CryptLib for Mainframe

Security Toolkit

API Handbook

Version 2.4.2

CryptLib for Mainframe Programmer's handbook

Copyright

Trademarks

Table of contents

Chapter

1

Introduction

Cryptography is the science that deals with the coverage of messages.

The development of new approaches and methods to secure messages has been driven in the past primarily by military requirements. This can be traced back to the Roman Empire. The roman military used simple cipher methods in order to hide plaintext from their enemies. The so called 'Caesar-Cipher' may exemplify this fact.

In the years past, cryptographic applications have made their way into many areas of the real world. One of the driving reasons for this is the fact that the fast expansion of the World Wide Web has enabled the building of large computer networks which in turn has lead to increased communication possibilities.

These require a more in depth discussion regarding the security of the transmitted data. Herein various interests play a role. These may be of personal character such as in the area of online banking or of business character such as in the area of clearing business transactions over the Internet.

The increased demand for cryptographic procedures has pressed ahead the development of audited, secure, easy to use and publicly available algorithms.

Today it's possible for any computer user to protect his personal data using methods regarded as secure. In this context, secure means that even the application of computer power currently estimated as unrealistically large doesn't provide the opportunity to systematically ascribe ciphertext into plaintext investing a reasonable amount of time. This can only be carried out if a secure token of information, called the 'key' is well known.

In the course of time some procedures have established as standards. That's because of the fact that these procedures are well documented and thus are well tested regarding security and stability. It's important to note that the security of these procedures relies only on the used key. Public knowledge of the internals of the used algorithm doesn't weaken the security of the chosen procedures.

CryptLib from XPS Software GmbH offers to the programmer a library containing standardized cryptographic procedures for use in proprietary development. CryptLib is available for the following operating systems: Win32, Linux, OS/2, OS/400, IBM iSeries, VSE/ESA, MVS/ESA, OS/390 and IBM zSeries.

CryptLib offers among others methods to calculate hash values, methods for symmetrical and asymmetrical encryption, methods to process X.509 certificates, methods for creation and check of digital signatures and support of the Public Key Cryptography Standards PKCS#7 (S/MIME) and PKCS#12 (public key).

Chapter

2

Installation

System requirements

Operating system requirements

CryptLib V1.5 will run under MVS/ESA from version 5.0 onwards, under OS/390 from version 1.3 onwards and under z/OS from version 1.1 onwards.

CryptLib will also run under VSE/ESA version 2.3 onwards.

Hardware requirements

CryptLib is designed to be used on an IBM System/390 or compatible processor supporting one or more of the operating systems previously specified. Since the CryptLib libraries can be delivered on tape or CD-ROM, to install CryptLib you will require either a tape or cartridge drive, or a client workstation with CD-ROM drive and an FTP connection to the host.

Installation steps

§ Unload the installation media (tape or CD-ROM)

Installation under MVS, OS/390 and z/OS

Unloading the CryptLib installation tape

The following libraries should be prepared for the installation files:

Name	Space	Lrecl	Blksz	Recfm
XPSCRYPT.V150.LOADLIB	6144,(100,50,50)		6144	U
XPSCRYPT.V150.MACLIB	3200,(25,5,10)	80	3200	FB

XPSCRYPT.V150.LOADLIB contains the executable program code. XPSCRYPT.V150.MACLIB contains the copy files, the sample programs, the license file as well as some sample certificates.

Example job:

```
//XPSC150   JOB ,'INSTALL CRYPTLIB',
//          CLASS=c,MSGCLASS=x
//TAPL      EXEC PGM=IEBCOPY
//LOADIN    DD DISP=(OLD,PASS),VOL=(,RETAIN,SER=XPSC15),
//             LABEL=(1,SL),DSN=XPSC150.LOADLIB,
//             UNIT=cart
//LOADOUT   DD DISP=(NEW,CATLG),DSN=xpscrypt.V150.loadlib,
//             SPACE=(6144,(100,50,50),,,ROUND),DCB=SYS1.LINKLIB,
//             VOL=SER=mvs001,UNIT=dasd
//MACIN     DD DISP=(OLD,PASS),VOL=(,RETAIN,SER=XPSC15),
//             LABEL=(2,SL),DSN=XPSC150.MACLIB,
//             UNIT=cart
//MACOUT    DD DISP=(NEW,CATLG),DSN=xpscrypt.V150.maclib,
//             SPACE=(3200,(25,5,10),,,ROUND),DCB=SYS1.MACLIB,
//             VOL=SER=mvs001,UNIT=dasd
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 COPY  INDD=LOADIN,OUTDD=LOADOUT
 COPY  INDD=MACIN,OUTDD=MACOUT
```

Abb. 1: Installation job - MVS

Unloading the CryptLib installation CD-ROM

The installation libraries from the distribution CD-ROM must first be transferred to the host on which CryptLib will be installed using some form of FTP program. The libraries are provided in TSO transmit (XMIT) format and should be transferred to the host as binary files. Before transferring the files you should allocate the space to receive them as follows:

Name	Space	Lrecl	Blksz	Recfm
XMIT.XPSCRYPT.V150.LOADLIB	**600,(100)**	**80**	**3200**	**FB**
XMIT.XPSCRYPT.V150.MACLIB	**200,(20)**	**80**	**3200**	**FB**

Then you can transfer the following files from the subfolder **"\MVS"** on the CD-ROM to the host. When transferring the files you should rename them as follows:

Clientname	Hostname
XPSC150L.BIN	XMIT.XPSCRYPT.V150.LOADLIB
XPSC150M.BIN	XMIT.XPSCRYPT.V150.MACLIB

Next, receive the libraries from the files with the following commands:

§ For the loadlib:

```
RECEIVE INDSN(XMIT.XPSCRYPT.V150.LOADLIB)
```

After entering the command you will receive the following prompt:

```
INMR901I Dataset XPSCRYPT.V150.LOADLIB from XPSSYST on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

Enter the file name as follows: (you may need additional parameters depending on your installations standards)

```
DSN(xpscrypt.V150.loadlib)
```

§ For the maclib:

```
RECEIVE INDSN(XMIT.XPSCRYPT.V150.MACLIB)
```

After entering the command you will receive the following prompt:

```
INMR901I Dataset XPSCRYPT.V150.MACLIB from XPSSYST on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

Enter the file name as follows: (you may need additional parameters depending on your installations standards)

```
DSN(xpscrypt.V150.maclib)
```

Using CryptLib

Because of the fact that CryptLib is completely programmed in 370-Assembler its use requires no specific system dependencies. This means that CryptLib can be used in batch programming as well as in programs intended to run under CICS or IMS. CryptLib may be called from any available mainframe programming languages such as COBOL, PL/1, RPG or Assembler.

In order to make CryptLib available at runtime the following job control statements have to be added to the executer job:

Example job:

```
//STEPLIB  DD DISP=SHR,DSN=xpscrypt.v150.loadlib
//XPSDATA  DD DISP=SHR,DSN=xpscrypt.v150.maclib
```

Abb. 2: Job control - MVS

Installation under VSE/ESA and z/VSE

Unloading the CryptLib installation type

Example job:

```
// JOB XPSCRYPT    INSTALL XPS-CRYPT150
// ASSGN SYS006,tape
// EXEC LIBR
 RESTORE SUB=XPS.CRYPTLIB:lib.sublib -
 TAPE=SYS006 LIST=YES REPLACE=YES
/*
/&
```

Abb. 3: Installation job - VSE

Using CryptLib

Because of the fact that CryptLib is completely programmed in 370-Assembler its use requires no specific system dependencies. This means that CryptLib can be used in batch programming as well as in programs intended to run under CICS. CryptLib may be called from any available mainframe programming languages such as COBOL, PL/1, RPG or Assembler.

