Interazione Marcatrice Laser-Token USB

Specifiche tecniche

"InfoCamere"

Versione 1.7.2

Storia delle revisioni

Versione	Descrizione
Prima versione	Prima versione
Versione 1.6	• Precisazione sul parametro machine_id della funzione di inizializzazione del token (§ 4.4.2.3) • Inserimento funzione recupero nomi file immagine nel token (§ 4.4.2.8) • Inserimento funzione recupero seriale smart card del token (§ 4.4.2.9) • Correzione di un errore nell'esempio di utilizzo delle API (§ 5.1) • Aggiornamento integrale Allegato 2 (§ 5.2)
Versione 1.7	• Inserimento dei file del marchio demo (§ 3.2) • Aggiornamento lista codici di errore (§ 4.3) • Nuova sezione per codici di warning (§ 4.4) • Precisazioni sui codici restituiti dalla funzione *epunch_initialize* (§ 4.5.2.3) • Inserimento descrizione standard PLT (§ 5.3)
Versione 1.7.1	• Correzione di un refuso in merito al numero di grandezze del marchio del titolo G= 1..4 anziché G=1..5 (§ 3.2)
Versione 1.7.2	• Precisazioni sulla gestione delle anomalie (§ 3.3.3 e § 5.4)

Termini e Definizioni

Termine	Definizione
Tecnologia laser	Tecnologia utilizzata per apporre sugli oggetti in metallo prezioso il marchio di identificazione e l'indicazione del titolo legale mediante un dispositivo in grado di emettere un raggio luminoso amplificato che altera, attraverso un processo di riscaldamento termico localizzato, lo stato cromatico di una superficie.
Marcatrice laser	Combinazione di un dispositivo di produzione di un raggio laser e di un controller (tipicamente un computer) che sovrintende a tutte le variabili che intervengono nel processo di marcatura laser.
Token USB	Dispositivo di memoria di massa, dotato di misure e accorgimenti anti intrusione, utilizzato dalle marcatrici laser per l'applicazione del marchio di identificazione e del titolo con tecnologia laser.
Codice di sblocco (PIN)	Codice assegnato al Token USB che ne consente lo sblocco per l'abilitazione all'utilizzo.
File immagine	File vettoriale contenente la versione informatica di un marchio di identificazione di un'impresa o dell'indicazione del titolo legale.

Sommario

1 Scopo e campo di applicazione del documento

Questo documento si rivolge ai fabbricanti di macchinari per la marcatura di oggetti in metallo prezioso mediante tecnologia laser (di seguito “marcatrici laser” o “marcatrici”) al fine di fornire le specifiche tecniche necessarie alla progettazione e realizzazione del software di governo dell’interazione fra la marcatrice e i dispositivi di memoria di massa che custodiscono le immagini del marchio di identificazione di un’impresa e dell’indicazione del titolo legale (token USB).

Allo scopo di contestualizzare i contenuti descritti nel seguito del documento, si ricorda che il processo di marcatura mediate tecnologia laser prevede l’interazione di due componenti tecnologiche opportunamente collegate:

1. La *marcatrice* **-** per l’apposizione di un marchio mediante tecnologia laser
2. Il *token USB* - che custodisce le immagini del marchio identificativo di un’impresa e dell’indicazione del titolo legale utilizzate dalla marcatrice per apporre il marchio stesso

Tale interazione viene abilitata tramite l’inserimento di un *codice di sblocco* (PIN) in possesso dell’impresa assegnataria del token USB.

Vengono quindi presentate in questo documento, le specifiche tecnico-funzionali dei Token USB e le specifiche di interfaccia marcatrice-Token USB (*Application Programming Interface*) che il software di controllo delle marcatrici dovrà rispettare per accedere in sicurezza alle informazioni presenti nei token e procedere con l’apposizione del marchio di identificazione e del titolo.

2 Specifiche hardware

I Token USB possiedono le seguenti caratteristiche hardware

- FLASH MEMORY inizializzata con moduli software hardenizzati per l'archiviazione sicura dei file di marchio identificativo
- HID / CCID Smartcard reader plug-in (SIM format)
- Interfaccia USB 2.0 full speed
- PKCS#11 on HID/CCID Interface
- Smart Card CNS dotata di certificazione Common Criteria con livello pari a EAL 4+ conformemente ad almeno uno dei seguenti Protection Profile:
  - CWA 14169 type 2
  - CWA 14169 type 3

3 Specifiche funzionali

3.1 Sistemi operativi supportati

- MS Windows XP, Vista, Seven,8.0, 8.1, Server 2003, Server 2008 (32 e 64 bit) o superiori
- Mac Os X:Snow Leopard (10.6 - Intel), Lion 10.7 , Mountain Lion 10.8, Mavericks 10.9.X o superiori

3.2 Contenuto della FLASH MEMORY

Il file system della FLASH MEMORY è organizzato come descritto nel seguito:

```
/additional_data/
/bin/api.dll
/epunch.exe
/imgs_id/
/imgs_titles/
/log/
```

- La cartella `additional_data/` contiene dati per applicazioni future

- La cartella `bin/` contiene l'api di interfacciamento ad uso delle marcatrici: `api.dll`. `api.dll` espone l'interfaccia descritta in § **Errore. L'origine riferimento non è stata trovata.**.

- L'eseguibile `/epunch.exe` rappresenta l'eseguibile utile alle operazioni di gestione del token quali: cambio PIN, sblocco PIN, etc…

- La cartella *imgs_id/* contiene i file dei marchi identificativi (i cinque file del marchio assegnato all'impresa e cinque file di un marchio demo da utilizzare per le fasi di test).
  I nomi dei file consentono di individuare il corretto file da utilizzare per la marcatura di uno specifico lotto di metallo prezioso e adottano la seguente convenzione:

  a. " ***Marchio_G.****XXX***"** per i file del marchio dell'impresa

  con

  G = 1..5 (I valori di questa variabile identificano una delle 5 grandezze descritte nel DPR 05/2002 n. 150 all'allegato III e successive modificazioni)
  XXX = estensione del file.
  Ad esempio: Marchio_1.plt.

  b. " ***DEMO_G.****XXX***"** per i file demo

  con

  G = 1..5 (I valori di questa variabile identificano una delle 5 grandezze descritte nel DPR 05/2002 n. 150 all'allegato III e successive modificazioni)
  XXX = estensione del file.
  Ad esempio: DEMO_1.plt.

- La cartella *imgs_titles/* contiene i file dei titoli legali del marchio.

  I nomi dei file consentono di individuare il corretto file da utilizzare per la marcatura di un specifico lotto di metallo prezioso e adottano la seguente convenzione:

  "Titolo_MM_TTT_G.XXX"

  con

  MM = *Pt, Pd, Au, Ag* (i valori di questa variabile identificano i metalli preziosi descritti nel Dlg 05/1999 n. 251 art. 1);
  TTT = *375, 585, 750, 800, 850, 900, 925, 950* (i valori di questa variabile identificano il valore del titolo legale associato ad un metallo prezioso come descritto nel Dlg 05/1999 n. 251 art. 3 coma 2);
  G= 1..4 (i valori di questa variabile identificano una delle 4 grandezze descritte nel DPR 05/2002 n. 150 all'allegato V e successive modificazioni)
  XXX = estensione del file.
  Ad esempio: Titolo_Au_375_1.plt.

