API Reference Manual

QDigitSign

Index

1 – Target

This document contains the reference manual related to the QDigitSign library.

2 – Definitions and acronyms

3 – History

1.1	Sdk guide for improvement on verification before the signature (QdigitSign.dll [1.0.2.3], QDigitSignNet.dll [1.0.1.1])
1.2	Added the certificate selection and the language setting for PDF Signature. (QdigitSign.dll [1.0.3.2], QDigitSignNet.dll [1.0.2.1])

4 – API Descriptions

QDigitSign library supplies a set of functions (API) that allows to sign files and documents and to verify digital signatures.

4.1 Pre-requisite

- Visual C 2012 runtime (VCRedist) is needed to use the library. Download it from:

  http://support.microsoft.com/kb/2019667

- In case of using the library as a COM interface, it is needed to register **`QDigitSignNet.dll`** using the **regasm** command on the 32-bit .NET version:
  `<<PATH>>\Microsoft.NET\Framework\v2.0.50727>regasm /codebase <<PATH>>\QDigitSignNet.dll`

4.2 General functions

```
long qdigitsign_set(int option, void* value);
```

Sets the general options of the library.
The options are defined in the file DigitSign.h:

QDIGITSIGN_OPT_LOG_LEVEL
Set the log level. Allowed values are:

```
LOG_TYPE_ERROR
LOG_TYPE_WARNING
LOG_TYPE_MESSAGE
LOG_TYPE_DEBUG
```

The default value is LOG_TYPE_ERROR
Ex.

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_LOG_LEVEL, LOG_TYPE_MESSAGE);
```

QDIGITSIGN_OPT_LOG_FILE
Sets the log file
Ex.

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_LOG_FILE, "log.txt");
```

The default value is no log file

QDIGITSIGN_OPT_CACERT_DIR
Sets the folder that contains the trusted CA certificates. The certificates are searched recursively.
Ex.

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_CACERT_DIR, "/usr/local/cacert");
```

The default value is no CA folder

QDIGITSIGN_OPT_PROXY
Sets the proxy.
Ex.

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_PROXY, "http://proxyserver.com");
```

or:

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_PROXY, "http://proxyserver.com:8080");
```

Default is: no proxy

QDIGITSIGN_OPT_PROXY_PORT
Set the proxy port
Ex.

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_PROXY_PORT, "8080");
```

Default is: no proxy

QDIGITSIGN_OPT_TCP_TIMEOUT
Sets the timeout.
Ex.

```
nRet = qdigitsign_set(QDIGITSIGN_OPT_TCP_TIMEOUT, 3000);
```

QDIGITSIGN_OPT_OID_MAP_FILE
Sets the file for OID descriptions to be used to map OID to text descriptions.

Ex. the OID “2.5.4.3” means “CN”

```
Ex.
nRet = qdigitsign_set(QDIGITSIGN_OPT_OID_MAP_FILE, “OID.txt”);
```

Il valore di default è OID.txt

```
QDIGITSIGN_OPT_CONFIG_FILE
```

Sets the configuration file containing the options. Using this file one can set once all options.
The format of this config file is the same as the Java properties file:

```
options=value
```

Ex.
cacertdir=C:\Progetti\Cryptware\Iamboo\elencopubblico\elencopubblico
loglevel=4
logfile=log.log

```
Ex.
nRet = qdigitsign_set(QDIGITSIGN_OPT_CONFIG_FILE, “conf.conf”);
```

Available option:

General Option	
proxy	ex. proxy=http://proxyserver.com or with post specification proxy=http://proxyserver.com:8080)
proxyport	ex. proxyport=8080
proxyusrpass	proxy credential in the format: user:password
oidmap	OID mapping file (ex. oidmap=OID.txt). default OID.txt
logfile	ex. logfile=log.txt
loglevel	ex. loglevel=3. Allowed value 1-5. default 1
cacertdir	Trtusted CA certificates (ex. cacertdir=/usr/local/cacert)
timeout	Connection timeout in milliseconds (ex. timeout=4000)