Example programs in Assembler and COBOL have been stored in the installation library with an extension of "Z".

In order to make CryptLib available at runtime the following job control statement has to be added to the executer job:

Example job:

```
// LIBDEF PHASE,SEARCH=(xps.cryptlib)
```

Abb. 4: Job control - VSE

Chapter

3

Key generation

Common information

Random numbers play an integral part in cryptography. Generating a new symmetrical key as well as a new asymmetrical key begins with the generation of a random number. The integrity of the chosen random number is very important. If it is predictable the generated key can be recalculated given the chosen procedure is known. The security of the whole system depends on the privacy of the key. Therefore it's possible to provide a specific initialization value using the *seed* parameter which will be incorporated into the process of key generation.

A RSA public/private key pair can be generated using the *GENERATE-RSA-KEY* method. The method *GENERATE-KEY* can be used to generate random keys, initialization vectors (iv) or any other random values.

Methods

GENERATE-KEY

Generate a random key for symmetrical encryption and decryption.

Syntax	**Cobol**	
	MOVE GENERATE-KEY TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **KEY, KEYLEN, SEED, SEEDLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GENERATE_KEY,KEY,KEYLEN,SEED,SEEDLEN,RC),VL	
Return code	None.	
Parameter	**Description**	**Use**
KEY	Address of the storage area to be used to return the generated symmetrical key.	Output
KEYLEN	Length of the key to generate.	Input
SEED	Address of storage area holding variable data to be incorporated into the key generation.	Input
SEEDLEN	Length of the variable data.	Input

COBOL example:

```
*-------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE GENERATE KEY                       *
*-------------------------------------------------------------*
```

```
 ID DIVISION.
 PROGRAM-ID.
     SAMPLE.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  SEED            PIC X(32)        VALUE X"0F0E0D0C0B0A09080706
-    "0504030201000F0E0D0C0B0A090807060504030201000".
 01  KEY             PIC X(32).
*
 77  CRYPT-FUNCTION  PIC X.
 77  KEYLEN          PIC 9(8)   COMP  VALUE 32.
 77  SEEDLEN         PIC 9(8)   COMP  VALUE 32.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**            PROCEDURE DIVISION                               **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* GENERATE RANDOM AES-KEY                                      *
* ------------------------------------------------------------ *
     MOVE GENERATE-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, KEY, KEYLEN, SEED, SEEDLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
     STOP RUN.
 ENDRUN.
```

GENERATE-RSA-KEY

Generate a random RSA public/private key pair.

Syntax	**Cobol**	
	MOVE GENERATE-RSA-KEY TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **PRIVKEY, PUBKEY, SEED, SEEDLEN, KEYLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GENERATE_RSA_KEY,PRIVKEY,PUBKEY,SEED,SEEDLEN,KEYLEN,RC),VL	
Return code	None.	
Parameter	**Description**	**Use**
PRIVKEY	Address of the storage area to be used to return the generated private RSA key.	Output
PUBKEY	Address of the storage area to be used to return the generated public RSA key.	Output
SEED	Address of storage area holding variable data to be incorporated into the key generation.	Input
SEEDLEN	Length of the variable data.	Input
KEYLEN	Desired key length in bits (maximum 4096).	Input

COBOL example:

```
*-------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE GENERATE RSA-KEY                   *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     SAMPLE.
*
```

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  SEED            PIC X(32)        VALUE X"0F0E0D0C0B0A09080706
-    "0504030201000F0E0D0C0B0A090807060504030201000".
*
 77  CRYPT-FUNCTION  PIC X.
 77  KEYLEN          PIC 9(8)   COMP  VALUE 1024.
 77  SEEDLEN         PIC 9(8)   COMP  VALUE 32.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
*
 COPY XPSCLRSA.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* GENERATE RANDOM RSA-KEY                                      *
* ------------------------------------------------------------ *
     MOVE GENERATE-RSA-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, RSA-PRIVATE-KEY,
         RSA-PUBLIC-KEY, SEED, SEEDLEN, KEYLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
     STOP RUN.
 ENDRUN.                                                      }
```

Chapter

4

Encryption

Common information

A distinction is drawn between two basic encryption modes:

Symmetrical

When using a symmetrical encryption method, message encryption and decryption is carried out using the identical key. This demands the sender (encryption) and the receiver (decryption) of the message to know and use the same key. If CBC is chosen as mode of operation an additional initialization vector is needed. The length of this initialization vector equals the block length of the chosen algorithm (DES, TripleDES, RC2, RC4, Blowfish = 8 Byte, AES = 16 Byte).

Asymmetrical (public key)

Methods using non identical keys for encryption and decryption are called asymmetrical. Both keys are created during the process of key generation and it is impossible to suggest the one key from the other. This enables one key to be made publicly available why it is called the public key. Using the public key the sender of a message can encrypt the message and send it to the owner of the public/private key pair who in turn is able to decrypt the public key encrypted message using the private key only known to him.

CryptLib supports a number of symmetrical encryption algorithms (AES, DES, TripleDES, RC2, RC4, and Blowfish) as well as the public/private key encryption algorithm RSA. Using the method *INIT-CTX* the programmer can chose the desired mode for encryption and decryption. After context initialization is complete the methods *ENCRYPT* and *DECRYPT* can be used to encrypt and decrypt messages regardless of the chosen encryption algorithm.

Methods

INIT-CTX

Initialize the cryptographic context.

Syntax	Cobol

	`MOVE INIT-CTX TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, KEY, RC.`

	Assembler
	`CALL XPSCRYPT,(INIT_CTX,CTX,KEY,RC),VL`

Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Address of storage area holding the context needed for encryption and decryption. The description of the required context structure can be found in the copy books XPSCLCTX (COBOL) and XPSCLASM (Assembler) respectively.	Input
Fields	**Description**	
CX-ALGO	Algorithm used to encrypt/decrypt the data. CryptLib supports the following algorithms: AES — Advanced Encryption Standard (Rijndael) DES — Data Encryption Standard with ctx.keylength = 56 TripleDES EDE2 — Data Encryption Standard with ctx.keylength = 112 TripleDES EDE3 Data — Encryption Standard with ctx.keylength = 168 RC2 — Rivest Cipher No. 2 RC4 — Rivest Cipher No. 4 Blowfish — Schneier RSA — Public-/Private-Key method from Rivest, Shamir and Adleman	
CX-MODE	Mode of operation. CryptLib supports the following modes for symmetrical encryption (AES, DES, RC2, RC4 and Blowfish): ECB — Electronic-Codebook-Mode CBC — Cipher-Block-Chaining CryptLib supports the following modes for asymmetrical encryption (RSA): PUBLIC — (public key encryption/decryption) *ctx.key* must specify the storage address of a RSA_PUBLIC_KEY structure PRIVATE — (private key encryption(decryption) *ctx.key* must specify the storage address of a RSA_PRIVATE_KEY structure	
CX-KLEN	Key length. CryptLib supports the following key lengths: AES — 128, 192, 256 DES — 56, 112, 168 RC2 — 40, 64, 128 RC4 — 40, 64, 128 Blowfish — 128 RSA — 512, 1024, 2048, 4096	
CX-IV	Address of storage area holding the initialization vector to be used for symmetrical encryption.	
Parameter	**Description**	**Use**
KEY	Storage address of the key.	Input

ENCRYPT

Encrypt data. The type of encryption is dependent on the *CX-ALGO* parameter chosen for context initialization using the *INIT-CTX* method.

Syntax	Cobol
	MOVE ENCRYPT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, INPUT, INPUTLEN, OUTPUT, OUTPUTLEN, RC.**
	Assembler
	CALL XPSCRYPT,(ENCRYPT,CTX,INPUT,INPUTLEN,OUTPUT,OUTPUTLEN,RC),VL

Return code	Length of encrypted data or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context to be used for encryption.	Input
INPUT	Storage address of the data to encrypt.	Input
INPUTLEN	Length of the data to encrypt.	Input
OUTPUT	Address of storage area to be used to hold the encrypted data.	Output
OUTPUTLEN	Length of the storage area specified using the *OUTPUT* parameter.	Output

DECRYPT

Decrypt data. The type of decryption is dependent on the *CX-ALGO* parameter chosen for context initialization using the *INIT-CTX* method.

Syntax	**Cobol**	
	MOVE DECRYPT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, INPUT, INPUTLEN, OUTPUT, OUTPUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(DECRYPT,CRX,INPUT,INPUTLEN,OUTPUT,OUTPUTLEN,RC),VL	
Return code	Length of decrypted data or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context to be used for decryption.	Input
INPUT	Storage address of the data to decrypt.	Input
INPUTLEN	Length of the data to decrypt.	Input
OUTPUT	Address of storage area to be used to hold the decrypted data.	Output
OUTPUTLEN	Length of the storage area specified using the *OUTPUT* parameter.	Output

GET-RESULT-LENGTH

Determine the length of the storage needed to store the encryption/decryption result.
Hint: Using RSA this method can only be used to determine the resulting storage size for encryption. In the case of decryption the resulting storage size is always lower or equal to the input size.

Syntax	**Cobol**	
	MOVE GET-RESULT-LENGTH TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, INPUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_RESULT_LENGTH,CTX,INPUTLEN,RC),VL	
Return code	Length of encrypted/decrypted data or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context to be used for encryption/decryption.	Input
INPUTLEN	Length of the data to encrypt/decrypt.	Input

RESET-CTX

Reset the initialization vector (iv) for symmetrical encryption and decryption.

Syntax	Cobol	
	MOVE RESET-CTX TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(RESET_CTX,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	The storage address of the context.	Input

CLEANUP-CTX

Deallocation of the storage needed for encryption and decryption.

Syntax	Cobol	
	MOVE CLEANUP-CTX TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(CLEANUP_CTX,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	The storage address of the context.	Input

COBOL example (AES):