- La cartella *log/*contiene gli oggetti utili al tracciamento dell'uso del Token USB sulle marcatrici

Per maggiori dettagli sullo standard utilizzato per codificare i file .plt custoditi nel token USB si veda § 5.3.

3.3 Operatività

Il seguente paragrafo contiene la descrizione della procedura che deve essere obbligatoriamente seguita per il corretto uso del Token USB da parte della marcatrice:

3.3.1 Collegamento ed utilizzo del Token USB

1. Il Token USB viene collegato alla postazione che pilota la marcatrice e, attraverso il software della marcatrice installato nel PC, viene selezionato il percorso della libreria *api.dll* del Token USB

2. Il software della marcatrice richiede il PIN in possesso del titolare del Token USB

3. Il software della marcatrice carica la libreria d'interfacciamento (*api.dll*) e richiama la funzione epunch_initialize () passando in input i parametri descritti in § 4.5.2.3

4. Se la chiamata precedente non ritorna errore (vedi § 4.5.2.3) il software della marcatrice richiede di specificare il formato del marchio da utilizzare per la marcatura laser in procinto di lavorazione.
   Il software può anche richiedere la selezione dell'eventuale titolo legale da utilizzare.
   In questa fase possono essere selezionati più file di marchi identificativi come anche più file dei titoli legali.

5. Il software della marcatrice, per ogni file specificato allo step precedente, richiama la funzione epunch_open () passando in input i parametri descritti in § 4.5.2.5

6. Il software della marcatrice, per ogni file correttamente aperto, registra i valori degli handle e richiama la funzione epunch_read () passando in input i parametri descritti in § 4.5.2.7.
   In questa fase i valori letti vengono salvati in variabili collegate alla MEMORIA VOLATILE del PC

   I FILE LETTI IN QUESTA MODALITA' NON DOVRANNO ESSERE SALVATI SU DISCO, DUPLICATI O UTILIZZATI PER ALTRE FINALITA' MA DOVRANNO PERMANERE NELLA MEMORIA VOLATILE DEL PC CHE PILOTA LA MARCATRICE SOLO ED ESCLUSIVAMENTE PER L'ARCO DI TEMPO NECESSARIO AL LORO UTILIZZO. OGNI SALVATAGGIO SU DISCO O UTILIZZO

DIVERSO DALLA MARCATURA DEI PREDETTI FILE RAPPRESENTA UN USO IMPROPRIO E NON AUTORIZZATO DEL MARCHIO.

7. La marcatrice utilizza il file memorizzati nella MEMORIA VOLATILE del PC per marcare i lotti di metallo prezioso

8. Il software della marcatrice mantiene registrazione degli handle in uso

3.3.2 Scollegamento del Token USB e chiusura della sessione di lavoro

Per la corretta chiusura di una sessione di lavoro con il Token USB e successivo scollegamento dalla marcatrice è necessario seguire la procedura descritta di seguito:

1. Il software della marcatrice, per ogni handle registrato in fase d'utilizzo del Token USB, chiama la funzione epunch_close ().

2. Completata la sequenza di chiamate epunch_close (), DEVE essere effettuata la chiamata epunch_finalize() per chiudere correttamente la sessione di lavoro con il dispositivo e garantire l'integrità della suo contenuto.

   SE NON VIENE EFFETTUATA LA CHIAMATA ALLA epunch_finalize() IL TOKEN NON POTRA' PIU' ESSERE UTILIZZATO DALLE MARCATRICI

3. Il Token USB DEVE essere scollegato dal PC pilota della marcatrice seguendo la procedura di disconnessione hardware del particolare Sistema Operativo in uso

Per maggiori informazioni è comunque possibile far riferimento all'esempio di codice riportato in allegato (§ 5).

3.3.3 Gestione delle anomalie

In caso si verifichi un'anomalia nel funzionamento del token USB, le funzioni contenute nella libreria `api.dll`, restituiranno degli opportuni codici di errore o codici di warning come previsto in § 4.5.

AL VERIFICARSI DI UN'ANOMALIA, IL SOFTWARE DELLA MARCATRICE DOVRÀ SEGNALARE ALL'UTENTE L'ACCADUTO RIPORTANDO A VIDEO UN OPPORTUNO MESSAGGIO ESPLICATIVO. IL TESTO DEL MESSAGGIO ESPLICATIVO ASSOCIATO AD OGNUNO DEI POSSIBILI CODICI DI ERRORE/WARNING RESTITUITI DALLE FUNZIONI DELLA LIBRERIA E DA PRESENTARE ALL'UTENTE È DESCRITTO NELLA TABELLA PRESENTE IN § 5.4.

4 Specifiche di Interfaccia

4.1 Definizioni di tipo

Ridefinizioni di tipo (typedef)

- typedef int epunch_error_t
- typedef int epunch_handle_t

4.1.1 Descrizione dettagliata

Questa libreria utilizza soltanto due definizioni di tipo per gestire i codici di errore e gli handle ai file immagine presenti sul token.

4.1.2 Documentazione delle ridefinizioni di tipo (typedef)

4.1.2.1 typedef int epunch_error_t

Tipo per i codici di errore del modulo epunch.

4.1.2.2 typedef int epunch_handle_t

Tipo per gli handle a file nel modulo epunch.

4.2 Codici di errore

Definizioni

- #define EPUNCH_ERROR_OK (0)
- #define EPUNCH_ERROR_NOT_INITIALIZED (1)
- #define EPUNCH_ERROR_ALREADY_INITIALIZED (2)
- #define EPUNCH_ERROR_CORRUPTED_BINARY (3)
- #define EPUNCH_ERROR_MACHINE_ID_INVALID (4)
- #define EPUNCH_ERROR_CORRUPTED_DATA (5)
- #define EPUNCH_ERROR_CORRUPTED_LOG (6)
- #define EPUNCH_ERROR_EXPIRED_CERTIFICATE (7)
- #define EPUNCH_ERROR_MISSING_SECURITY_CHIP (8)
- #define EPUNCH_ERROR_CORRUPTED_SECURITY_CHIP (9)
- #define EPUNCH_ERROR_PIN_INCORRECT (10)
- #define EPUNCH_ERROR_PIN_INVALID (11)

- #define EPUNCH_ERROR_PIN_LOCKED (12)
- #define EPUNCH_ERROR_GENERIC_SECURITY_CHIP (13)
- #define EPUNCH_ERROR_FILE_NOT_FOUND (14)
- #define EPUNCH_ERROR_FILE_CORRUPTED (15)
- #define EPUNCH_ERROR_FILE_BAD_ENCRYPTION (16)
- #define EPUNCH_ERROR_BAD_FILE_HANDLE (17)
- #define EPUNCH_ERROR_GENERIC_FILE (18)
- #define EPUNCH_ERROR_GENERIC (19)
- #define EPUNCH_ERROR_LOG_SYSTIME_MISMATCH (20)

4.2.1 Descrizione dettagliata

Tutte le funzioni di questa libreria ritornano dei codici di errore (con il tipo epunch_error_t). Questi codici di errore possono essere trasformati programmaticamente in una stringa leggibile o stampati con le funzioni epunch_strerror e epunch_perror.