Digital Signature Options	
p11	pkcs#11 module (ex. p11=cryptoki.dll)
slot	PKCS#1111 slot to be used (ex. slot=0 first slot). default 0
pin	pin token (ex. Pin=12345678)
alias	Certificate alias (ex. alias=CNS0)

cades	CAdES (ex. cades=1)
xades	XAdES (ex. xades=1)
detached	detached signature (ex. detached=1)
inputfile	input file (ex. inputfile=documento.doc)
outputfile	output file (ex. outputfile=documento.doc.p7m). default value: <inputfile>.p7m
inputfiletype	Input file type. Available values: p7m, pdf, xml or auto (default) (ex. inputfiletype=auto)
tsaurl	TSA url
tsauser	TSA username
tsapass	TSA password
revocation	Disable Verify on Revocation (ex. revocation=0), the default is 1
pdfsubfilter	PDF subfilter. Allowed values: detached, cades (es. pdfsubfilter=cades)
pdfname	PDF Name field (ex. pdfname=Paolo_Rossi). Use underscore _ for <space>
pdflocation	PDF Location field. (ex. pdflocation=Milano). Use underscore _ for <space>
pdfreason	PDF Reason field. (ex. pdfname=Signature). Use underscore _ for <space>

Verification Options	
inputfile	input file (ex. inputfile=documento.doc.p7m)
outputfile	XML output file (ex. outputfile=documento.doc.xml). Default value: no xml result file
inputfiletype	Input file type. Available values: p7m, pdf, xml, m7m, tsd, tst, tsr or auto (ex. -infiletype pdf). Default value: auto
revocation	Enable verification (ex. -revocation)
plaintext	Plaintext file for detached signature verification (ex. plaintext=documento.doc)

4.3 Digital Signature Functions

`QDIGITSIGN_CTX qdigitsign_sign_init(void);`

Digital Signature initialization

Ex.

```
ctx = qdigitsign_sign_init();
```

`Certificate[] Sign.GetCertificates();`
On the **Sign** class the method **GetCertificates** fill the Certificate matrix with all the certificate found on the smartcard.

`long qdigitsign_sign_set(QDIGITSIGN_CTX ctx, int option, void* value);`

Sets the digital signature options. Such options are defined in the file DigitSign.h:

`QDIGITSIGN_OPT_PKCS11`
Sets the PKCS#11 module
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_PKCS11, "smaoscki.dll");
```

`QDIGITSIGN_OPT_SLOT`
Sets the slot where the token is present. If not set the library uses the first available slot.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_SLOT, 1);
```

Default value is the first available slot

`QDIGITSIGN_OPT_PIN`
Sets the PIN.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_PIN, "12345678");
```

`QDIGITSIGN_OPT_ALIAS`
Sets the certificate alias.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_ALIAS, "CNS0");

Or the code below to use a specific certificate for the Signature.
nRet = SetStringOption(QDigitSign.QDIGITSIGN_OPT_ALIAS, certs[0].Label);
```

`QDIGITSIGN_OPT_CADES`
Sets the CAdES to generate a CAdES compliant signature.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_CADES, 1);
```

default value is 0 (no CAdES)

`QDIGITSIGN_OPT_XADES`
Set the XAdES to generata a XAdES compliant XML signature.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_XADES, 1);
```

default value is 0 (no XAdES)

`QDIGITSIGN_OPT_DETACHED`
Sets the generation of a detached signature.
Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_DETACHED, 1);
```

default value is 0 (no detached)

QDIGITSIGN_OPT_INPUTFILE
Set the input file to sign
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_INPUTFILE, "document.doc");
```

