the
                    selected card
960     LPSTR           lpstrRdr;               // REQUIRED [IN|OUT] Name of
        selected reader
961     DWORD           nMaxRdr;                // REQUIRED [IN|OUT]
962     LPSTR           lpstrCard;              // REQUIRED [IN|OUT] Name of
        selected card
963     DWORD           nMaxCard;               // REQUIRED [IN|OUT]
964     DWORD           dwActiveProtocol;       // [OUT] set only if dwShareMode
        not NULL
965     SCARDHANDLE     hCardHandle;            // [OUT] set if a card connection
        was indicated
966 } OPENCARDNAME_EXA, *POPENCARDNAME_EXA, *LPOPENCARDNAME_EXA;
967 typedef struct {
```

```
968     DWORD           dwStructSize;           // REQUIRED
969     SCARDCONTEXT    hSCardContext;          // REQUIRED
970     HWND            hwndOwner;              // OPTIONAL
971     DWORD           dwFlags;                // OPTIONAL -- default is
          SC_DLG_MINIMAL_UI
972     LPCWSTR         lpstrTitle;             // OPTIONAL
973     LPCWSTR         lpstrSearchDesc;        // OPTIONAL (eg. "Please insert
          your <brandname> smart card.")
974     HICON           hIcon;                  // OPTIONAL 32x32 icon for your
          brand insignia
975     POPENCARD_SEARCH_CRITERIAW pOpenCardSearchCriteria; // OPTIONAL
976     LPOCNCONNPROCW  lpfnConnect;            // OPTIONAL - performed on
          successful selection
977     LPVOID          pvUserData;             // OPTIONAL parameter to
          lpfnConnect
978     DWORD           dwShareMode;            // OPTIONAL - if lpfnConnect is
          NULL, dwShareMode and
979     DWORD           dwPreferredProtocols;   // OPTIONAL dwPreferredProtocols
          will be used to
980                                             //          connect to the
                    selected card
981     LPWSTR          lpstrRdr;               // REQUIRED [IN|OUT] Name of
          selected reader
982     DWORD           nMaxRdr;                // REQUIRED [IN|OUT]
983     LPWSTR          lpstrCard;              // REQUIRED [IN|OUT] Name of
          selected card
984     DWORD           nMaxCard;               // REQUIRED [IN|OUT]
985     DWORD           dwActiveProtocol;       // [OUT] set only if dwShareMode
          not NULL
986     SCARDHANDLE     hCardHandle;            // [OUT] set if a card connection
          was indicated
987 } OPENCARDNAME_EXW, *POPENCARDNAME_EXW, *LPOPENCARDNAME_EXW;
988 #ifdef UNICODE
989 typedef OPENCARDNAME_EXW OPENCARDNAME_EX;
990 typedef POPENCARDNAME_EXW POPENCARDNAME_EX;
991 typedef LPOPENCARDNAME_EXW LPOPENCARDNAME_EX;
992 #else
993 typedef OPENCARDNAME_EXA OPENCARDNAME_EX;
994 typedef POPENCARDNAME_EXA POPENCARDNAME_EX;
995 typedef LPOPENCARDNAME_EXA LPOPENCARDNAME_EX;
996 #endif // UNICODE
997
998 #define OPENCARDNAMEA_EX OPENCARDNAME_EXA
999 #define OPENCARDNAMEW_EX OPENCARDNAME_EXW
1000 #define POPENCARDNAMEA_EX POPENCARDNAME_EXA
1001 #define POPENCARDNAMEW_EX POPENCARDNAME_EXW
1002 #define LPOPENCARDNAMEA_EX LPOPENCARDNAME_EXA
1003 #define LPOPENCARDNAMEW_EX LPOPENCARDNAME_EXW
1004
```

```
1005
1006 //
1007 // Smart Card Reader Selection Provider
1008 //
1009 // Only UNICODE is supported. Invoke smart card reader selection provider by calling
1010 // CredUIPromptForWindowsCredentials() supplying SCARD_READER_SEL_AUTH_PACKAGE as
1011 // pulAuthPackage, an instance of READER_SEL_REQUEST as pvInAuthBuffer and setting
1012 // CREDUIWIN_AUTHPACKAGE_ONLY in dwFlags. Upon successful return, an instance of
1013 // READER_SEL_RESPONSE will be returned in ppvOutAuthBuffer.