4.2.2 Documentazione delle definizioni

4.2.2.1 #define EPUNCH_ERROR_OK (0)

La funzione è terminata con successo. Non si è verificato alcun errore.

4.2.2.2 #define EPUNCH_ERROR_NOT_INITIALIZED (1)

La libreria non è stata inizializzata con la funzione epunch_initialize.

4.2.2.3 #define EPUNCH_ERROR_ALREADY_INITIALIZED (2)

La libreria è stata già inizializzata con la funzione epunch_initialize. Non va inizializzata nuovamente.

4.2.2.4 #define EPUNCH_ERROR_CORRUPTED_BINARY (3)

Il file della libreria non è integro.

4.2.2.5 #define EPUNCH_ERROR_MACHINE_ID_INVALID (4)

Il codice identificativo di macchina non è valido. Riferirsi alla descrizione della funzione epunch_initialize per conoscere i valori ammissibili per il codice identificativo di macchina.

4.2.2.6 #define EPUNCH_ERROR_CORRUPTED_DATA (5)

Il file dati presenti sul token non sono integri.

4.2.2.7 #define EPUNCH_ERROR_CORRUPTED_LOG (6)

Il file di log presente sul token non è integro.

4.2.2.8 #define EPUNCH_ERROR_EXPIRED_CERTIFICATE (7)

Il certificato presente sul chip di sicurezza del token è scaduto.

4.2.2.9 #define EPUNCH_ERROR_MISSING_SECURITY_CHIP (8)

Il chip di sicurezza è assente sul token o non funzionante.

4.2.2.10 #define EPUNCH_ERROR_CORRUPTED_SECURITY_CHIP (9)

Il contenuto del chip di sicurezza non è integro.

4.2.2.11 #define EPUNCH_ERROR_PIN_INCORRECT (10)

Il codice PIN fornito per l'autenticazione sul chip di sicurezza non è corretto. Attenzione: dopo tre tentativi con codici errati il token risulterà bloccato.

4.2.2.12 #define EPUNCH_ERROR_PIN_INVALID (11)

Il codice PIN fornito per l'autenticazione non è valido. I codici PIN sono composti da soli caratteri numerici. La loro lunghezza è compresa tra sei e otto caratteri.

4.2.2.13 #define EPUNCH_ERROR_PIN_LOCKED (12)

Il codice PIN del token è ormai bloccato. A questa situazione si perviene dopo tre tentativi con codici errati.

4.2.2.14 #define EPUNCH_ERROR_GENERIC_SECURITY_CHIP (13)

Errore generico sul chip di sicurezza. Il verificarsi di questo errore con una certa frequenza può essere indice di un malfunzionamento del chip di sicurezza.

4.2.2.15 #define EPUNCH_ERROR_FILE_NOT_FOUND (14)

Il file che si sta tentando di aprire non è presente sul token.

4.2.2.16 #define EPUNCH_ERROR_FILE_CORRUPTED (15)

Il file che si sta tentando di aprire non è integro. Questo errore si verifica quando la firma sul file immagine che si sta aprendo non è valida.

4.2.2.17 #define EPUNCH_ERROR_FILE_BAD_ENCRYPTION (16)

Il file che si sta tentando di aprire non è decifrabile. Questo errore potrebbe essere indice di un cattivo funzionamento del chip di sicurezza o una alterazione dei file dati presenti sul token.

4.2.2.18 #define EPUNCH_ERROR_BAD_FILE_HANDLE (17)

Il file handle non è valido. Potrebbe corrispondere ad un file già chiuso.

4.2.2.19 #define EPUNCH_ERROR_GENERIC_FILE (18)

Errore generico nell'accesso ai file. Il verificarsi di questo errore con una certa frequenza può essere indice di un malfunzionamento della memoria del token.

4.2.2.20 #define EPUNCH_ERROR_GENERIC (19)

Errore generico. Il verificarsi di questo errore con una certa frequenza può essere indice di un malfunzionamento del token.

4.2.2.21 #define EPUNCH_ERROR_LOG_SYSTIME_MISMATCH (20)

L'ultima timestamp riportata nel file di log risulta essere posteriore alla data e all'ora del sistema.

4.3 Codici di warning

Definizioni

- #define EPUNCH_WARNING_EXPIRING_CERTIFICATE (-1)

4.3.1 Descrizione dettagliata

Alcune funzioni di questa libreria ritornano dei codici di warning (con il tipo epunch_error_t). Questi codici di warning possono essere trasformati programmaticamente in una stringa leggibile o stampati con le funzioni epunch_strerror e epunch_perror.

4.3.2 Documentazione delle definizioni

4.3.2.1 #define EPUNCH_WARNING_EXPIRING_CERTIFICATE (-1)

Il certificato presente sul chip di sicurezza del token ha una validità temporale residua minore di 45 giorni.

4.4 Tipi di file immagine da aprire

Definizioni

- #define EPUNCH_TYPE_ID (0)
- #define EPUNCH_TYPE_TITLES (1)

4.4.1 Descrizione dettagliata

Le costanti definite in questa sezione vengono utilizzate nella funzione epunch_open per specificare il tipo di file immagine che si desidera aprire. A seconda del valore specificato la funzione cercherà il file in una differente cartella e con una differente estensione e formato.

4.4.2 Documentazione delle definizioni

4.4.2.1 #define EPUNCH_TYPE_ID (0)

Il file è di tipo firmato e cifrato. La funzione epunch_open lo cercherà nella cartella `/imgs_id` contenuta nella radice del token e aggiungerà in automatico al nome del file specificato le due estensioni di formato per la firma digitale e la crittografia `.p7m.p7e`.

4.4.2.2 #define EPUNCH_TYPE_TITLES (1)

Il file è di tipo in chiaro. La funzione epunch_open lo cercherà nella cartella `/imgs_titles` contenuta nella radice del token e non aggiungerà al nome del file specificato nella chiamata alcuna ulteriore estensione.

4.5 Funzioni

Funzioni

- const char * epunch_strerror (epunch_error_t error)
- void epunch_perror (epunch_error_t error, const char *message)
- epunch_error_t epunch_initialize (const char *machine_id, const char *pin)
- epunch_error_t epunch_finalize ()
- epunch_error_t epunch_open (const char *filename, int type, epunch_handle_t *pfh)
- epunch_error_t epunch_close (epunch_handle_t fh)
- epunch_error_t epunch_read (epunch_handle_t fh, void *buf, size_t *count)
- epunch_error_t epunch_read (epunch_handle_t fh, void *buf, size_t *count)
- epunch_error_t epunch_get_sc_serial (char atr[])

4.5.1 Descrizione dettagliata

Tutte le funzioni di questa libreria ritornano dei codici di errore (con il tipo epunch_error_t). Questi codici di errore possono essere trasformati programmaticamente in una stringa leggibile o stampati con le funzioni epunch_strerror e epunch_perror.

4.5.2 Documentazione delle funzioni

4.5.2.1 const char epunch_strerror (* epunch_error_t *error)*

Ritorna una stringa con la descrizione testuale di un codice di errore.

Parametri

in	*error*	Codice errore (vedi § 4.2)

Restituisce

La descrizione testuale dell'errore.
Questa funzione ritorna un puntatore ad una stringa che descrive il codice di errore passato con l'argomento `error`.