QDIGITSIGN_OPT_OUTPUTFILE
Set the signed file to generate. If not set the output file is "<inputfile>.p7m" or <inputfile>.pdf if pdf signature.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_ OUTPUTFILE, "docum.doc.p7m");
```

default value is: inputfile.p7m or inputfile.pdf

QDIGITSIGN_OPT_INPUTFILE_TYPE
Set the file type. Supported type are:

QDIGITSIGN_FILETYPE_AUTO
automatic recognition (default value)

QDIGITSIGN_FILETYPE_PLAINTEXT
The input file is treated as plaintext and the signature generates a .p7m

QDIGITSIGN_FILETYPE_P7M
The input file is treated as .p7m file and a second signature is appended to the file (multiple signature)

QDIGITSIGN_FILETYPE_PDF
The input file is treated as .pdf and the signature generates a new signed pdf file

QDIGITSIGN_FILETYPE_XML
The input file is treated as.xml and a XML signed file is generated

default value is: QDIGITSIGN_FILETYPE_AUTO

QDIGITSIGN_OPT_TSA_URL
Sets the TSA url for timestamp.
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_TSA_URL, "http://tsa.net");
```

QDIGITSIGN_OPT_TSA_USERNAME
Sets the TSA username (optional)
Ex.
```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_TSA_USERNAME, "pippo");
```

QDIGITSIGN_OPT_TSA_PASSWORD
Sets the TSA password (optional)
Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_TSA_PASSWORD, "pluto");
```

QDIGITSIGN_OPT_VERIFY_REVOCATION

Forces the CRL verification before signature

Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_VERIFY_REVOCATION, 1);
```

QDIGITSIGN_OPT_PDF_SUBFILTER

Sets the subfilter to be used in pdf signature. Allowed values are:

```
QDIGITSIGN_PDF_SUBFILTER_PKCS_DETACHED -> "adbe.pkcs7.detached"
QDIGITSIGN_PDF_SUBFILTER_ETSI_CADES -> "ETSI.CAdES.detached"
```

Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_PDF_SUBFILTER,
                                QDIGITSIGN_PDF_SUBFILTER_ETSI_CADES);
```

default value is: QDIGITSIGN_PDF_SUBFILTER_PKCS_DETACHED

QDIGITSIGN_OPT_PDF_REASON_LABEL

Set the label to use for the "Signed By" result.

Ex:

```
nRet = SetStringOption(QDIGITSIGN_OPT_PDF_REASON_LABEL, "Reason:");
```

QDIGITSIGN_OPT_PDF_REASON

Sets the value of Reason field in pdf signature.

Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_PDF_REASON, "FirmaDigitale");
```

QDIGITSIGN_OPT_PDF_NAME_LABEL

Set the label to use for the "Signed By" result.

Ex:

```
nRet = SetStringOption(QDIGITSIGN_OPT_PDF_NAME_LABEL, "Signed By:");
```

QDIGITSIGN_OPT_PDF_NAME

Sets the value of Name field in pdf signature.

Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_PDF_NAME, "Paolo Rossi");
```

QDIGITSIGN_OPT_PDF_LOCATION_LABEL

Set the label to use for the "Location" result.

Ex:

```
nRet = SetStringOption(QDIGITSIGN_OPT_PDF_LOCATION_LABEL, "Location:");
```

QDIGITSIGN_OPT_PDF_LOCATION

Sets the value of Location field in pdf signature.

Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_PDF_LOCATION, "Milano");
```

QDIGITSIGN_OPT_CONFIG_FILE

Sets the configuration file for signature operations:

Ex.

```
nRet = qdigitsign_sign_set(QDIGITSIGN_OPT_CONFIG_FILE, "conf.conf");
```

The available options are listed in the table 1

long qdigitsign_sign_sign(QDIGITSIGN_CTX ctx);

Compute the digital signature using the options set

```
Ex.
nRet = qdigitsign_sign_sign(ctx);
```

long qdigitsign_sign_cleanup(QDIGITSIGN_CTX ctx);

Release the allocated memory

```
Ex.
nRet = qdigitsign_sign_cleanup(ctx);
```

4.4 Verification Functions

QDIGITSIGN_CTX qdigitsign_verify_init(void);

Initialize verification engine

```
Ex.
ctx = qdigitsign_verify_init(ctx);
```

long qdigitsign_verify_set(QDIGITSIGN_CTX ctx, int option, void* value);