1014 //
1015
1016 #define SCARD_READER_SEL_AUTH_PACKAGE   ((DWORD)-629)
1017
1018 //
1019 // READER_SEL_REQUEST
1020 //     Reader selection request to reader selection provider
1021 //
1022 // Members:
1023 //
1024 // dwShareMode:
1025 //     Share mode used by SCardConnect to connect to smart cards
1026 // dwPreferredProtocols:
1027 //     Acceptable protocols for SCardConnect to connect to smart cards
1028 //
1029 // MatchType:
1030 //     Indicates how the caller wants the reader selection provider to verify smart
1031 //     cards.
1032 //
1033 //     If MatchType is set to RSR_MATCH_TYPE_READER_AND_CONTAINER, reader selection
1034 //     provider will match smart cards based on whether they are in the given
1035 //     reader and have the given key container. Reader name and container name are
1036 //     both optional. Reader name and container name, if any, need to be appended
1037 //     after READER_SEL_REQUEST structure and set their offsets and lengths in
1038 //     ReaderAndContainerParameter member.
1039 //
1040 //     If MatchType is set to RSR_MATCH_TYPE_SERIAL_NUMBER, reader selection
1041 //     provider will match smart cards based on their serial numbers / card IDs.
1042 //     Serial number is required. It needs to be appended after READER_SEL_REQUEST
1043 //     structure as a byte array and set its offset and length in
```

```
1044 //     SerialNumberParameter member.
1045 //
1046 //     If MatchType is set to RSR_MATCH_TYPE_ALL_CARDS, reader selection provider
1047 //     will allow all recognized cards to be selected by user without any filtering.
1048 //     The card may not be personalized for Base CSP / Smart Card KSP yet, or even
1049 //     have its own CSP.
1050 //
1051 // ReaderAndContainerParameter.cbReaderNameOffset:
1052 //     Byte offset of reader name UNICODE string from the beginning of
1053 //     READER_SEL_REQUEST structure
1054 // ReaderAndContainerParameter.cchReaderNameLength:
1055 //     Number of characters in reader name UNICODE string including the terminating
1056 //     NULL character
1057 // ReaderAndContainerParameter.cbContainerNameOffset:
1058 //     Byte offset of container name UNICODE string from the beginning of
1059 //     READER_SEL_REQUEST structure
1060 // ReaderAndContainerParameter.cchContainerNameLength:
1061 //     Number of characters in container name UNICODE string including the
1062 //     terminating NULL character
1063 // ReaderAndContainerParameter.dwDesiredCardModuleVersion:
1064 //     The desired smart card module version
1065 // ReaderAndContainerParameter.dwCspFlags:
1066 //     CSP and KSP flags to indicate how smart cards will be used
1067 //     (Valid flags include CRYPT_NEWKEYSET, CRYPT_DELETEKEYSET, CRYPT_VERIFYCONTEXT
1068 //     and CRYPT_DEFAULT_CONTAINER_OPTIONAL)
1069 //
1070 // SerialNumberParameter.cbSerialNumberOffset:
1071 //     Byte offset of serial number byte array from the beginning of
1072 //     READER_SEL_REQUEST structure
1073 // SerialNumberParameter.cbSerialNumberLength:
1074 //     Number of bytes in serial number byte array
1075 // SerialNumberParameter.dwDesiredCardModuleVersion:
1076 //     The desired smart card module version
1077 //
1078
1079 typedef enum {
1080     RSR_MATCH_TYPE_READER_AND_CONTAINER = 1,
1081     RSR_MATCH_TYPE_SERIAL_NUMBER,