4.5.2.2 void epunch_perror (epunch_error_t *error,* const char **message)*

Stampa un messaggio di errore.

Parametri

in	*error*	Codice errore (vedi § 4.2)
in	*message*	Messaggio aggiuntivo da stampare.

Restituisce
Nulla

Questa funzione permette di stampare il messaggio contenuto nel parametro `message` seguito da due punti (:) e da una descrizione testuale dell'errore contenuto nel parametro `error`. Il messaggio sarà inviato allo stream di errore.

4.5.2.3 epunch_error_t epunch_initialize (const char **machine_id,* const char **pin)*

Inizializza la libreria.

Parametri

in	*machine_id*	Identificativo della macchina punzionatrice (stringa null-termined).
in	*pin*	Codice PIN del token (stringa null-termined).

Restituisce
Un codice errore (vedi § 4.2).

Questa funzione inizializza la libreria `epunch`, esegue dei test diagnostici sullo stato del token e del suo chip di sicurezza quindi autentica il software che invoca la libreria attraverso il PIN passato come argomento. È necessario invocare questa funzione prima di qualsiasi altra funzione della libreria altrimenti si riceverà il messaggio di errore EPUNCH_ERROR_NOT_INITIALIZED. Le uniche eccezioni sono le funzioni epunch_strerror e epunch_perror che non richiedono che la libreria sia inizializzata.
Il parametro `machine_id` è una stringa che contiene alcuni parametri significativi della macchina punzonatrice che invoca la libreria come il suo ID, la marca e il modello. Viene riportato in maniera identica nei log prodotti sul token. Tale stringa può contenere solo caratteri alfabetici (maiuscoli e minuscoli), caratteri numerici e i simboli trattino ('-', ASCII 45), punto ('.', ASCII 46) e barra ('/', ASCII 47). Qualora la stringa passata contenga caratteri non validi, la funzione ritornerà l'errore EPUNCH_ERROR_MACHINE_ID_INVALID.

Il PIN passato con il parametro `pin` è il numero autorizzativo che consente al chip di sicurezza di decrittare i file immagine presenti nel token. I PIN hanno una lunghezza tra i sei e gli otto caratteri numerici. Valori del PIN con lunghezze non in questo intervallo o contenenti caratteri non numerici sono non validi e faranno restituire alla funzione l'errore EPUNCH_ERROR_PIN_INVALID. Se il PIN utilizzato è valido ma errato la funzione restituirà l´errore EPUNCH_ERROR_PIN_INCORRECT. Al terzo tentativo di inizializzazione della libreria con un PIN errato la funzione restituirà l'errore EPUNCH_ERROR_PIN_LOCKED ed il token passerà in uno stato di blocco dal quale non sarà possibile uscire se non utilizzando il codice PUK del token attraverso un apposito software di diagnostica. Durante la fase di diagnostica, nel caso in cui vengano riscontrati problemi con il token o il chip di sicurezza la libreria ritornerà gli errori EPUNCH_ERROR_CORRUPTED_BINARY (binario della libreria non integro), EPUNCH_ERROR_EXPIRED_CERTIFICATE (il certificato sul chip di

sicurezza è scaduto), EPUNCH_ERROR_MISSING_SECURITY_CHIP (chip di sicurezza mancante o non funzionante), EPUNCH_ERROR_CORRUPTED_SECURITY_CHIP (chip di sicurezza non integro), EPUNCH_ERROR_CORRUPTED_LOG (file di log non integro), EPUNCH_ERROR_CORRUPTED_DATA (file del token non integri), EPUNCH_ERROR_LOG_SYSTIME_MISMATCH (ultima timestamp riportata nel log posteriore a data e ora di sistema). Può essere inoltre restituito il warning EPUNCH_WARNING_EXPIRING_CERTIFICATE (certificato con meno di 45 giorni alla scadenza).

4.5.2.4 epunch_error_t epunch_finalize ( )

Finalizza la libreria.

Restituisce
Un codice di errore (vedi § 4.2).

Questa funzione finalizza la libreria `epunch`. Dopo aver invocato questa funzione la libreria si troverà nello stato non inizializzata e sarà pertanto necessario chiamare nuovamente la epunch_initialize per poter usare la libreria.

4.5.2.5 epunch_error_t epunch_open (const char ** filename,* int *type,* epunch_handle_t **pfh)*

Apre un file immagine dalla memoria del token.

Parametri

in	*filename*	Nome del file da aprire per la lettura.
in	*type*	Tipo del file immagine da aprire per la lettura (vedi Tipi di file immagine da aprire).
out	*pfh*	Puntatore alla memoria nella quale sarà scritto lo handle da utilizzare con la funzione epunch_open.

Restituisce
Un codice errore (vedi § 4.2).

Questa funzione consente di aprire (per poi leggere successivamente con la funzione epunch_read) un file immagine dal token. Il parametro `filename` specifica il nome del file immagine che si desidera aprire. Non è necessario specificare il percorso poiché verrà automaticamente calcolato dalla libreria in funzione del parameteo `type`. Qualora il file immagine sia firmato e cifrato non sarà nemmeno necessario specificare le estensioni aggiuntive per tale formato. Verranno aggiunte automaticamente dalla funzione. Qualora il file richiesto non dovesse essere presente sul token la funzione ritornerà l'errore EPUNCH_ERROR_FILE_NOT_FOUND. Nel caso in cui il file fosse di tipo firmato e cifrato la funzione potrebbe ritornare gli errori EPUNCH_ERROR_FILE_BAD_ENCRYPTION e EPUNCH_ERROR_FILE_CORRUPTED rispettivamente nel caso in cui non fosse possibile decrittare il file o nel caso in cui, una volta decrittato non dovesse essere valida la firma digitale.
La funzione ritornerà, in caso di successo, uno handle (da utilizzare successivamente con le funzioni epunch_read ed epunch_close) nella memoria puntata dal parametro `pfh`.

4.5.2.6 epunch_error_t *epunch_close (* epunch_handle_t *fh)*

Chiude un file immagine

Parametri

in	*fh*	Handle del file da chiudere.

Restituisce
Un codice errore (vedi § 4.2).

Questa funzione chiude un file aperto con la epunch_open. Se lo handle passato è non valido si otterrà l'errore EPUNCH_ERROR_BAD_FILE_HANDLE.

4.5.2.7 epunch_error_t *epunch_read (* epunch_handle_t *fh,* void **buf,* size_t **count)*

Legge il contenuto di un file immagine.

Parametri

in	*fh*	Handle del file dal quale leggere.
in	*buf*	Puntatore al buffer nel quale la funzione scriverà i byte letti.
in, out	*count*	Puntatore ad un valore `size_t` che conterrà in ingresso il numero massimo di byte che possono essere letti dalla funzione e in uscita i byte effettivamente letti.

Restituisce
Un codice errore (vedi § 4.2).

