ESIGNER – Manuale Utente

SOMMARIO

1 ESIGNER DINAMIC LINK LIBRARY

1.1 Introduzione

Il presente manuale descrive l'utilizzo delle funzioni di firma attraverso un'interfaccia di tipo DLL (Dynamic Link Library).

1.2 A chi è destinato

Programmatori e system integrator.

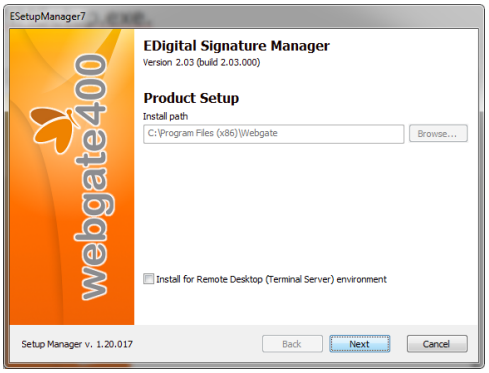
1.3 Prerequisiti

Si richiede esperienza di programmazione C/C++ oppure Visual Basic.

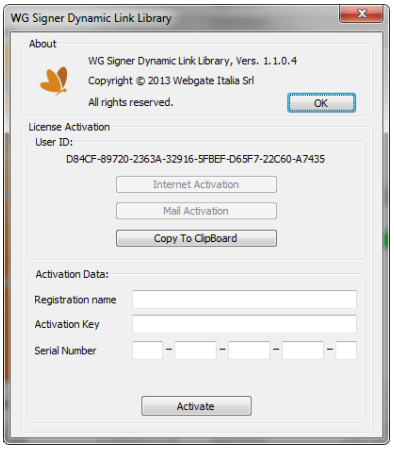
2 INSTALLAZIONE

Avviare l'installazione con il file EDIGSIGNMSetup.exe con privilegi di amministrazione.

Seguire procedura guidata accettando i dati di default proposti.

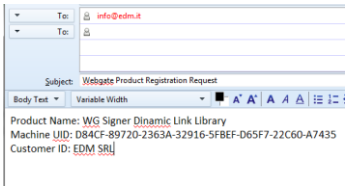


Durante la fase di installazione verrà richiesta l'attivazione del prodotto.

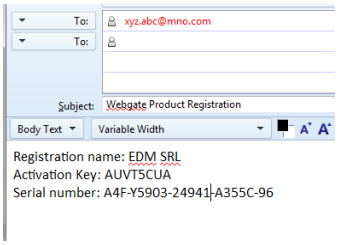


La finestra di dialogo propone il nome del prodotto da attivare e genera un ID legato al computer in uso. Questi ID deve essere comunicato ad EDM SRL per la generazione di una chiave di protezione per l'uso del prodotto.

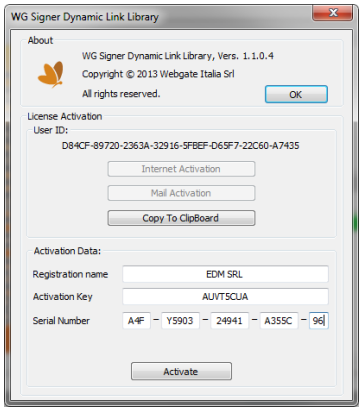
Con il tasto 'Copy to clipboard' possiamo prelevare l'ID e incollarlo nel testo di una mail da inviare ad info@edm.it, indicando il prodotto e il Customer ID assegnato, se non ancora assegnato verrà assegnato alla prima registrazione del prodotto.



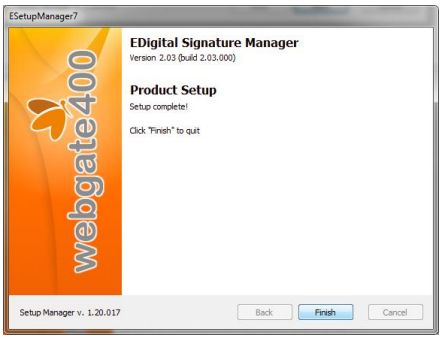
Riceverete una mail con i codici di sblocco da inserire nella finestra di dialogo.