Sets the verification options

QDIGITSIGN_OPT_INPUTFILE
The file to verify

```
Ex.
nRet = qdigitsign_verify_set(QDIGITSIGN_OPT_INPUTFILE, "doc.p7m");
```

QDIGITSIGN_OPT_OUTPUTFILE
Set the xml output file containing the verification results

```
Ex.
nRet = qdigitsign_verify_set(QDIGITSIGN_OPT_INPUTFILE, "doc.xml");
```

QDIGITSIGN_OPT_INPUTFILE_TYPE
Set the file type. Supportati types are:

```
QDIGITSIGN_FILETYPE_AUTO
QDIGITSIGN_FILETYPE_P7M
QDIGITSIGN_FILETYPE_PDF
QDIGITSIGN_FILETYPE_XML
QDIGITSIGN_FILETYPE_M7M
```

```
QDIGITSIGN_FILETYPE_TSR
QDIGITSIGN_FILETYPE_TST
QDIGITSIGN_FILETYPE_TSD
QDIGITSIGN_FILETYPE_XML

Ex.
nRet = qdigitsign_verify_set(QDIGITSIGN_OPT_INPUTFILE_TYPE,
                             QDIGITSIGN_FILETYPE_P7M);
```

The default value is: `QDIGITSIGN_FILETYPE_AUTO`

`long qdigitsign_verify_verify(QDIGITSIGN_CTX ctx, VERIFY_RESULT* pVerifyResult);`

Computes the verification algorithm using the set options. The structure VERIFY_RESULT will contains the verification info for the digital signatures and timestamps contained in the input file.

The structure VERIFY_RESULT is defined as shown below:

```
typedef struct _REVOCATION_INFO
{
      int nType;  // OCSP, CRL
      char szExpiration[40];
      char szThisUpdate[40];
      int nRevocationStatus;
      char szRevocationDate[40];
} REVOCATION_INFO;


typedef struct _SIGNER_INFO
{
      char szCN[MAX_LEN * 2];
      char szSN[MAX_LEN * 2];
      char szCACN[MAX_LEN * 2];
      char** pszExtensions;
      int nExtensionsCount;
      char szExpiration[MAX_LEN];
      char szValidFrom[MAX_LEN];
      long bitmask;
      char szDigestAlgorithm[MAX_LEN];
      char szSigningTime[MAX_LEN];
      char szCertificateV2[MAX_LEN];
      BOOL b2011Error;
      BYTE* pCertificate;
      int nCertLen;
      void* pTimeStamp;
      REVOCATION_INFO* pRevocationInfo;
      void* pCounterSignatures;
      int nCounterSignatureCount;
} SIGNER_INFO;

typedef struct _TS_INFO
{
```

```
	SIGNER_INFO signerInfo;
	char szTimestamp[MAX_LEN];
	char szTimeStampImprintAlgorithm[MAX_LEN];
	char szTimeStampMessageImprint[MAX_LEN];
	char szTimeStampSerial[MAX_LEN];
} TS_INFO;


typedef struct _SIGNER_INFOS
{
	SIGNER_INFO* pSignerInfo;
	int nCount;
} SIGNER_INFOS;

typedef struct _VERIFY_INFO
{
	SIGNER_INFOS* pSignerInfos;
	TS_INFO* pTSInfo;

} VERIFY_INFO;

typedef struct _VERIFY_RESULT
{
	int nResultType;
	BOOL bVerifyCRL;
	VERIFY_INFO verifyInfo;
	long nErrorCode;
	char szInputFile[MAX_PATH];
	char szPlainTextFile[MAX_PATH];
} VERIFY_RESULT;
```

`long qdigitsign_verify_cleanup_result(VERIFY_RESULT* pVerifyResult);`

Release the allocated memory for verification results

`long qdigitsign_verify_cleanup(QDIGITSIGN_CTX ctx);`

Release the allocated memory

5 – Error Codes

QdigitSign specific Errors

QDIGITSIGN_ERROR_BASE 0x84000000
QDIGITSIGN_ERROR_UNEXPECTED QDIGITSIGN_ERROR_BASE + 1
QDIGITSIGN_ERROR_FILE_NOT_FOUND QDIGITSIGN_ERROR_BASE + 2