Questa funzione legge il contenuto di un file immagine precedentemente aperto con la funzione epunch_open. I byte letti verranno scritti in un buffer di memoria allocato da chi effettua l'invocazione. La dimensione del buffer deve essere passata attraverso il parametro `count`. Il numero di byte effettivamente letti verranno restituiti nuovamente nel parametro `count`. È possibile leggere il file in più passi. Se il file è più lungo del valore passato con il parametro `count` è possibile leggere la parte rimanente con ulteriori chiamate alla funzione. Quando il numero di byte letti è inferiore a quanto richiesto c'è la certezza che il file è stato letto completamente e non saranno necessarie ulteriori invocazioni della funzione. Se lo handle passato alla funzione non è valido si otterrà l'errore EPUNCH_ERROR_BAD_FILE_HANDLE.

*4.5.2.8 epunch_error_t epunch_get_filename (char **filename)*

Legge i nomi dei file nelle cartelle `/imgs_id` e `/imgs_titles`.

Parametri

in, out	*filename*	Puntatore a puntatore a carattere, inizialmente impostato a NULL.

Restituisce
Un codice errore (vedi § 4.2).

Questa funzione restituisce, ad ogni chiamata, in maniera sequenziale, il nome di un file letto dalla memoria del token, ovvero dalle cartelle `/imgs_id` e `/imgs_titles` (esattamente in quest'ordine; è possibile determinare da quale delle due sia stato letto il file sulla base del prefisso del suo nome).

Inizialmente, la funzione deve essere invocata con un puntatore ad un puntatore a NULL: al termine di ogni invocazione, il puntatore avrà come valore l'indirizzo di una stringa contenente il nome del file appena letto, oppure NULL se l'elenco dei file è terminato. Tale indirizzo è generato con un'allocazione dinamica di memoria effettuata internamente alla libreria e può essere considerato valido fino alla successiva invocazione della funzione per l'ottenimento del nome del file seguente: a quel punto, la memoria precedentemente allocata viene liberata e un nuovo buffer (con un indirizzo diverso) viene predisposto per contenere il nuovo nome del file letto. E' quindi opportuno che la stringa ottenuta di volta in volta, ad ogni iterazione, venga utilizzata prima della successiva chiamata alla funzione o che venga copiata in un'area di memoria locale all'applicazione chiamante.

Se la funzione viene eseguita con successo, viene restituito il valore EPUNCH_ERROR_OK, altrimenti EPUNCH_ERROR_NOT_INITIALIZED (in caso di libreria non inizializzata) o EPUNCH_ERROR_GENERIC_FILE (per qualsiasi altro errore).

4.5.2.9 epunch_error_t epunch_get_sc_serial (char atr[])

Legge il seriale della smart card.

Parametri

out	*atr*	Buffer di 17 caratteri ASCII a 8 bit, allocato dal chiamante, dove viene scritto il valore del numero seriale della smart card.

Restituisce
Un codice errore (vedi § 4.2).

Questa funzione consente di leggere il numero seriale della smart card inserita nel token. Il parametro `atr` rappresenta un buffer di 17 caratteri ASCII (16 per il seriale più 1 per il terminatore) che deve essere allocato da chi invoca la funzione e che conterrà, al termine della esecuzione della routine, il numero seriale della smart card (terminato da '\0'). Nel caso che il buffer sia più lungo di 17 caratteri la parte eccedente non viene utilizzata.
Se la funzione viene eseguita con successo, viene restituito il valore EPUNCH_ERROR_OK; se si richiama la funzione prima dell'inizializzazione della libreria, viene restituito il valore EPUNCH_ERROR_NOT_INITIALIZED; se si verifica un errore di comunicazione con la smart card, viene restituito il valore EPUNCH_ERROR_GENERIC_SECURITY_CHIP.

5 Allegati

5.1 Allegato 1 – Esempio di codice d'uso del Token USB

```
#include <stdio.h>
#include <stdlib.h>
#include "epunch.h"

int main() {
  epunch_error_t err;
  int fd;
  const size_t buf_size = 65536;
  char *buf;
  size_t count;

  if ((err = epunch_initialize("TEST01", "12345678"))) {
    epunch_perror(err, "Cannot initialize the library");
    epunch_finalize();
    exit(1);
  }

  if ((err = epunch_open("img1", &fd))) {
    epunch_perror(err, "Cannot open the image file");
    epunch_finalize();
    exit(1);
  }

  if ((buf = malloc(buf_size)) == NULL) {
    fprintf(stderr, "Cannot allocate memory\n");
    epunch_close(fd);
    epunch_finalize();
    exit(1);
  }

  do {
    count = buf_size;
    if ((epunch_read(fd, buf, &count))) {
      epunch_perror(err, "Cannot read the image file");
      free(buf);
      epunch_close(fd);
      epunch_finalize();
      exit(1);
    }
    /* do something with the "buf" */
  } while(count == buf_size);

  free(buf);
  epunch_close(fd);
  epunch_finalize();
```

```
  return 0;
}
```