Inserire i dati ricevuti e premere il tasto 'Activate'.



In caso non siate in possesso dei dati di sblocco la finestra comparirà ogni volta che tenterete di usare il prodotto, potrete a quel punto inserire i dati comunicati.



3 FUNZIONI DISPONIBILI IN ESIGNER

3.1 Sign

La funzione Sign consente di firmare documenti producendo in uscita un file .p7m ed opzionalmente una marca temporale.

3.1.1 Parametri

	Parametro	Tipo	Descrizione
1	Operation	Char *	Tipo di operazione richiesta ➢ S : firma file ➢ S+D : firma directory ➢ S+T : firma + marca file ➢ S+T+D : firma + marca directory
2	Method	Char *	Metodo utilizzato per la firma dei file: ➢ F :File .cer ➢ P :File .pfx ➢ T :Token (usb o smart card)
3	Input Path	Char *	Percorso del file da firmare (S) o nome directory (S+D)
4	Output Path	Char *	Percorso del file firmato (S) o nome directory (S+D)
5	TSR Output Path	Char *	Percorso del file di marca temporale (.tsr) (S+T) o nome directory (S+D+T)
6	Extention	Char *	Estensione del file firmato (.P7M)
7	CER file path	Char *	Percorso del file contenente il certificato del firmatore (Method = 'F')
8	PEM file path	Char *	Percorso del file contenente la chiave privata del firmatore (Method = 'F')
9	PFX file path	Char *	Percorso del file contenente il certificato e la chiave privata del firmatore (Method = 'P')
10	PKCS11 DLL file path	Char *	Percorso del file che identifica il driver PKCS11 da utilizzare (Method = 'T'). Normalmente il driver viene fornito in forma di DLL dal produttore del token o della smart card utilizzata
11	Passphrase	Char *	Pin o password da utilizzare per accedere alle informazioni contenute nel token o nei files (CER, PFX, etc)
12	Token cache Path	Char *	Directory di lavoro assegnata per elaborazioni effettuate dalla funzione di firma
13	Cert index on token	Char *	Indice del certificato all'interno del token da utilizzare per identificazione del firmatore, si utilizza per indirizzare il certificato corretto all'interno del token qualore ce ne fossero più di uno presente. "0" indica il primo, "1" il secondo e così via.
14	TSA URL	Char *	URL nella forma http://xx.xx.xx.xx:nnnn/abcdefg da utilizzare per accedere al servizio di marca temporale fornito da una Time Stamping Autority
15	TSA Username	Char *	Utente servizio di marca temporale
16	TSA Password	Char *	Password di accesso servizio di marca temporale
17	TSA Policy	Char *	Policy adottata dalla TSA
18	TSA Encoding	Char *	Encoding della marca fornito dalla TSA
19	RootCA Path	Char *	Percorso di memorizzazione dei certificaty delle Certification Autority

3.1.2 Valori di ritorno

Valore	Parametro	Descrizione
0	NO ERROR	Nessun errore operazione conclusa con successo
-1	TOKEN OPERATION ERROR	Errore generico di accesso alle funzioni di firma o verifica
-2	CERT READ FAILURE	Errore lettura certificato
-3	PKEY READ FAILURE	Errore lettura chiave privata

-4	PFX PARSE FAILURE	Errore lettura file PFX
-5	CERT RETRIEVE TOKEN FAILURE	Errore recupero certificato da Token
-6	FILE OPENING ERROR	Errore apertura file
-7	TSA URI CONTACT ERROR	Errore contatto TSA
-8	TSA RESPONSE ERROR	Errore risposta TSA
-9	TSA RESPONSE UNPARSABLE	Risposta TSA illeggibile
-10	TOKEN LOGIN FAILED	Accesso token fallito
-11	PKCS11 DLL NOT FOUND	Non trovata dll token
-12	TOKEN SLOT NOT FOUND	
-13	OUTPUT FILE OPENING ERROR	Errore apertura file di output
-14	OUTPUT FILE ALREADY EXISTS ERROR	File di output esiste già
-15	ERROR TIMESTAMP FILE	Errore scrittura file marca temporale
-16	TIMESTAMPING FILE OPENING ERROR	Errore apertura file marca temporale

3.2 Verify

La funzione Verify consente di verificare documenti firmati in precedenza o ricevuti firmati da terzi.