1082     RSR_MATCH_TYPE_ALL_CARDS
1083 } READER_SEL_REQUEST_MATCH_TYPE;
1084
1085 typedef struct {
1086     DWORD                           dwShareMode;
1087     DWORD                           dwPreferredProtocols;
```

```
1088      READER_SEL_REQUEST_MATCH_TYPE   MatchType;
1089      union {
1090          struct {
1091              DWORD                   cbReaderNameOffset;
1092              DWORD                   cchReaderNameLength;
1093              DWORD                   cbContainerNameOffset;
1094              DWORD                   cchContainerNameLength;
1095              DWORD                   dwDesiredCardModuleVersion;
1096              DWORD                   dwCspFlags;
1097          } ReaderAndContainerParameter;
1098          struct {
1099              DWORD                   cbSerialNumberOffset;
1100              DWORD                   cbSerialNumberLength;
1101              DWORD                   dwDesiredCardModuleVersion;
1102          } SerialNumberParameter;
1103      };
1104  } READER_SEL_REQUEST, *PREADER_SEL_REQUEST;
1105
1106  //
1107  // READER_SEL_RESPONSE
1108  //     Reader selection response from reader selection provider
1109  //
1110  // Members:
1111  // cbReaderNameOffset:
1112  //     Byte offset of matched reader name UNICODE string from the beginning of
1113  //     READER_SEL_RESPONSE structure
1114  // cchReaderNameLength:
1115  //     Number of characters in matched reader name UNICODE string including
        the
1116  //     terminating NULL character
1117  // cbCardNameOffset:
1118  //     Byte offset of matched card name UNICODE string from the beginning of
1119  //     READER_SEL_RESPONSE structure
1120  // cchCardNameLength:
1121  //     Number of characters in matched card name UNICODE string including the
1122  //     terminating NULL character
1123  //
1124
1125  typedef struct {
1126      DWORD                           cbReaderNameOffset;
1127      DWORD                           cchReaderNameLength;
1128      DWORD                           cbCardNameOffset;
1129      DWORD                           cchCardNameLength;
1130  } READER_SEL_RESPONSE, *PREADER_SEL_RESPONSE;
1131
1132
1133  //
1134  // SCardUIDlgSelectCard replaces GetOpenCardName
1135  //
```

```
1136
1137 extern WINSCARDAPI LONG WINAPI
1138 SCardUIDlgSelectCardA(
1139     LPOPENCARDNAMEA_EX);
1140 extern WINSCARDAPI LONG WINAPI
1141 SCardUIDlgSelectCardW(
1142     LPOPENCARDNAMEW_EX);
1143 #ifdef UNICODE
1144 #define SCardUIDlgSelectCard  SCardUIDlgSelectCardW
1145 #else
1146 #define SCardUIDlgSelectCard  SCardUIDlgSelectCardA
1147 #endif // !UNICODE
1148
1149
1150 //
1151 // "Smart Card Common Dialog" definitions for backwards compatibility
1152 //  with the Smart Card Base Services SDK version 1.0
1153 //
1154
1155 typedef struct {
1156     DWORD           dwStructSize;
1157     HWND            hwndOwner;
1158     SCARDCONTEXT    hSCardContext;
1159     LPSTR           lpstrGroupNames;
1160     DWORD           nMaxGroupNames;
1161     LPSTR           lpstrCardNames;
1162     DWORD           nMaxCardNames;
1163     LPCGUID         rgguidInterfaces;
1164     DWORD           cguidInterfaces;
1165     LPSTR           lpstrRdr;
1166     DWORD           nMaxRdr;
1167     LPSTR           lpstrCard;
1168     DWORD           nMaxCard;
1169     LPCSTR          lpstrTitle;