5.2 Allegato 2 – File header per integrazione della libreria api.dll

```
/**
 * @file  epunch.h
 * @brief Header file per l'utilizzo delle funzioni del modulo epunch.
 *
 * Questo header file C contiene tutte le definizioni delle funzioni,
tipi e
 * macro per necessarie per l'utilizzo del modulo epunch.
 */

#ifndef _EPUNCH_H
#define _EPUNCH_H

#include <stddef.h>

/** @defgroup EPUNCH_TYPEDEF Definizioni di tipo
 *
 * Questa libreria utilizza soltanto due definizioni di tipo per
gestire
 * i codici di errore e gli handle ai file immagine presenti sul
token.
 * @{
 */

/** Tipo per i codici di errore del modulo epunch. */
typedef int epunch_error_t;


/** Tipo per gli handle a file nel modulo epunch. */
typedef int epunch_handle_t;

/* @} */

/** @defgroup EPUNCH_ERROR_ Codici di errore
 *
 * Tutte le funzioni di questa libreria ritornano dei codici di errore
 * (con il tipo #epunch_error_t ). Questi codici di errore possono
 * essere trasformati programmaticamente in una stringa leggibile o
 * stampati con le funzioni #epunch_strerror e #epunch_perror.
 * @{
 */

/** La funzione è terminata con successo. Non si è verificato alcun
errore. */
#define EPUNCH_ERROR_OK (0)

/** La libreria non è stata inizializzata con la funzione
```

```
 *  #epunch_initialize. */
#define EPUNCH_ERROR_NOT_INITIALIZED (1)

/** La libreria è stata già inizializzata con la funzione
 *  #epunch_initialize. Non va inizializzata nuovamente. */
#define EPUNCH_ERROR_ALREADY_INITIALIZED (2)

/** Il file della libreria non è integro. */
#define EPUNCH_ERROR_CORRUPTED_BINARY (3)

/** Il codice identificativo di macchina non è valido. Riferirsi alla
 *  descrizione della funzione #epunch_initialize per conoscere i
 *  valori ammissibili per il codice identificativo di macchina. */
#define EPUNCH_ERROR_MACHINE_ID_INVALID (4)

/** Il file dati presenti sul token non sono integri. */
#define EPUNCH_ERROR_CORRUPTED_DATA (5)

/** Il file di log presente sul token non è integro. */
#define EPUNCH_ERROR_CORRUPTED_LOG (6)

/** Il certificato presente sul chip di sicurezza del token è scaduto.
*/
#define EPUNCH_ERROR_EXPIRED_CERTIFICATE (7)

/** Il chip di sicurezza è assente sul token o non funzionante. */
#define EPUNCH_ERROR_MISSING_SECURITY_CHIP (8)

/** Il contenuto del chip di sicurezza non è integro. */
#define EPUNCH_ERROR_CORRUPTED_SECURITY_CHIP (9)

/** Il codice PIN fornito per l'autenticazione sul chip di sicurezza
non è
 *  corretto. Attenzione: dopo tre tentativi con codici errati il
token
 *  risulterà bloccato. */
#define EPUNCH_ERROR_PIN_INCORRECT (10)

/** Il codice PIN fornito per l'autenticazione non è valido. I codici
PIN
 *  sono composti da soli caratteri numerici. La loro lunghezza è
compresa
 *  tra sei e otto caratteri. */
#define EPUNCH_ERROR_PIN_INVALID (11)

/** Il codice PIN del token è ormai bloccato. A questa situazione si
perviene
 *  dopo tre tentativi con codici errati. */
#define EPUNCH_ERROR_PIN_LOCKED (12)

/** Errore generico sul chip di sicurezza. Il verificarsi di questo
errore
 *  con una certa frequenza può essere indice di un malfunzionamento
del chip
```

```
 *  di sicurezza. */
#define EPUNCH_ERROR_GENERIC_SECURITY_CHIP (13)

/** Il file che si sta tentando di aprire non è presente sul token. */
#define EPUNCH_ERROR_FILE_NOT_FOUND (14)

/** Il file che si sta tentando di aprire non è integro. Questo errore
si
 *  verifica quando la firma sul file immagine che si sta aprendo non
è
 *  valida. */
#define EPUNCH_ERROR_FILE_CORRUPTED (15)

/** Il file che si sta tentando di aprire non è decifrabile.
 *  Questo errore potrebbe essere indice di un cattivo funzionamento
del chip
 *  di sicurezza o una alterazione dei file dati presenti sul token.
*/
#define EPUNCH_ERROR_FILE_BAD_ENCRYPTION (16)

/** Il file handle non è valido. Potrebbe corrispondere ad un file già
chiuso.
 *  */
#define EPUNCH_ERROR_BAD_FILE_HANDLE (17)

/** Errore generico nell'accesso ai file. Il verificarsi di questo
errore
 *  con una certa frequenza può essere indice di un malfunzionamento
della
 *  memoria del token. */
#define EPUNCH_ERROR_GENERIC_FILE (18)

/** Errore generico. Il verificarsi di questo errore con una certa
frequenza
 *  può essere indice di un malfunzionamento del token. */
#define EPUNCH_ERROR_GENERIC (19)

/** @} */

/** @defgroup EPUNCH_TYPE_ Tipi di file immagine da aprire
 *
 * Le costanti definite in questa sezione vengono utilizzate nella
 * funzione #epunch_open per specificare il tipo di file immagine che
 * si desidera aprire. A seconda del valore specificato la funzione
 * cercherà il file in una differente cartella e con una differente
 * estensione e formato.  @{
 */

/** Il file è di tipo firmato e cifrato. La funzione #epunch_open lo
 *  cercherà nella cartella @c /imgs_id contenuta nella radice del
 *  token e aggiungerà in automatico al nome del file specificato le
 *  due estensioni di formato per la firma digitale e la crittografia
 *  <tt>.p7m.p7e</tt>. */
#define EPUNCH_TYPE_ID (0)
```

```
/** Il file è di tipo in chiaro. La funzione #epunch_open lo cercherà
 *  nella cartella @c /imgs_titles contenuta nella radice del token e
 *  non aggiungerà al nome del file specificato nella chiamata alcuna
 *  ulteriore estensione. */
#define EPUNCH_TYPE_TITLES (1)

/** @} */

/** @defgroup EPUNCH_FUNCTIONS Funzioni
 *
 * Tutte le funzioni di questa libreria ritornano dei codici di errore
 * (con il tipo ::epunch_error_t). Questi codici di errore possono
 * essere trasformati programmaticamente in una stringa leggibile o
 * stampati con le funzioni #epunch_strerror e #epunch_perror.
 * @{
 */

/** Ritorna una stringa con la descrizione testuale di un codice di
errore.
 *  @param[in] error    Codice di errore (vedi @ref EPUNCH_ERROR_).
 *  @returns La descrizione testuale del codice di errore.
 *
 *  Questa funzione ritorna un puntatore ad una stringa che descrive
il codice
 *  di errore passato con l'argomento @p error.
 */
const char *epunch_strerror(epunch_error_t error);


/** Stampa un messaggio di errore.
 *  @param[in] error     Codice di errore.
 *  @param[in] message   Messaggio aggiuntivo da stampare.
 *  @returns Nulla
 *
 *  Questa funzione permette di stampare il messaggio contenuto nel
 *  parametro @p message seguito da due punti (:) e da una descrizione
 *  testuale dell'errore contenuto nel parametro @p error.  Il
 *  messaggio sarà inviato allo stream di errore.
 */
void epunch_perror(epunch_error_t error, const char *message);


/** Inizializza la libreria.
 *  @param[in] machine_id   Stringa contenente parametri specifici
della
                            macchina punzionatrice come ID, marca e
modello
                            (stringa null-termined).
 *  @param[in] pin          Codice PIN del token (stringa null-
termined).
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 *  Questa funzione inizializza la libreria @c epunch, esegue dei test
```

```
 *  diagnostici sullo stato del token e del suo chip di sicurezza
 *  quindi autentica il software che invoca la libreria attraverso il
 *  PIN passato come argomento. È necessario invocare questa funzione
 *  prima di qualsiasi altra funzione della libreria altrimenti si
 *  riceverà il messaggio di errore #EPUNCH_ERROR_NOT_INITIALIZED. Le
 *  uniche eccezioni sono le funzioni #epunch_strerror e
 *  #epunch_perror che non richiedono che la libreria sia
 *  inizializzata.
 *
 *  Il parametro @p machine_id è una stringa che contiene alcuni
parametri
 *  significativi della macchina punzionatrice che invoca la libreria
come
 *  il suo ID, la marca e il modello. Viene riportato in maniera
identica
 *  nei log prodotti sul token. Tale stringa può contenere solo
caratteri
 *  alfabetici (maiuscoli e minuscoli), caratteri numerici e i simboli
trattino
 *  (<tt>'-'</tt>, ASCII 45), punto (<tt>'.'</tt>, ASCII 46) e
 *  barra (<tt>'/'</tt>, ASCII 47). Qualora la stringa passata
contenga
 *  caratteri non validi, la funzione ritornerà l'errore
 *  #EPUNCH_ERROR_MACHINE_ID_INVALID.
 *
 *  Il PIN passato con il parametro @p pin è il numero autorizzativo
 *  che consente al chip di sicurezza di decrittare i file immagine