3.2.1 Parametri

	Parametro	Tipo	Descrizione
1	Operation	Char *	Tipo di operazione richiesta ➢ V : Verifica file firmato ➢ V+D : Verifica directory
2	Input Path	Char *	Percorso del file da verificare (V) o nome directory (V+D)
3	Response Path	Char *	Percorso in cui memorizzare i file di verifica (V+D)
4	RootCA Path	Char *	Percorso di memorizzazione dei certificaty delle Certification Autority
5	RootTSA Path	Char *	Percorso di memorizzazione dei certificaty delle Time Stamping Autority
6	Result	Char *	Diagnostica operazione di verifica

3.2.2 Valori di ritorno

Valore	Parametro	Descrizione
0	NO ERROR	Nessun errore operazione conclusa con successo
-1	TOKEN OPERATION ERROR	Errore generico di accesso alle funzioni di verifica
-20	FILE NOT SIGNED	File non firmato
-21	FAKE P7M STREAM	Formato P7M non valido
-22	INVALID SIGNATURE	Firma non valida
-23	CERTIFICATE REVOKED	Certificato revocato
-24	FILE OPENING ERROR	Errore apertura file
-25	SIGNED BUT TIME STAMP ERROR	Firma valida, marca temporale non valida
-30	ROOT CA CERT NOT FOUND	Certificato Root CA non trovato
-31	CRL NOT RETRIEVED	Certificate Revocation List non recuperata
-32	CERTIFICATE REVOKED BUT SIGNATURE VALID	Firma valida, ma certificato revocato
-33	SELFSIGNED CERTIFICATE	Firma valida con certificato autofirmato
-34	SIGNER CERTIFICATE EXPIRED	Certificato scaduto: firma valida se apposta durante il periodo di validità del certificato