```
*-----------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM AES-ENCRYPTION                 *
*-----------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     AESTSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  BLOCK1          PIC X(32)        VALUE "XPS Software GmbH, Ha
-                                          "ar/Muenchen".
 01  OUT             PIC X(32)        VALUE SPACES.
 01  OUT2            PIC X(32)        VALUE SPACES.
 01  SEED1           PIC X(32)        VALUE X"0F0E0D0C0B0A09080706
-    "0504030201000F0E0D0C0B0A09080706050403020100".
 01  SEED2           PIC X(32)        VALUE X"0001020304050607080 9
-                                           "0A0B0C0D0E0F".
 01  AESKEY          PIC X(32).
 01  AESIV           PIC X(16).
 COPY XPSCLCTX.
*
 77  CRYPT-FUNCTION  PIC X.
 77  KEYLEN          PIC 9(8)   COMP  VALUE 32.
 77  IVLEN           PIC 9(8)   COMP  VALUE 16.
 77  BLOCKLEN        PIC 9(8)   COMP  VALUE 32.
 77  OUTLEN          PIC 9(8)   COMP  VALUE 32.
 77  OUTLEN2         PIC 9(8)   COMP  VALUE 32.
 77  SEEDLEN         PIC 9(8)   COMP  VALUE 16.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
```

```
**            PROCEDURE DIVISION                               **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* GENERATE RANDOM AES-KEY                                      *
* ------------------------------------------------------------ *
     MOVE GENERATE-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, AESKEY, KEYLEN, SEED1, SEEDLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* GENERATE RANDOM IV (INITIALIZATION VECTOR FOR CIPHER-CBC)    *
* ------------------------------------------------------------ *
     MOVE GENERATE-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, AESIV, IVLEN, SEED2, SEEDLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* FILL CIPHER-CONTEXT AND INIT CONTEXT                         *
* ------------------------------------------------------------ *
     MOVE AES TO CX-ALGO.
     MOVE CBC TO CX-MODE.
     MOVE 256 TO CX-KLEN.
     MOVE AESIV TO CX-IV.
     MOVE INIT-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, AESKEY, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* ENCRYPT DATA                                                 *
* ------------------------------------------------------------ *
     MOVE ENCRYPT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, BLOCK1, BLOCKLEN,
               OUT, OUTLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* RESET CONTEXT (CHANGE FROM ENCRYPT TO DECRYPT)               *
* ------------------------------------------------------------ *
     MOVE RESET-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* DECRYPT DATA                                                 *
* ------------------------------------------------------------ *
     MOVE DECRYPT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, OUT, OUTLEN,
               OUT2, OUTLEN2, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* CLEANUP CONTEXT                                              *
* ------------------------------------------------------------ *
     MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RC.
     STOP RUN.
 ENDRUN.
```

COBOL example (TripleDES):

```
*-------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM DES-ENCRYPTION             *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     DESTSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  BLOCK1          PIC X(32)        VALUE "XPS Software GmbH, Ha
-                                          "ar/Muenchen".
```

```
 01  OUT             PIC X(32)        VALUE SPACES.
 01  OUT2            PIC X(32)        VALUE SPACES.
 01  DESKEY          PIC X(24)        VALUE X"0F0E0D0C0B0A09080706
-    "0504030201000F0E0D0C0B0A0908".
 01  DESIV           PIC X(8)         VALUE X"0001020304050607".
 COPY XPSCLCTX.
*
 77  CRYPT-FUNCTION  PIC X.
 77  KEYLEN          PIC 9(8)   COMP  VALUE 24.
 77  IVLEN           PIC 9(8)   COMP  VALUE 8.
 77  BLOCKLEN        PIC 9(8)   COMP  VALUE 32.
 77  OUTLEN          PIC 9(8)   COMP  VALUE 32.
 77  OUTLEN2         PIC 9(8)   COMP  VALUE 32.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**            PROCEDURE DIVISION                               **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* FILL CIPHER-CONTEXT AND INIT CONTEXT                         *
* ------------------------------------------------------------ *
     MOVE DES TO CX-ALGO.
     MOVE CBC TO CX-MODE.
     MOVE 168 TO CX-KLEN.
     MOVE DESIV TO CX-IV.
     MOVE INIT-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, DESKEY, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* ENCRYPT DATA                                                 *
* ------------------------------------------------------------ *
     MOVE ENCRYPT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, BLOCK1, BLOCKLEN,
               OUT, OUTLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* RESET CONTEXT (CHANGE FROM ENCRYPT TO DECRYPT)               *
* ------------------------------------------------------------ *
     MOVE RESET-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* DECRYPT DATA                                                 *
* ------------------------------------------------------------ *
     MOVE DECRYPT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, OUT, OUTLEN,
               OUT2, OUTLEN2, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------ *
* CLEANUP CONTEXT                                              *
* ------------------------------------------------------------ *
     MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RC.
     STOP RUN.
 ENDRUN.
```

COBOL example (RSA):