 *  presenti sul token. I PIN hanno una lunghezza tra i sei e gli otto
 *  caratteri numerici. Valori del PIN con lunghezze non in questo
 *  intervallo o contenenti caratteri non numerici sono non validi e
 *  faranno restituire alla funzione l'errore
 *  EPUNCH_ERROR_PIN_INVALID. Se il PIN utilizzato è valido ma errato
 *  la funzione restituirà l´errore EPUNCH_ERROR_PIN_INCORRECT. Al
 *  terzo tentativo di inizializzazione della libreria con un PIN
 *  errato la funzione restituirà l'errore EPUNCH_ERROR_PIN_LOCKED ed
 *  il token passerà in uno stato di blocco dal quale non sarà
 *  possibile uscire se non utilizzando il codice PUK del token
 *  attraverso un apposito software di diagnostica.
 *
 *  Durante la fase di diagnostica, nel caso in cui vengano
 *  riscontrati problemi con il token o il chip di sicurezza la
 *  libreria ritornerà gli errori #EPUNCH_ERROR_CORRUPTED_BINARY
 *  (binario della libreria non integro),
 *  #EPUNCH_ERROR_MISSING_SECURITY_CHIP (chip di sicurezza mancante o
 *  non funzionante), #EPUNCH_ERROR_CORRUPTED_SECURITY_CHIP (chip di
 *  sicurezza non integro), #EPUNCH_ERROR_CORRUPTED_LOG (file di log
 *  non integro), #EPUNCH_ERROR_CORRUPTED_DATA (file del token non
 *  integri).
 */
epunch_error_t epunch_initialize(const char *machine_id, const char
*pin);


/** Finalizza la libreria.
```

```
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 *  Questa funzione finalizza la libreria @c epunch. Dopo aver
 *  invocato questa funzione la libreria si troverà nello stato
 *  <em>non inizializzata</em> e sarà pertanto necessario chimare
 *  nuovamente la #epunch_initialize per poter usare la libreria.
 */
epunch_error_t epunch_finalize();

/** Legge il seriale della smart card.
 *  @param[out] atr Buffer di 17 caratteri ASCII a 8 bit, allocato dal
 *                  chiamante, dove viene scritto il valore del
 *                  numero seriale della smart card.
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 *  Questa funzione consente di leggere il numero seriale della smart
card
 *  inserita nel token. Il parametro @p atr rappresenta un buffer di
 *  17 caratteri ASCII (16 per il seriale più 1 per il terminatore)
che deve
 *  essere allocato da chi invoca la funzione e che conterrà, al
termine della
 *  esecuzione della routine, il numero seriale della smart card
 *  (terminato da '\0'). Nel caso che il buffer sia più lungo di 17
caratteri,
 *  la parte eccedente non viene utilizzata.

 *  Se la funzione viene eseguita con successo, viene restituito il
valore
 *  #EPUNCH_ERROR_OK; se si richiama la funzione prima
dell'inizializzazione
 *  della libreria, viene restituito il valore
#EPUNCH_ERROR_NOT_INITIALIZED;
 *  se si verifica un errore di comunicazione con la smart card, viene
 *  restituito il valore #EPUNCH_ERROR_GENERIC_SECURITY_CHIP.
 */
epunch_error_t epunch_get_sc_serial(char atr[]);

/** Legge i nomi dei file nelle cartelle @c /imgs_id e @c /imgs_titles
 *  @param[in,out]  filename    Puntatore a puntatore a carattere,
inizialmente
 *                              impostato a NULL.
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 * Questa funzione restituisce, ad ogni chiamata, in maniera
sequenziale, il
 * nome di un file letto dalla memoria del token, ovvero dalle
cartelle
 * @c /imgs_id e @c /imgs_titles (esattamente in quest'ordine; è
possibile
 * determinare da quale delle due sia stato letto il file sulla base
 * del prefisso del suo nome).
```

```
 * Inizialmente, la funzione deve essere invocata con un puntatore ad
un
 * puntatore a NULL: al termine di ogni invocazione, il puntatore avrà
come
 * valore l'indirizzo di una stringa contenente il nome del file
appena letto,
 * oppure NULL se l'elenco dei file è terminato. Tale indirizzo è
generato
 * con un'allocazione dinamica di memoria effettuata internamente alla
libreria
 * e può essere considerato valido fino alla successiva invocazione
della
 * funzione per l'ottenimento del nome del file seguente: a quel
punto, la
 * memoria precedentemente allocata viene liberata e un nuovo buffer
(con un
 * indirizzo diverso) viene predisposto per contenere il nuovo nome
del file
 * letto. E' quindi opportuno che la stringa ottenuta di volta in
volta, ad
 * ogni iterazione, venga utilizzata prima della successiva chiamata
alla
 * funzione o che venga copiata in un'area di memoria locale
all'applicazione
 * chiamante.

 *
 */
epunch_error_t epunch_get_filename(char **filename);

/** Apre un file immagine dalla memoria del token.
 *  @param[in]  filename   Nome del file da aprire per la lettura.
 *  @param[in]  type       Tipo del file immagine da aprire per la
lettura
 *                         (vedi @ref EPUNCH_TYPE_).
 *  @param[out] pfh        Puntatore alla memoria nella quale sarà
scritto
 *                         lo handle da utilizzare con la funzione
 *                         #epunch_open.
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 *  Questa funzione consente di aprire (per poi leggere
 *  successivamente con la funzione #epunch_read) un file immagine dal
 *  token. Il parametro @p filename specifica il nome del file
 *  immagine che si desidera aprire. Non è necessario specificare il
 *  percorso poiché verrà automaticamente calcolato dalla libreria in
 *  funzione del parameteo @p type. Qualora il file immagine sia
 *  firmato e cifrato non sarà nemmeno necessario specificare le
 *  estensioni aggiuntive per tale formato. Verranno aggiunte
 *  automaticamente dalla funzione. Qualora il file richiesto non
 *  dovesse essere presente sul token la funzione ritornerà l'errore
 *  #EPUNCH_ERROR_FILE_NOT_FOUND. Nel caso in cui il file fosse di
 *  tipo firmato e cifrato la funzione potrebbe ritornare gli errori
 *  #EPUNCH_ERROR_FILE_BAD_ENCRYPTION e #EPUNCH_ERROR_FILE_CORRUPTED
```

```
 *  rispettivamente nel caso in cui non fosse possibile decrittare il
 *  file o nel caso in cui, una volta decrittato non dovesse essere
 *  valida la firma digitale.

 *  La funzione ritornerà, in caso di successo, uno handle (da
 *  utilizzare successivamente con le funzioni #epunch_read ed
 *  #epunch_close) nella memoria puntata dal parametro @p pfh.
 */
epunch_error_t epunch_open(const char *filename, int type,
                           epunch_handle_t *pfh);

/** Chiude un file immagine.
 *  @param[in] fh   Handle del file chiudere.
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 *  Questa funzione chiude un file aperto con la #epunch_open. Se lo
handle
 *  passato è non valido si otterrà l'errore
#EPUNCH_ERROR_BAD_FILE_HANDLE.
 */
epunch_error_t epunch_close(epunch_handle_t fh);


/** Legge il contenuto di un file immagine.
 *  @param[in]     fh      Handle del file dal quale leggere.
 *  @param[out]    buf     Puntatore al buffer nel quale la funzione
scriverà
 *                         i byte letti.
 *  @param[in,out] count   Puntatore ad un valore @c size_t che
conterrà in
 *                         ingresso il numero massimo di byte che
possono
 *                         essere letti dalla funzione e in uscita i
byte
 *                         effettivamente letti.
 *  @returns Un codice di errore (vedi @ref EPUNCH_ERROR_).
 *
 *  Questa funzione legge il contenuto di un file immagine
 *  precedentemente aperto con la funzione #epunch_open. I byte letti
 *  verranno scritti in un buffer di memoria allocato da chi effettua
 *  l'invocazione. La dimensione del buffer deve essere passata
 *  attraverso il parametro @p count. Il numero di byte effettivamente
 *  letti verranno restituiti nuovamente nel parametro @p count. È
 *  possibile leggere il file in più passi. Se il file è più lungo del
 *  valore passato con il parametro @p count è possibile leggere la
 *  parte rimanente con ulteriori chiamate alla funzione. Quando il
 *  numero di byte letti è inferiore a quanto richiesto c'è la
 *  certezza che il file è stato letto completamente e non saranno
 *  necessarie ulteriori invocazioni della funzione. Se lo handle
 *  passato alla funzione non è valido si otterrà l'errore
 *  #EPUNCH_ERROR_BAD_FILE_HANDLE.
 */
epunch_error_t epunch_read(epunch_handle_t fh, void *buf, size_t
*count);
```

```
      /** @} */

#endif /* _EPUNCH_H */
```