4 ESIGNERDLL HEADER FILE

```
////////////////////////////////////////////////////////////////////////////
#ifndef _ESIGNER_DLL
#define _ESIGNER_DLL
////////////////////////////////////////////////////////////////////////////

#define ESIGNER_USE_SHA1_DIGEST                          0
#define ESIGNER_USE_SHA256_DIGEST                        1

#define ESIGNER_NO_ERROR                                 0
#define ESIGNER_TOKEN_OPERATION_ERROR                   -1
#define ESIGNER_CERT_READ_FAILURE                       -2
#define ESIGNER_PKEY_READ_FAILURE                       -3
#define ESIGNER_PFX_PARSE_FAILURE                       -4
#define ESIGNER_CERT_RETRIEVE_TOKEN_FAILURE             -5
#define ESIGNER_FILE_OPENING_ERROR                      -6
#define ESIGNER_TSA_URI_CONTACT_ERROR                   -7
#define ESIGNER_TSA_RESPONSE_ERROR                      -8
#define ESIGNER_TSA_RESPONSE_UNPARSABLE                 -9
#define ESIGNER_TOKEN_LOGIN_FAILED                      -10
#define ESIGNER_PKCS11_DLL_NOT_FOUND                    -11
#define ESIGNER_TOKEN_SLOT_NOT_FOUND                    -12
#define ESIGNER_OUTPUT_FILE_OPENING_ERROR               -13
#define ESIGNER_OUTPUT_FILE_ALREADY_EXISTS_ERROR        -14
#define ESIGNER_ERROR_TIMESTAMP_FILE                    -15
#define ESIGNER_TIMESTAMPING_FILE_OPENING_ERROR         -16
#define ESIGNER_FILE_NOT_SIGNED                         -20
#define ESIGNER_FAKE_P7M                                -21
#define ESIGNER_SIGNATURE_INVALID                       -22
#define ESIGNER_CERTIFICATE_REVOKED                     -23
#define ESIGNER_SIGNED_BUT_TIME_STAMP_ERROR             -25
#define ESIGNER_ROOT_CA_CERT_NOT_FOUND                  -30
#define ESIGNER_CRL_NOT_RETRIEVED                       -31
#define ESIGNER_CERTIFICATE_REVOKED_BUT_SIGNATURE_VALID -32
#define ESIGNER_CERTIFICATE_SELFSIGNED                  -33
#define ESIGNER_SIGNER_CERTIFICATE_EXPIRED              -34
#define ESIGNER_SIGNER_CERTIFICATE_OUT_OF_TIME_VALIDITY -35

typedef int  (PASCAL *  ESignerSignProto) (
  char *operation,
  char *method,
  char *inputFile,
  char *outputFile,
  char *timestampOutputFile,
  char *extension,
  char *certificateFile,
  char *kprivFile,
  char *pfxFile,
  char *pkcs11Dll,
  char *pin_passphrase,
  char *directoryTokenCache,
  char *indexCertificateOnToken,
  char *tsaURI,
  char *tsaUsername,
  char *tsaPassword,
  char *tsaPolicy,
  char *tsaCoding,
  char *rootCADir);

typedef int  (PASCAL * ESignerVerifyProto)(
  char *operation,
  char *inputFileName,
  char *responseFileName,
  char *rootCADir,
  char *rootTSADir,
  char *resultDescription);

////////////////////////////////////////////////////////////////////////////
#endif
////////////////////////////////////////////////////////////////////////////
```

5 ESEMPI DI UTILIZZO

5.1 Caricare la dll

```
HINSTANCE hDLL = LoadLibrary("C:\\TestSignerDll\\ESIGNER.DLL");

if (hDLL != NULL){
   ESignerSignProto   lpfnSign   = (ESignerSignProto   )::GetProcAddress(hDLL, "Sign");
   ESignerVerifyProto lpfnVerify = (ESignerVerifyProto )::GetProcAddress(hDLL, "Verify");
   if (lpfnSign != NULL){
     // Uso delle funzioni
   }
   FreeLibrary(hDLL);
}
```

5.2 Firmare un file singolo

```
      int result = lpfnSign(
        "S", // Sign file
        "P", // File PFX
        "C:\\TestSignerDll\\File.pdf",     // file in input
        "C:\\TestSignerDll\\Signed\\File", // file in output
        NULL,
        ".p7m",
        NULL,
        NULL,
        "C:\\TestSignerDll\\Giuseppe_Verdi.pfx",
        NULL,
        "12345678",
        NULL,
        "0",
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL);
```

5.3 Firmare una directory

```
  result =  lpfnSign ("S+D", "P",
                        "c:\\TestSignerDll\\Files",
                        "c:\\TestSignerDll\\Signed",
                        NULL,
                        ".p7m",
                        NULL, NULL, " C:\\TestSignerDll\\Giuseppe_Verdi.pfx", NULL, "1234",
                        NULL, “0”, NULL, NULL, NULL, NULL, NULL, NULL);
////////////////////////////////////////////////////////////////////////////////////////
```

5.4 Verificare un file singolo

```
char resultString[512];
memset(resultString, 0, sizeof(resultString));
result = lpfnVerify ("V",  "C:\\TestSignerDll\\Signed\\File.pdf.p7m",
                NULL,
                "c:\\TestSignerDll\\RootCA",
                NULL,
                resultString);
```

5.5 Verificare una directory

```
result = lpfnVerify ("V+D",  "C:\\TestSignerDll\\Signed",
                "C:\\TestSignerDll\\Verified",
                "c:\\TestSignerDll\\RootCA",
                "c:\\TestSignerDll\\RootTSA",
                NULL);
```