1170     DWORD           dwFlags;
1171     LPVOID          pvUserData;
1172     DWORD           dwShareMode;
1173     DWORD           dwPreferredProtocols;
1174     DWORD           dwActiveProtocol;
1175     LPOCNCONNPROCA  lpfnConnect;
1176     LPOCNCHKPROC    lpfnCheck;
1177     LPOCNDSCPROC    lpfnDisconnect;
1178     SCARDHANDLE     hCardHandle;
1179 } OPENCARDNAMEA, *POPENCARDNAMEA, *LPOPENCARDNAMEA;
1180 typedef struct {
1181     DWORD           dwStructSize;
1182     HWND            hwndOwner;
1183     SCARDCONTEXT    hSCardContext;
1184     LPWSTR          lpstrGroupNames;
```

```
1185      DWORD           nMaxGroupNames;
1186      LPWSTR          lpstrCardNames;
1187      DWORD           nMaxCardNames;
1188      LPCGUID         rgguidInterfaces;
1189      DWORD           cguidInterfaces;
1190      LPWSTR          lpstrRdr;
1191      DWORD           nMaxRdr;
1192      LPWSTR          lpstrCard;
1193      DWORD           nMaxCard;
1194      LPCWSTR         lpstrTitle;
1195      DWORD           dwFlags;
1196      LPVOID          pvUserData;
1197      DWORD           dwShareMode;
1198      DWORD           dwPreferredProtocols;
1199      DWORD           dwActiveProtocol;
1200      LPOCNCONNPROCW  lpfnConnect;
1201      LPOCNCHKPROC    lpfnCheck;
1202      LPOCNDSCPROC    lpfnDisconnect;
1203      SCARDHANDLE     hCardHandle;
1204  } OPENCARDNAMEW, *POPENCARDNAMEW, *LPOPENCARDNAMEW;
1205  #ifdef UNICODE
1206  typedef OPENCARDNAMEW OPENCARDNAME;
1207  typedef POPENCARDNAMEW POPENCARDNAME;
1208  typedef LPOPENCARDNAMEW LPOPENCARDNAME;
1209  #else
1210  typedef OPENCARDNAMEA OPENCARDNAME;
1211  typedef POPENCARDNAMEA POPENCARDNAME;
1212  typedef LPOPENCARDNAMEA LPOPENCARDNAME;
1213  #endif // UNICODE
1214
1215  // Backwards compatibility macros
1216  #define OPENCARDNAME_A OPENCARDNAMEA
1217  #define OPENCARDNAME_W OPENCARDNAMEW
1218  #define POPENCARDNAME_A POPENCARDNAMEA
1219  #define POPENCARDNAME_W POPENCARDNAMEW
1220  #define LPOPENCARDNAME_A LPOPENCARDNAMEA
1221  #define LPOPENCARDNAME_W LPOPENCARDNAMEW
1222
1223  extern WINSCARDAPI LONG WINAPI
1224  GetOpenCardNameA(
1225      LPOPENCARDNAMEA);
1226  extern WINSCARDAPI LONG WINAPI
1227  GetOpenCardNameW(
1228      LPOPENCARDNAMEW);
1229  #ifdef UNICODE
1230  #define GetOpenCardName  GetOpenCardNameW
1231  #else
1232  #define GetOpenCardName  GetOpenCardNameA
1233  #endif // !UNICODE
```

```
1234
1235 extern WINSCARDAPI LONG WINAPI
1236 SCardDlgExtendedError (void);
1237
1238 #if (NTDDI_VERSION >= NTDDI_VISTA)
1239
1240 //
1241 // Smartcard Caching API
1242 //
1243
1244 extern WINSCARDAPI LONG WINAPI
1245 SCardReadCacheA(
1246     _In_  SCARDCONTEXT hContext,
1247     _In_  UUID *CardIdentifier,
1248     _In_  DWORD FreshnessCounter,
1249     _In_  LPSTR LookupName,
1250     _Out_writes_bytes_(*DataLen) PBYTE Data,
1251     _Out_ DWORD *DataLen);
1252 extern WINSCARDAPI LONG WINAPI
1253 SCardReadCacheW(
1254     _In_  SCARDCONTEXT hContext,
1255     _In_  UUID *CardIdentifier,
1256     _In_  DWORD FreshnessCounter,
1257     _In_  LPWSTR LookupName,
1258     _Out_writes_bytes_(*DataLen) PBYTE Data,
1259     _Out_ DWORD *DataLen);
1260 #ifdef UNICODE
1261 #define SCardReadCache  SCardReadCacheW
1262 #else
1263 #define SCardReadCache  SCardReadCacheA
1264 #endif // !UNICODE
1265
1266 extern WINSCARDAPI LONG WINAPI