5.3 Allegato 3 – Standard PLT

5.3.1 Standard di riferimento

Lo standard di riferimento dei file custoditi nel token è il PCL-6, standard creato da HP per far comunicare tutti i plotter con un formato universalmente riconosciuto.
La realizzazione pratica del PCL-6 adottata è HP-GL/2 che prevede file con estensione .PLT.

5.3.2 Il file PLT

Il file PLT è un file di testo contenente la sequenza di istruzioni da inviare al plotter. Le istruzioni sono separate da ";", non è previsto un delimitatore di fine riga.
Il file PLT nel token contengono istruzioni che si possono suddividere in:

- istruzioni di impostazione - istruzioni che indicano come si deve disegnare il file vettoriale (vengono date all'inizio del file e sono valide per tutto il file);
- istruzioni di creazione del disegno - istruzioni che movimentano la "penna", ossia il device imputato alla scrittura (o alla marcatura nel caso del laser).

5.3.3 Istruzioni PLT

Lo standard HP-GL/2 prevede una cospicua serie di istruzioni per definire forma, dimensioni e colori della penna. Nel caso specifico, data l'universalità del file da creare, è stato utilizzato un insieme limitato e comune di istruzioni.

5.3.3.1 Formati utilizzati

Il testo del PLT è in formato standard UTF-8, quindi vengono utilizzati solo e solamente caratteri ASCII numerici e alfanumerici più i simboli di ".", "-", "," e ";".
Le coordinate utilizzano un formato comune: sono sempre numeri reali, rappresentati da un numero variabile di cifre. La virgola è rappresentata dal ".", il simbolo per le coordinate negative è il "-".
Le coordinate sono sempre in ordine "x,y" ove "," rappresenta il carattere di separazione tra i valori.
Le coordinate sono sempre assolute, non vengono utilizzati comandi di spostamento relativo.

La risoluzione standard per il disegno è di 40 pt/mm, ove pt è l'unità di riferimento della penna., quindi:
- un valore di 1.0 identifica una coordinata di 1/40 di mm;
- un valore di 40.0 identifica uno spostamento di 1.0 mm;
- un valore di 0.01 identifica una coordinata di 1/4000 mm.

5.3.3.2 Istruzioni di impostazione

Le istruzioni di impostazione sono due, IP e SP, che servono solo ad inizializzare il plotter facendolo lavorare nelle condizioni standard.

IP

- **Sintassi:** IP;
- **Descrizione:** comando di apertura del file;
- **Parametri:** nessuno;
- **Esempio:** IP;

SP

- **Sintassi:** SP[numero penna];
- **Descrizione:** comando di selezione della penna da utilizzare;
- **Parametri:** numero penna da utilizzare. Viene sempre richiesta la penna numero 1 (default);
- **Esempio:** SP1;

5.3.3.3 Istruzioni di creazione del disegno

Le istruzioni di creazione del disegno sono due, PU e PD, che servono a descrivere al plotter gli spostamenti da fare e dove utilizzare la penna.

PU

- **Sintassi:** PU[x],[y];
- **Descrizione:** comando di posizionamento senza utilizzo di penna (posizionamento assoluto);
- **Parametri:** coordinata assoluta del punto sul quale muoversi sul piano;
- **Esempio:** PU0.0089,-0.078;

PD

- **Sintassi:** PD[x0],[y0],[x1],[y1],[.....],[xn],[yn];
- **Descrizione:** comando di scrittura con la penna;
- **Parametri:** coordinate assolute da 0 a n dei movimenti da effettuare a penna attiva (marcatura);
- **Esempio:** PD0.04,0.06,0.05,0.05,0.07,0.07;

5.4 Allegato 4 – Tabella di associazione codici di errore/warning e messaggi esplicativi

Codice di errore/warning	Testo messaggio esplicativo
0	*Questo codice evidenzia che la funzione è terminata con successo. In*

	questo caso non devono essere visualizzati messaggi esplicativi all'utente.
1	La libreria non è stata inizializzata.
2	La libreria è stata già inizializzata. Non va inizializzata nuovamente.
3	Il file della libreria non è integro.
4	Il codice identificativo della marcatrice non è valido.
5	I file dati presenti nel token USB non sono integri.
6	Il file di log presente nel token non è integro.
7	Attenzione: il termine di utilizzo del token USB è stato superato e l'accesso ai contenuti del dispositivo è stato bloccato. Le istruzioni per rinnovare il token sono disponibili sul sito metrologialegale.unioncamere.it alla sezione "Metalli preziosi - tecnologia laser > Rinnovo token USB".
8	Il chip di sicurezza nel token USB è assente o non funzionante.
9	Il contenuto del chip di sicurezza non è integro.
10	Il codice PIN fornito per l'autenticazione non è corretto. Attenzione: dopo tre tentativi con codici errati il token USB risulterà bloccato.
11	Il codice PIN fornito per l'autenticazione non è valido.
12	Il codice PIN del token è bloccato.
13	Errore generico. Possibile malfunzionamento del chip di sicurezza.
14	Il file che si sta tentando di aprire non è presente nel token USB.
15	Il file che si sta tentando di aprire non è integro. La firma digitale del file non è valida.
16	Il file che si sta tentando di aprire non è decifrabile.
17	Il file handle non è valido. Potrebbe corrispondere ad un file già chiuso.
18	Errore generico nell'accesso ai file. Possibile malfunzionamento della memoria del token USB.
19	Errore generico. Possibile malfunzionamento del token.
20	Attenzione: l'orologio di sistema risulta incongruente con il tracciato di ultimo utilizzo del token USB. L'accesso ai contenuti del dispositivo è stato disabilitato. Per ripristinare il funzionamento del token modificare le impostazioni dell'orologio di sistema.
-1	Attenzione: mancano meno di 60 giorni al termine di utilizzo del token USB, i contenuti del dispositivo non saranno più accessibili a partire dal prossimo 31 gennaio. Le istruzioni per rinnovare il token entro tale data sono disponibili sul sito metrologialegale.unioncamere.it alla sezione "Metalli

	preziosi - tecnologia laser > Rinnovo token USB”.