QDIGITSIGN_ERROR_DETACHED_PKCS7 QDIGITSIGN_ERROR_BASE + 3

QDIGITSIGN_ERROR_CERT_REVOKED QDIGITSIGN_ERROR_BASE + 4

QDIGITSIGN_ERROR_INVALID_FILE QDIGITSIGN_ERROR_BASE + 5

QDIGITSIGN_ERROR_INVALID_P11 QDIGITSIGN_ERROR_BASE + 6

QDIGITSIGN_ERROR_INVALID_ALIAS QDIGITSIGN_ERROR_BASE + 7

QDIGITSIGN_ERROR_INVALID_SIGOPT QDIGITSIGN_ERROR_BASE + 8

QDIGITSIGN_ERROR_ARRS_BASE QDIGITSIGN_ERROR_BASE + 0x00100000

PKCS#11

See reference on the sites: http://wiki.ncryptoki.com/How-NCryptoki-manages-PKCS-11-errors.ashx

6 – Constant

```
#define QDIGITSIGN_OPT_PKCS11                   1
#define QDIGITSIGN_OPT_SLOT                     2
#define QDIGITSIGN_OPT_PIN                      3
#define QDIGITSIGN_OPT_ALIAS                    4
#define QDIGITSIGN_OPT_CADES                    5
#define QDIGITSIGN_OPT_XADES                    5
#define QDIGITSIGN_OPT_DETACHED                 6
#define QDIGITSIGN_OPT_INPUTFILE                7
#define QDIGITSIGN_OPT_OUTPUTFILE               8
#define QDIGITSIGN_OPT_INPUTFILE_TYPE           9
#define QDIGITSIGN_OPT_TSA_URL                  10
#define QDIGITSIGN_OPT_TSA_USERNAME             11
#define QDIGITSIGN_OPT_TSA_PASSWORD             12
#define QDIGITSIGN_OPT_VERIFY_REVOCATION        13
#define QDIGITSIGN_OPT_LOG_LEVEL                14
#define QDIGITSIGN_OPT_LOG_FILE                 15
#define QDIGITSIGN_OPT_INPUTFILE_PLAINTEXT      16
#define QDIGITSIGN_OPT_CACERT_DIR               17
#define QDIGITSIGN_OPT_PDF_SUBFILTER            18
#define QDIGITSIGN_OPT_CONFIG_FILE              19
#define QDIGITSIGN_OPT_PROXY                    20
#define QDIGITSIGN_OPT_PROXY_PORT               21
#define QDIGITSIGN_OPT_PROXY_USRPASS            22
#define QDIGITSIGN_OPT_OID_MAP_FILE             23
#define QDIGITSIGN_OPT_TCP_TIMEOUT              24
#define QDIGITSIGN_OPT_PDF_REASON               25
```