1267 SCardWriteCacheA(
1268     _In_ SCARDCONTEXT hContext,
1269     _In_ UUID *CardIdentifier,
1270     _In_ DWORD FreshnessCounter,
1271     _In_ LPSTR LookupName,
1272     _In_reads_bytes_(DataLen) PBYTE Data,
1273     _In_ DWORD DataLen);
1274 extern WINSCARDAPI LONG WINAPI
1275 SCardWriteCacheW(
1276     _In_ SCARDCONTEXT hContext,
1277     _In_ UUID *CardIdentifier,
1278     _In_ DWORD FreshnessCounter,
1279     _In_ LPWSTR LookupName,
1280     _In_reads_bytes_(DataLen) PBYTE Data,
1281     _In_ DWORD DataLen);
1282 #ifdef UNICODE
```

```
1283  #define SCardWriteCache  SCardWriteCacheW
1284  #else
1285  #define SCardWriteCache  SCardWriteCacheA
1286  #endif // !UNICODE
1287
1288  #endif // (NTDDI_VERSION >= NTDDI_VISTA)
1289
1290  #if (NTDDI_VERSION >= NTDDI_WIN8)
1291
1292  _Success_(return == SCARD_S_SUCCESS)
1293  extern WINSCARDAPI LONG WINAPI
1294  SCardGetReaderIconA(
1295    _In_     SCARDCONTEXT hContext,
1296    _In_     LPCSTR szReaderName,
1297    _When_(_Old_(*pcbIcon) == SCARD_AUTOALLOCATE, _At_((LPBYTE *)pbIcon,
        _Outptr_result_bytebuffer_all_maybenull_(*pcbIcon)))
1298    _When_(_Old_(*pcbIcon) != SCARD_AUTOALLOCATE, _Out_writes_bytes_to_
        (*pcbIcon, *pcbIcon) _Post_z_)
1299           LPBYTE pbIcon,
1300    _Inout_  LPDWORD pcbIcon);
1301  _Success_(return == SCARD_S_SUCCESS)
1302  extern WINSCARDAPI LONG WINAPI
1303  SCardGetReaderIconW(
1304    _In_     SCARDCONTEXT hContext,
1305    _In_     LPCWSTR szReaderName,
1306    _When_(_Old_(*pcbIcon) == SCARD_AUTOALLOCATE, _At_((LPBYTE *)pbIcon,
        _Outptr_result_bytebuffer_all_maybenull_(*pcbIcon)))
1307    _When_(_Old_(*pcbIcon) != SCARD_AUTOALLOCATE, _Out_writes_bytes_to_
        (*pcbIcon, *pcbIcon) _Post_z_)
1308           LPBYTE pbIcon,
1309    _Inout_  LPDWORD pcbIcon);
1310  #ifdef UNICODE
1311  #define SCardGetReaderIcon  SCardGetReaderIconW
1312  #else
1313  #define SCardGetReaderIcon  SCardGetReaderIconA
1314  #endif // !UNICODE
1315
1316  _Success_(return == SCARD_S_SUCCESS)
1317  extern WINSCARDAPI LONG WINAPI
1318  SCardGetDeviceTypeIdA(
1319    _In_     SCARDCONTEXT hContext,
1320    _In_     LPCSTR szReaderName,
1321    _Inout_  LPDWORD pdwDeviceTypeId);
1322  _Success_(return == SCARD_S_SUCCESS)
1323  extern WINSCARDAPI LONG WINAPI
1324  SCardGetDeviceTypeIdW(
1325    _In_     SCARDCONTEXT hContext,
1326    _In_     LPCWSTR szReaderName,
1327    _Inout_  LPDWORD pdwDeviceTypeId);
```

```
1328  #ifdef UNICODE
1329  #define SCardGetDeviceTypeId  SCardGetDeviceTypeIdW
1330  #else
1331  #define SCardGetDeviceTypeId  SCardGetDeviceTypeIdA
1332  #endif // !UNICODE
1333
1334  _Success_(return == SCARD_S_SUCCESS)
1335  extern WINSCARDAPI LONG WINAPI
1336  SCardGetReaderDeviceInstanceIdA(
1337    _In_    SCARDCONTEXT hContext,
1338    _In_    LPCSTR szReaderName,
1339    _When_(_Old_(*pcchDeviceInstanceId) == SCARD_AUTOALLOCATE, _At_((LPSTR *)
          szDeviceInstanceId, _Outptr_result_buffer_maybenull_
          (*pcchDeviceInstanceId) _At_(*_Curr_, _Post_z_ _Post_
          _NullNull_terminated_)))
1340    _When_(_Old_(*pcchDeviceInstanceId) != SCARD_AUTOALLOCATE, _Out_writes_opt_
          (*pcchDeviceInstanceId) _Post_ _NullNull_terminated_)
1341          LPSTR szDeviceInstanceId,
1342    _Inout_ LPDWORD pcchDeviceInstanceId);
1343  _Success_(return == SCARD_S_SUCCESS)