```
*-------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM RSA-ENCRYPTION             *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     RSATSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  BLOCK1          PIC X(32)        VALUE "XPS Software GmbH, Ha
-                                          "ar/Muenchen".
```

```
 01  OUT             PIC X(128)       VALUE SPACES.
 01  OUT2            PIC X(128)       VALUE SPACES.
 01  SEED            PIC X(32)        VALUE X"0F0E0D0C0B0A09080706
-    "0504030201000F0E0D0C0B0A09080706050403020100".
 COPY XPSCLCTX.
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  KEYLEN          PIC 9(8)   COMP  VALUE 512.
 77  BLOCKLEN        PIC 9(8)   COMP  VALUE 32.
 77  OUTLEN          PIC 9(8)   COMP  VALUE 128.
 77  OUTLEN2         PIC 9(8)   COMP  VALUE 128.
 77  SEEDLEN         PIC 9(8)   COMP  VALUE 32.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**            PROCEDURE DIVISION                               **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------- *
* GENERATE RANDOM RSA-KEY                                       *
* ------------------------------------------------------------- *
     MOVE GENERATE-RSA-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, RSA-PRIVATE-KEY,
         RSA-PUBLIC-KEY, SEED, SEEDLEN, KEYLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------- *
* FILL CIPHER-CONTEXT AND INIT CONTEXT                          *
* ------------------------------------------------------------- *
     MOVE RSA    TO CX-ALGO.
     MOVE PUBLIC TO CX-MODE.
     MOVE KEYLEN TO CX-KLEN.
     MOVE INIT-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RSA-PUBLIC-KEY, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------- *
* ENCRYPT DATA                                                  *
* ------------------------------------------------------------- *
     MOVE ENCRYPT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, BLOCK1, BLOCKLEN,
               OUT, OUTLEN, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------- *
* ENCRYPT DATA                                                  *
* ------------------------------------------------------------- *
     MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
     MOVE RSA     TO CX-ALGO.
     MOVE PRIVATE TO CX-MODE.
     MOVE KEYLEN  TO CX-KLEN.
     MOVE INIT-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RSA-PRIVATE-KEY, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------- *
* RESET CONTEXT (CLEANUP OLD CONTEXT AND INIT NEW CONTEXT)      *
* (CHANGE FROM PUBLIC-KEY TO PRIVATE-KEY)                       *
* ------------------------------------------------------------- *
     MOVE DECRYPT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, OUT, OUTLEN,
               OUT2, OUTLEN2, RC.
     IF RC < 0
        DISPLAY RC
        GOBACK.
* ------------------------------------------------------------- *
* CLEANUP CONTEXT                                               *
* ------------------------------------------------------------- *
     MOVE CLEANUP-CTX TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, CIPHER-CTX, RC.
     STOP RUN.
 ENDRUN.
```

Chapter

5

Digital signature

Common information

The implementation of digital signatures is one of the major applications for asymmetrical encryption. To create a digital signature at first a hash value for the data about to be digitally signed is created. Then, in a second step this hash value will be encrypted using the private key. Later on the result can be decrypted using the associated public key. Based on re-generation of the hash value and comparison of the result with the decrypted hash value, the integrity of the originally digitally signed data can be guaranteed.

CryptLib supports digital signatures created with the **RSA** algorithm. The supported methods for hash value generation are **MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512** and **RipeMD160**.

Use the following steps to create a digital signature:

§ Initialize the context using the *SIGN-INIT* method to specify the desired hash method.

§ Add data to be digitally signed using the *SIGN-UPDATE* method. This method can be called as often as needed.

§ Call the *SIGN-FINAL* method to finalize the procedure and to get hold of the digital signature.

Use the following steps to verify a digital signature:

§ Initialize the context using the *VERIFY-INIT* method to specify the desired hash method.

§ Add data to be verified using the *VERIFY-UPDATE* method. This method can be called as often as needed.

§ Call the *VERIFY-FINAL* method to finalize the procedure and to verify the digital signature.

Methods

SIGN-INIT

Initialize the signature context.

Syntax	Cobol

	`MOVE SIGN-INIT TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`HASHALGO, CTX, RC.`

	Assembler	
	CALL XPSCRYPT,(SIGN_INIT,HASHALGO,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
HASHALGO	The type of hash to be used. CryptLib supports MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160.	Input
CTX	Address pointer to receive the storage address of the created context. This will be used for subsequent processing.	Output

SIGN-UPDATE

Add data to be signed.

Syntax	**Cobol**	
	MOVE SIGN-UPDATE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, DATALEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(SIGN_UPDATE,CTX,DATA,DATALEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context initialized for creating the signature.	Input
DATA	Storage address of the data to be signed.	Input
DATALEN	Length of the data to be signed.	Input

SIGN-FINAL

Create the digital signature using the private key.

Syntax	**Cobol**	
	MOVE SIGN-FINAL TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, SIGN, SIGNLEN, PRIVKEY, RC.**	
	Assembler	
	CALL XPSCRYPT,(SIGN_FINAL,CTX,SIGN,SIGNLEN,PRIVKEY,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context to be used to sign the data.	Input
SIGN	Address of storage area about to hold the created digital signature.	Output
SIGNLEN	Address of storage area about to hold the length of the created digital signature.	Output
PRIVKEY	The signer's private RSA key.	Input

VERIFY-INIT

Initialize the context for verification.

Syntax	Cobol	
	MOVE VERIFY-INIT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **HASHALGO, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(VERIFY-INIT,HASHALGO,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
HASHALGO	The type of hash to be used. CryptLib supports MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160.	Input
CTX	Address pointer to receive the storage address of the created context. This will be used for subsequent processing.	Output

VERIFY-UPDATE

Add data to be verified.

Syntax	Cobol	
	MOVE VERIFY-UPDATE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, DATALEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(VERIFY_UPDATE,CTX,DATA,DATALEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context initialized for verification.	Input
DATA	Storage address of data to be verified.	Input
DATALEN	Length of the data to be verified.	Input

VERIFY-FINAL

Verify the digital signature using the public key.

Syntax	Cobol	
	MOVE VERIFY-FINAL TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, SIGN, SIGNLEN, PUBKEY, RC.**	
	Assembler	
	CALL XPSCRYPT,(VERIFY_FINAL,CTX,SIGN,SIGNLEN,PUBKEY,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context initialized for verification.	Input
SIGN	Storage address of the digital signature.	Input
SIGNLEN	Length of the digital signature.	Input

PUBKEY	The signer's public RSA key.	Input

COBOL example:

```
*-------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM: DIGITALE SIGNATURE        *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     SIGNTSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS12-FILE     PIC X(8)       VALUE "XPSUSERP".
 01  PWD             PIC X(8)       VALUE "xpsuser1".
 01  DATA1           PIC X(32)      VALUE "XPS Software GmbH, Haar
-                                          "/Muenchen".
 01  DATA2           PIC X(21)      VALUE "Muenchener Strasse 17".
 01  HASHSHA1        PIC 9(8) COMP  VALUE 26.
 01  HASHMD5         PIC 9(8) COMP  VALUE 5.
 01  SIGNATURE       PIC X(128).
 01  PKCS12-CTX      POINTER.
 01  CERT-CTX        POINTER.
 01  SIGN-CTX        POINTER.
 01  ADDR-PKCS12     POINTER.
 01  ADDR-CERT       POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DATALEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  DATALEN1        PIC 9(8)   COMP  VALUE 32.
 77  DATALEN2        PIC 9(8)   COMP  VALUE 21.
 77  PWDLEN          PIC 9(8)   COMP  VALUE 8.
 77  RC              PIC S9(8)  COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* READ PKCS-12 FILE "XPSUSERP" FROM MACLIB                     *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS12-FILE, ADDR-PKCS12, DATALEN, RC.
     IF RC < 0
        DISPLAY "FILE 'XPSUSERP' NOT FOUND  RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* CONVERT PASSWORD FROM EBCDIC TO ASCII  (PASSWORD LOWER CASE|) *
* ------------------------------------------------------------ *
     MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
* ------------------------------------------------------------ *
* IMPORT PKCS-12 FILE                                          *
* ------------------------------------------------------------ *
     MOVE IMPORT-PKCS12 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS12,
        DATALEN, PWD, PWDLEN, PKCS12-CTX, RC.
     IF RC < 0
        DISPLAY "ERROR IMPORT-FILE: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* GET PRIVATE-KEY FROM PKCS-12 FILE                            *
* ------------------------------------------------------------ *
     MOVE GET-PRIVATE-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS12-CTX, RSA-PRIVATE-KEY, RC.
     IF RC < 0
        DISPLAY "ERROR GET-PRIVATE-KEY: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* GET CERTIFICATE FROM PKCS-12 FILE                            *
* ------------------------------------------------------------ *
     MOVE GET-FIRST-CERT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS12-CTX, ADDR-CERT, RC.
     IF RC < 0
        DISPLAY "ERROR GET-FIRST-CERT: RC = " RC
```

```
       GOBACK.
* ------------------------------------------------------------ *
* IMPORT X.509 CERTIFICATE                                     *
* ------------------------------------------------------------ *
    MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        ADDR-CERT, CERT-CTX, RC.
    IF RC < 0
       DISPLAY "ERROR IMPORT-CERTIFICATE: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* GET PUBLIC-KEY FROM CERTIFICATE                              *
* ------------------------------------------------------------ *
    MOVE RSA-PUBLIC-LEN TO DATALEN
    MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, RSA-PUBLIC-KEY, DATALEN, RC.
    IF RC < 0
       DISPLAY "ERROR GET-PUBLIC-KEY: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* CLEANUP CERTIFICATE CONTEXT                                  *
* ------------------------------------------------------------ *
    MOVE CLEANUP-CERTIFICATE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        CERT-CTX, RC.
    IF RC < 0
       DISPLAY "ERROR CLEANUP-CERTIFICATE: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* CLEANUP PKCS-12 CONTEXT                                      *
* ------------------------------------------------------------ *
    MOVE CLEANUP-PKCS12 TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, PKCS12-CTX, RC.
* ------------------------------------------------------------ *
* RELEASE FILE-STORAGE "XPSTESTP"                              *
* ------------------------------------------------------------ *
    MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        ADDR-PKCS12, RC.
    IF RC < 0
       DISPLAY "ERROR RELEASE-FILE: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* INITIALIZE SIGNATURE-CONTEXT                                 *
* ------------------------------------------------------------ *
    MOVE SIGN-INIT TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HASHSHA1, SIGN-CTX, RC.
    IF RC < 0
       DISPLAY "ERROR SIGN-INIT: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* UPDATE DATA                                                  *
* ------------------------------------------------------------ *
    MOVE SIGN-UPDATE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        SIGN-CTX, DATA1, DATALEN1, RC.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        SIGN-CTX, DATA2, DATALEN2, RC.
* ------------------------------------------------------------ *
* FINALIZE SIGNATURE                                           *
* ------------------------------------------------------------ *
    MOVE SIGN-FINAL TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        SIGN-CTX, SIGNATURE, DATALEN, RSA-PRIVATE-KEY, RC.
    IF RC < 0
       DISPLAY "ERROR SIGN-FINAL: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* INITIALIZE VERIFY-CONTEXT                                    *
* ------------------------------------------------------------ *
    MOVE VERIFY-INIT TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HASHSHA1, SIGN-CTX, RC.
    IF RC < 0
       DISPLAY "ERROR VERIFY-INIT: RC = " RC
       GOBACK.
* ------------------------------------------------------------ *
* UPDATE DATA                                                  *
* ------------------------------------------------------------ *
    MOVE VERIFY-UPDATE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        SIGN-CTX, DATA1, DATALEN1, RC.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        SIGN-CTX, DATA2, DATALEN2, RC.
* ------------------------------------------------------------ *
* FINALIZE SIGNATURE                                           *
* ------------------------------------------------------------ *
```

```
    MOVE VERIFY-FINAL TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        SIGN-CTX, SIGNATURE, DATALEN, RSA-PUBLIC-KEY, RC.
    STOP RUN.
ENDRUN.
```

Chapter

6

Hash methods

Common information

Hash methods play an important role in the field of security. They are used every time a definite value needs to be calculated for input data of arbitrary size. The resulting check sum can later on be used to verify the integrity of the data.

CryptLib supports the hash methods **MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512** and **RipeMD160.** Besides these **HMAC** (Keyed-Hashing for Message Authentication) is supported.

Use the following steps to create a hash value:

§ Initialize the context using the *DIGEST-INIT* function to specify the desired hash method.

§ Add data to be hashed using the *DIGEST-UPDATE* function. This method can be called as often as needed.

§ Call the *DIGEST-FINAL* method to finalize the procedure and to get hold of the hash value.

Methods

DIGEST-INIT

Initialize the hash context.

Syntax	**Cobol**	

	`MOVE DIGEST-INIT TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`HASHALGO, CTX, RC.`	

	Assembler	
	`CALL XPSCRYPT,(DIGEST_INIT,HASHALGO,CTX,RC),VL`	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
HASHALGO	The type of hash to be used. CryptLib supports MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160.	Input
CTX	Address pointer to receive the storage address of the created context. This will be used for subsequent processing.	Output

DIGEST-UPDATE

Add data to be hashed.

Syntax	Cobol	
	MOVE DIGEST-UPDATE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, DATALEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(DIGEST_UPDATE,CTX,DATA,DATALEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context initialized for hashing.	Input
DATA	Storage address of data to be hashed.	Input
DATALEN	Length of the data to be hashed.	Input

DIGEST-FINAL

Create the hash value.

Syntax	Cobol	
	MOVE DIGEST-FINAL TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, HASHDATA, HASHLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(DIGEST_FINAL,CTX,HASHDATA,HASHLEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context initialized for hashing.	Input
HASHDATA	Address of storage area about to hold the created hash value.	Output
HASHLEN	Address of storage area about to hold the length of the created hash value.	Output

Cobol example:

```
*-----------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM: HASH-ROUTINES                 *
*-----------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     HASHTSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DATA1           PIC X(32)      VALUE "XPS Software GmbH, Haar
-                                         "/Muenchen".
 01  DATA2           PIC X(21)      VALUE "Muenchener Strasse 17".
 01  TESTKEY         PIC X(7)       VALUE "testkey".
 01  HASHSHA1        PIC 9(8)  COMP VALUE 26.
 01  HASHMD5         PIC 9(8)  COMP VALUE 5.
 01  DIGEST-MD5      PIC X(16).
 01  DIGEST-SHA1     PIC X(20).
 01  DIGEST-HMAC     PIC X(20).
 01  HASHCTX         POINTER.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DATALEN1        PIC 9(8)   COMP  VALUE 32.
```

```
 77  DATALEN2        PIC 9(8)   COMP  VALUE 21.
 77  KEYLEN          PIC 9(8)   COMP  VALUE 7.
 77  HASHMD5-LEN     PIC 9(8)   COMP  VALUE 16.
 77  HASHSHA1-LEN    PIC 9(8)   COMP  VALUE 20.
 77  RC              PIC S9(8)  COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* INITIALIZE HASH-CONTEXT MD5                                  *
* ------------------------------------------------------------ *
     MOVE DIGEST-INIT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHMD5, HASHCTX, RC.
     IF RC < 0
        DISPLAY "ERROR DIGEST-INIT: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* UPDATE DATA                                                  *
* ------------------------------------------------------------ *
     MOVE DIGEST-UPDATE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHCTX, DATA1, DATALEN1, RC.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHCTX, DATA2, DATALEN2, RC.
* ------------------------------------------------------------ *
* FINALIZE HASH                                                *
* ------------------------------------------------------------ *
     MOVE DIGEST-FINAL TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHCTX, DIGEST-MD5, HASHMD5-LEN, RC.
     IF RC < 0
        DISPLAY "ERROR DIGEST-FINAL: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* INITIALIZE HASH-CONTEXT SHA-1                                *
* ------------------------------------------------------------ *
     MOVE DIGEST-INIT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHSHA1, HASHCTX, RC.
     IF RC < 0
        DISPLAY "ERROR DIGEST-INIT: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* UPDATE DATA                                                  *
* ------------------------------------------------------------ *
     MOVE DIGEST-UPDATE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHCTX, DATA1, DATALEN1, RC.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHCTX, DATA2, DATALEN2, RC.
* ------------------------------------------------------------ *
* FINALIZE HASH                                                *
* ------------------------------------------------------------ *
     MOVE DIGEST-FINAL TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HASHCTX, DIGEST-SHA1, HASHSHA1-LEN, RC.
     IF RC < 0
        DISPLAY "ERROR DIGEST-FINAL: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* HMAC                                                         *
* ------------------------------------------------------------ *
     MOVE HMAC TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, DATA1, DATALEN1,
         TESTKEY, KEYLEN, DIGEST-HMAC, HASHSHA1, RC.
     IF RC < 0
        DISPLAY "ERROR HMAC: RC = " RC
        GOBACK.
     STOP RUN.
 ENDRUN.
```

HMAC

Create a MAC. A Message-Authentication-Code or MAC is a key dependent one way hash method. Therefore a MAC can only be created or verified using a key. This prevents firstly a MAC secured message to be changed by an unauthorized user who does not possess the key and secondly the MAC from unauthorized re-calculation. Thus a MAC can be used to guarantee the integrity of data without the need for data encryption.

Syntax	Cobol	
	MOVE HMAC TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **DATA, DATALEN, PWD, PWDLEN, HMAC, HASHALGO, RC.**	
	Assembler	
	CALL XPSCRYPT,(HMAC,DATA,DATALEN,PWD,PWDLEN,HMAC,HASHALGO,RC),VL	
Return code	Length of the HMAC or error code (< 0).	
Parameter	**Description**	**Use**
DATA	Storage address of the data.	Input
DATALEN	Length of the data.	Input
KEY	Storage address of the key.	Input
KEYLEN	Length of the key.	Input
HMAC	Address of storage area about to hold the created hash value.	Output
HASHALGO	The type of hash to be used. CryptLib supports MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160.	Input

Chapter

7

X.509 Certificates

Common information

Certificates exist since the invention of public key algorithms. Sometimes certificates are called digital passports. Simply spoken a certificate is no more than a signed piece of data. Two parties are involved in the process of creating a certificate: the certificate issuer and the certificate subject. Both of them need to have an asymmetrical pair of keys.

In order for the subject to be certified he has to transmit his public key to the signer. The issuer creates a data record including the name of the issuer, the name of the subject and the subjects' public key. Finally the resulting data record will be signed with the issuer's private key.

This data record in conjunction with the signature is called the certificate. Thus the issuer attests the owner of the certificate that he is also the owner of the associated public key. Assuming the subject's public key is available the unsophistication of the electronical passport can be verified.

CryptLib offers the program developer methods to verify the unsophistication of certificates as well as methods to extract data stored in the certificate.

Methods

IMPORT-CERTIFICATE

Reading a certificate object and checking its formal correctness.

Syntax	**Cobol**	

	`MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CERT, CTX, RC.`	

	Assembler	
	`CALL XPSCRYPT,(IMPORT_CERTIFICATE,CERT,CTX,RC),VL`	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CERT	Storage address of a X.509 certificate. CryptLib supports binary and Base64 encrypted certificates.	Input
CTX	Address pointer to receive the storage address of the created certificate context. This will be used for subsequent processing of the certificate.	Output

GET-PUBLIC-KEY

Extract the public key from the certificate.

Syntax	Cobol	
	MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, PUBKEY, PUBLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_PUBLIC_KEY,CTX,PUBKEY,PUBLEN,RC),VL	
Return code	Length of the public key or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
PUBKEY	Address of the storage area about to hold the extracted public key.	Output
PUBLEN	Maximum size of the storage area reserved for the public key.	Input

GET-CRYPT-ALGO

Extract encryption algorithm of the public key.

Syntax	Cobol	
	MOVE GET-CRYPT-ALGO TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_CRYPT_ALGO,CTX,OUT,OUTLEN,RC),VL	
Return code	Length of the name of the encryption algorithm or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted name of the encryption algorithm.	Output
OUTLEN	Maximum size of the storage area reserved for the name.	Input

GET-CRYPT-KEYLEN

Extract the key length of the public key.

Syntax	Cobol	
	MOVE GET-CRYPT-KEYLEN TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_CRYPT_KEYLEN,CTX,RC),VL	
Return code	Length of the public key or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input

GET-VERSION-INFO

Extract version number of the certificate.

Syntax	Cobol	
	MOVE GET-VERSION-INFO TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_VERSION_INFO,OUT,OUTLEN,RC),VL	
Return code	Length of the version number or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted version number.	Output
OUTLEN	Maximum size of the storage area reserved for the version number.	Input

GET-SERIAL-NUMBER

Extract the serial number of the certificate.

Syntax	Cobol	
	MOVE GET-SERIAL-NUMBER TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_SERIAL_NUMBER,CTX,OUT,OUTLEN,RC),VL	
Return code	Length of the serial number or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted serial number.	Output
OUTLEN	Maximum size of the storage area reserved for the serial number.	Input

GET-ISSUER-DN

Extract information about the issuer.

Syntax	Cobol
	MOVE GET-ISSUER-DN TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**
	Assembler
	CALL XPSCRYPT,(GET_ISSUER_DN,CTX,OUT,OUTLEN,RC),VL
Return code	Length of issuer information or error code (< 0).

Parameter	Description	Use
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted information about the issuer. Single elements (CN, O, OU etc.) will be separated by comma.	Output
OUTLEN	Maximum size of the storage area reserved for the issuer information.	Input

GET-SUBJECT-DN

Extract information about the subject.

Syntax	Cobol	

	`MOVE GET-SUBJECT-DN TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, OUT, OUTLEN, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_SUBJECT_DN,CTX,OUT,OUTLEN,RC),VL`	