#define QDIGITSIGN_OPT_PDF_NAME	26
#define QDIGITSIGN_OPT_PDF_LOCATION	27
#define QDIGITSIGN_OPT_PDF_PAGE	28
#define QDIGITSIGN_OPT_PDF_LEFT	29
#define QDIGITSIGN_OPT_PDF_BOTTOM	30
#define QDIGITSIGN_OPT_PDF_WIDTH	31
#define QDIGITSIGN_OPT_PDF_HEIGHT	32
#define QDIGITSIGN_OPT_PDF_IMAGEPATH	33
#define QDIGITSIGN_OPT_PDF_GRAPHOMETRIC_DATA	34
#define QDIGITSIGN_OPT_PDF_GRAPHOMETRIC_DATA_VER	35
#define QDIGITSIGN_OPT_ATR_LIST_FILE	36
#define QDIGITSIGN_OPT_HASH_ALGO	37
#define QDIGITSIGN_OPT_LICENSEE	38
#define QDIGITSIGN_OPT_PRODUCTKEY	39
#define QDIGITSIGN_OPT_RS_OTP_PIN	40
#define QDIGITSIGN_OPT_RS_HSMTYPE	41
#define QDIGITSIGN_OPT_RS_TYPE_OTP_AUTH	42
#define QDIGITSIGN_OPT_RS_USERNAME	43
#define QDIGITSIGN_OPT_RS_PASSWORD	44
#define QDIGITSIGN_OPT_RS_CERTID	45
#define QDIGITSIGN_OPT_RS_SERVICE_URL	46
#define QDIGITSIGN_OPT_PDF_DESCRIPTION	50
#define QDIGITSIGN_OPT_PDF_NAME_LABEL	51
#define QDIGITSIGN_OPT_PDF_REASON_LABEL	52
#define QDIGITSIGN_OPT_PDF_LOCATION_LABEL	53

7 – Examples

7.1 Digital Signature

```
QDIGITSIGN_CTX ctx;
int ret;

ret = qdigitsign_set(QDIGITSIGN_OPT_CONFIG_FILE, "config.conf");
```

```
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_set(QDIGITSIGN_OPT_LOG_FILE, "log.log");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_set(QDIGITSIGN_OPT_LOG_LEVEL, LOG_TYPE_DEBUG);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_set(QDIGITSIGN_OPT_OID_MAP_FILE, "OID.txt");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ctx = qdigitsign_sign_init();

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_PKCS11, "vcki.dll");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_SLOT, 1);
if(ret != 0)
{
      printf("Errore: %x", ret);
}


ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_CADES, 1);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_PIN, "1234");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_ALIAS, "signature");
if(ret != 0)
{
      printf("Errore: %x", ret);
```

```
}

ret = qdigitsign_sign_set(ctx,
                          QDIGITSIGN_OPT_TSA_URL,
                          "https://portal-pte.actalis.it/tsa/FrontEnd");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_TSA_USERNAME, "xxxxx");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_TSA_PASSWORD, "yyyyy");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_INPUTFILE, "test.txt");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx,
                          QDIGITSIGN_OPT_INPUTFILE_TYPE,
                          QDIGITSIGN_FILETYPE_AUTO);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_set(ctx, QDIGITSIGN_OPT_VERIFY_REVOCATION, 0);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_sign(ctx);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_sign_cleanup(ctx);
if(ret != 0)
{
      printf("Errore: %x", ret);
}
```

```
qdigitsign_cleanup();
```

7.2 Digital Signature Verification

```
QDIGITSIGN_CTX ctx;
VERIFY_RESULT verifyResult;
int ret

ret = qdigitsign_set(QDIGITSIGN_OPT_CONFIG_FILE, "config.conf");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_set(QDIGITSIGN_OPT_LOG_FILE, "log.log");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_set(QDIGITSIGN_OPT_LOG_LEVEL, LOG_TYPE_DEBUG);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_set(QDIGITSIGN_OPT_OID_MAP_FILE, "OID.txt");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ctx = qdigitsign_verify_init();

ret = qdigitsign_verify_set(ctx, QDIGITSIGN_OPT_INPUTFILE, "test.txt.p7m");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_verify_set(ctx, QDIGITSIGN_OPT_OUTPUTFILE, "test.txt.xml");
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_verify_set(ctx,
                            QDIGITSIGN_OPT_INPUTFILE_TYPE,
                            QDIGITSIGN_FILETYPE_P7M);
if(ret != 0)
{
      printf("Errore: %x", ret);
}
```

```
ret = qdigitsign_verify_set(ctx, QDIGITSIGN_OPT_VERIFY_REVOCATION, 1);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_verify_verify(ctx, &verifyResult);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_verify_cleanup_result(&verifyResult);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

ret = qdigitsign_verify_cleanup(ctx);
if(ret != 0)
{
      printf("Errore: %x", ret);
}

qdigitsign_cleanup();
```