1344  extern WINSCARDAPI LONG WINAPI
1345  SCardGetReaderDeviceInstanceIdW(
1346    _In_    SCARDCONTEXT hContext,
1347    _In_    LPCWSTR szReaderName,
1348    _When_(_Old_(*pcchDeviceInstanceId) == SCARD_AUTOALLOCATE, _At_((LPWSTR *)
          szDeviceInstanceId, _Outptr_result_buffer_maybenull_
          (*pcchDeviceInstanceId) _At_(*_Curr_, _Post_z_ _Post_
          _NullNull_terminated_)))
1349    _When_(_Old_(*pcchDeviceInstanceId) != SCARD_AUTOALLOCATE, _Out_writes_opt_
          (*pcchDeviceInstanceId) _Post_ _NullNull_terminated_)
1350          LPWSTR szDeviceInstanceId,
1351    _Inout_ LPDWORD pcchDeviceInstanceId);
1352  #ifdef UNICODE
1353  #define SCardGetReaderDeviceInstanceId  SCardGetReaderDeviceInstanceIdW
1354  #else
1355  #define SCardGetReaderDeviceInstanceId  SCardGetReaderDeviceInstanceIdA
1356  #endif // !UNICODE
1357
1358  _Success_(return == SCARD_S_SUCCESS)
1359  extern WINSCARDAPI LONG WINAPI
1360  SCardListReadersWithDeviceInstanceIdA(
1361    _In_    SCARDCONTEXT hContext,
1362    _In_    LPCSTR szDeviceInstanceId,
1363    _When_(_Old_(*pcchReaders) == SCARD_AUTOALLOCATE, _At_((LPSTR *)mszReaders,
          _Outptr_result_buffer_maybenull_(*pcchReaders) _At_(*_Curr_, _Post_z_
          _Post_ _NullNull_terminated_)))
1364    _When_(_Old_(*pcchReaders) != SCARD_AUTOALLOCATE, _Out_writes_opt_
          (*pcchReaders) _Post_ _NullNull_terminated_)
1365          LPSTR mszReaders,
```

```
1366    _Inout_ LPDWORD pcchReaders);
1367  _Success_(return == SCARD_S_SUCCESS)
1368  extern WINSCARDAPI LONG WINAPI
1369  SCardListReadersWithDeviceInstanceIdW(
1370    _In_    SCARDCONTEXT hContext,
1371    _In_    LPCWSTR szDeviceInstanceId,
1372    _When_(_Old_(*pcchReaders) == SCARD_AUTOALLOCATE, _At_((LPWSTR *)mszReaders, ↵
           _Outptr_result_buffer_maybenull_(*pcchReaders) _At_(*_Curr_, _Post_z_     ↵
         _Post_ _NullNull_terminated_)))
1373    _When_(_Old_(*pcchReaders) != SCARD_AUTOALLOCATE, _Out_writes_opt_           ↵
          (*pcchReaders) _Post_ _NullNull_terminated_)
1374          LPWSTR mszReaders,
1375    _Inout_ LPDWORD pcchReaders);
1376  #ifdef UNICODE
1377  #define SCardListReadersWithDeviceInstanceId                                   ↵
        SCardListReadersWithDeviceInstanceIdW
1378  #else
1379  #define SCardListReadersWithDeviceInstanceId                                   ↵
        SCardListReadersWithDeviceInstanceIdA
1380  #endif // !UNICODE
1381
1382  //
1383  ////////////////////////////////////////////////////////////////////////////// ↵
        //
1384  //
1385  //  Smart Card Auditing
1386  //
1387
1388  #define SCARD_AUDIT_CHV_FAILURE 0x0 // A smart card holder verification (CHV)
1389                                      // attempt failed.
1390
1391  #define SCARD_AUDIT_CHV_SUCCESS 0x1 // A smart card holder verification (CHV)
1392                                      // attempt succeeded.
1393
1394  _Success_(return == SCARD_S_SUCCESS)
1395  extern WINSCARDAPI LONG WINAPI
1396  SCardAudit(
1397    _In_ SCARDCONTEXT hContext,
1398    _In_ DWORD dwEvent);
1399
1400  #endif // (NTDDI_VERSION >= NTDDI_WIN8)
1401
1402  #endif /* WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP) */
1403  #pragma endregion
1404
1405  #ifdef __cplusplus
1406  }
1407  #endif
1408  #endif // _WINSCARD_H_
```

1409
1410