Return code	Length of subject information or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted information about the subject. Single elements (CN, O, OU etc.) will be separated by comma.	Output
OUTLEN	Maximum size of the storage area reserved for the subject information.	Input

GET-SIGNATURE-ALGO

Extract the signature algorithm used by the certificate signer.

Syntax	Cobol	

	`MOVE GET-SIGNATURE-ALGO TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, OUT, OUTLEN, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_SIGNATURE_ALGO,CTX,OUT,OUTLEN,RC),VL`	
Return code	Length of the name of the algorithm or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the name of the extracted encryption algorithm.	Output
OUTLEN	Maximum size of the storage area reserved for the name of the encryption algorithm.	Input

GET-SIGNATURE

Extract the signature of the certificate.

Syntax	Cobol	
	MOVE GET-SIGNATURE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_SIGNATURE,CTX,OUT,OUTLEN,RC),VL	
Return code	Length of the signature or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted signature.	Output
OUTLEN	Maximum size of the storage area reserved for the signature.	Input

GET-START-DATE

Extract begin of validity of the certificate.

Syntax	Cobol	
	MOVE GET-START-DATE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_START_DATE,CTX,OUT,OUTLEN,RC),VL	
Return code	Length of the date or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted start of validity. The date will be returned as DD.MM.YYYY HH:MM:SS.	Output
OUTLEN	Maximum size of the storage area reserved for the date.	Input

GET-END-DATE

Extract the end of validity of the certificate.

Syntax	Cobol	
	MOVE GET-END-DATE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_END_DATE,CTX,OUT,OUTLEN,RC),VL	
Return code	Length of the date or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the extracted end of validity. The date will be returned as DD.MM.YYYY HH:MM:SS.	Output

OUTLEN	Maximum size of the storage area reserved for the date.	Input

GET-ISSUER-DN-BLOB

Extract issuer's data in binary format including the ASN.1 control characters. The resulting BLOB may be used to follow the further path of certification.

Syntax	**Cobol**	
	MOVE GET-ISSUER-DN-BLOB TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_ISSUER_DN_BLOB,CTX,OUT,RC),VL	
Return ode	Length of the certificate issuer BLOB or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address pointer about to receive the address of the storage area holding the extracted issuer data BLOB.	Output

GET-SUBJECT-DN-BLOB

Extract subject's data in binary format including the ASN.1 control characters. The resulting BLOB may be used to follow the further path of certification.

Syntax	**Cobol**	
	MOVE GET-SUBJECT-DN-BLOB TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, KEY, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_SUBJECT_DN_BLOB,OUT,RC),VL	
Return code	Length of the certificate subject BLOB or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address pointer about to receive the address of the storage area holding the extracted subject data BLOB.	Output

GET-ISSUER-DN-BY-TYPE

Extract specific issuer element specified via parm *TYPE*.

Syntax	**Cobol**
	MOVE GET-ISSUER-DN-BY-TYPE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, TYPE, OUT, OUTLEN, RC.**
	Assembler
	CALL XPSCRYPT,(GET_ISSUER_DN_BY_TYPE,CTX,TYPE,OUT,OUTLEN,RC),VL

Return code	Length of issuer data or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
TYPE	Issuers' DN type. The following types are available: DN_C Country DN_SP State/Province DN_L Locality DN_O OrganizationName DN_OU OrganizationUnit DN_CN CommonName DN_EMAIL E-Mail DN_STREET Street DN_PHONE Phone DN_POSTAL PostalCode DN_TITLE Title	Input
OUT	Address of the storage area about to hold the extracted issuer data.	Output
OUTLEN	Maximum size of the storage area reserved for the issuer data.	Input

GET-SUBJECT-DN-BY-TYPE

Extract specific subject element specified via parm *TYPE*.

Syntax	**Cobol**	
	MOVE GET-SUBJECT-DN-BY-TYPE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, TYPE, OUT, OUTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_SUBJECT_DN_BY_TYPE,CTX,TYPE,OUT,OUTLEN,RC),VL	
Return code	Length of subject data or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
TYPE	Subjects' DN type. The following types are available: DN_C Country DN_SP State/Province DN_L Locality DN_O OrganizationName DN_OU OrganizationUnit DN_CN CommonName DN_EMAIL E-Mail DN_STREET Street DN_PHONE Phone DN_POSTAL PostalCode DN_TITLE Title	Input
OUT	Address of the storage area about to hold the extracted subject data.	Output
OUTLEN	Maximum size of the storage area reserved for the subject data.	Input

GET-FIRST-EXTENSION

The standard fields supported by X.509 certificates may not be sufficient for some applications. Because of this beginning with version 3 the X.509 syntax has been modified introducing an extension component. The extension component makes it possible to include arbitrary data in the certificate.

Syntax	**Cobol**	
	MOVE GET-FIRST-EXTENSION TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, EXTENSION, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_FIRST_EXTENSION,CTX,EXTENSION,RC),VL	
Return code	Length of the extension area (0 if not available) or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
EXTENSION	Address pointing to the following structure. The description of the required strcuture can be found in the copy books XPSCLEXT (COBOL) and XPSCLASM (Assembler) respectively.	Output
Felder	**Description**	
C-OIDBIN	Binary value of the OID (object identifier).	
C-OIDCHR	Character value of the OID (object identifier).	
C-ISCRIT	Flag to mark the extension as critical or uncritical. Critical extensions have to be considered always. If a program detects a critical extension and doesn't know it's meaning further use of the certificate should be avoided. Uncritical extensions may be ignored by the processing program.	
C-FTYPE	Field type of the extension. The following types are possible: BER_BOOLEAN BER_INTEGER BER_BITSTRING BER_OCTETSTRING BER_OBJECT_IDENTIFIER BER_OBJECT_DESCRIPTOR BER_EXTERNAL BER_REAL BER_ENUMERATED BER_EMBEDDED_PDV BER_STRING_UTF8 BER_RELATIVE_OID BER_STRING_NUMERIC BER_STRING_PRINTABLE BER_STRING_T61 BER_STRING_GRAPHIC BER_STRING_ISO646 BER_STRING_VIDEOTEXT BER_STRING_GENERAL BER_STRING_UNIVERSAL BER_CHAR_STRING BER_STRING_BMP	
C-VALUE	Integer value for the types BER_BOOLEAN, BER_INTEGER, BER_ENUMERATED.	
C-DATA	Storage address of the extensions binary data area.	
C-DLEN	Length of the extensions binary data area.	

GET-NEXT-EXTENSION

Extract the next certificate extension.

Syntax	**Cobol**	
	MOVE GET-NEXT-EXTENSION TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, EXTENSION, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_NEXT_EXTENSION,CTX,EXTENSION,RC),VL	
Return code	Length of the extension area (0 if not available) or error code (< 0).	
Parameter	**Description**	**Use**

CTX	Storage address of the certificate context.	Input
EXTENSION	Storage address of the extracted extension. For an explanation of the format of the extension area see method GET-FIRST-EXTENSION.	Output

GET-EXTENSION-BY-OID

Extract a certificate extension for a specific OID (object identifier).

Syntax	**Cobol**	
	MOVE GET-EXTENSION-BY-OID TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OID, EXTENSION, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_EXTENSION_BY_OID,CTX,OID,EXTENSION,RC),VL	
Return ode	Lentth of the extension area (0 if not available) or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OID	Storage address of the binary object identifier about to be extracted.	Input
EXTENSION	Storage address of the extracted extension. For an explanation of the format of the extension area see method GET-FIRST-EXTENSION.	Output

GET-FINGERPRINT

Create a fingerprint (hash value) for the certificate. A fingerprint may be used to enable visual examination of a certificate.

Syntax	**Cobol**	
	MOVE GET-FINGERPRINT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OUT, OUTLEN, HASHALGO, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_FINGERPRINT,CTX,OUT,OUTLEN,HASHALGO,RC),VL	
Return code	Length of the fingerprint or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
OUT	Address of the storage area about to hold the created fingerprint.	Output
OUTLEN	Maximum size of the storage area reserved for the fingerprint.	Input
HASHALGO	The type of hash to be used. CryptLib supports MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160.	Input

VERIFY-CERTIFICATE

Check the validity of a certificate.

Syntax	**Cobol**
	MOVE VERIFY-CERTIFICATE TO CRYPT-FUNCTION.

	CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, PUBKEY, RC.	
	Assembler	
	CALL XPSCRYPT,(VERIFY_CERTIFICATE,CTX,PUBKEY,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input
PUBKEY	Issuer's public key.	Input

CLEANUP-CERTIFICATE

Deallocate storage used by certificate routines.

Syntax	**Cobol**	
	MOVE CLEANUP-CERTIFICATE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(CLEANUP_CERTIFICATE,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the certificate context.	Input

COBOL example:

```
*-----------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM: GET X.509 CERTIFICATE         *
*-----------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     CERTTSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)         VALUE "XPSDATA".
 01  CERTUSER        PIC X(8)         VALUE "XPSUSERC".
 01  CERTTRST        PIC X(8)         VALUE "XPSTESTC".
 01  OUT             PIC X(1024)      VALUE LOW-VALUES.
 01  HEADER          PIC X(20)        VALUE SPACES.
 01  HEADER1         PIC X(20)        VALUE "VERSION:".
 01  HEADER2         PIC X(20)        VALUE "SERIAL-NUMBER:".
 01  HEADER3         PIC X(20)        VALUE "ISSUER-DN:".
 01  HEADER4         PIC X(20)        VALUE "START-DATE:".
 01  HEADER5         PIC X(20)        VALUE "END-DATE:".
 01  HEADER6         PIC X(20)        VALUE "CRYPT-ALGO:".
 01  HEADER7         PIC X(20)        VALUE "CRYPT-KEYLEN:".
 01  HEADER8         PIC X(20)        VALUE "PUBLIC-KEY:".
 01  HEADER9         PIC X(20)        VALUE "SIGNATURE-ALGO:".
 01  HEADER10        PIC X(20)        VALUE "SIGNATURE:".
 01  HEADER11        PIC X(20)        VALUE "EXTENSION-OID:".
 01  HEADER12        PIC X(20)        VALUE "EXTENSION-VALUE:".
 01  HEADER13        PIC X(20)        VALUE "FINGERPRINT-SHA1:".
 01  HEADER14        PIC X(20)        VALUE "FINGERPRINT-MD5:".
 01  OID             PIC X(32)    VALUE X"06096086480186F8420103".
 01  HASHSHA         PIC 9(8)   COMP  VALUE 26.
 01  HASHMD5         PIC 9(8)   COMP  VALUE 5.
 01  CERT-CTX        POINTER.
 01  ADDR-CERT       POINTER.
 01  DNTAB.
     05 DNNAME.
        10 PIC X(8)      VALUE "CN:".
        10 PIC X(8)      VALUE "O:".
        10 PIC X(8)      VALUE "OU:".
        10 PIC X(8)      VALUE "L:".
        10 PIC X(8)      VALUE "SP:".
```

```
           10 PIC X(8)      VALUE "STREET:".
           10 PIC X(8)      VALUE "POSTAL:".
           10 PIC X(8)      VALUE "C:".
           10 PIC X(8)      VALUE "EMAIL:".
           10 PIC X(8)      VALUE "PHONE:".
           10 PIC X(8)      VALUE "TITEL:".
        05 DNID.
           10 PIC 9(8) COMP VALUE 6.
           10 PIC 9(8) COMP VALUE 4.
           10 PIC 9(8) COMP VALUE 5.
           10 PIC 9(8) COMP VALUE 3.
           10 PIC 9(8) COMP VALUE 2.
           10 PIC 9(8) COMP VALUE 8.
           10 PIC 9(8) COMP VALUE 10.
           10 PIC 9(8) COMP VALUE 1.
           10 PIC 9(8) COMP VALUE 7.
           10 PIC 9(8) COMP VALUE 9.
           10 PIC 9(8) COMP VALUE 11.
 01  REDEFINES DNTAB.
        05 DN-NAME PIC X(8)      OCCURS 11.
        05 DN-ID   PIC 9(8) COMP OCCURS 11.
*
 COPY XPSCLRSA.
 COPY XPSCLEXT.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DATALEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  DUMPLEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  OUTLEN          PIC 9(8)   COMP  VALUE 1024.
 77  RC              PIC S9(8)  COMP  VALUE ZEROES.
 77  IX              PIC S9(2)        VALUE ZERO.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
 01  EXTDATA         PIC X.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* READ X.509 FILE "XPSTESTC" FROM MACLIB (TRUSTED-SIGNER CERT)  *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, CERTTRST, ADDR-CERT, DATALEN, RC.
     IF RC < 0
        DISPLAY "FILE 'XPSTESTC' NOT FOUND: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* IMPORT X.509 CERTIFICATE               (TRUSTED-SIGNER CERT)  *
* ------------------------------------------------------------ *
     MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         ADDR-CERT, CERT-CTX, RC.
     IF RC < 0
        DISPLAY "ERROR IMPORT-CERTIFICATE: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* GET PUBLIC-KEY FROM CERTIFICATE        (TRUSTED-SIGNER CERT)  *
* ------------------------------------------------------------ *
     MOVE RSA-PUBLIC-LEN TO DATALEN
     MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         CERT-CTX, RSA-PUBLIC-KEY, DATALEN, RC.
     IF RC < 0
        DISPLAY "ERROR GET-PUBLIC-KEY: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* RELEASE CERTIFICATE CONTEXT            (TRUSTED-SIGNER CERT)  *
* ------------------------------------------------------------ *
     MOVE CLEANUP-CERTIFICATE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         CERT-CTX, RC.
     IF RC < 0
        DISPLAY "ERROR CLEANUP-CERTIFICATE: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* RELEASE FILE-STORAGE "XPSTESTC"                               *
* ------------------------------------------------------------ *
     MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         ADDR-CERT, RC.
     IF RC < 0
        DISPLAY "ERROR RELEASE-FILE: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* READ X.509 FILE "XPSUSERC" FROM MACLIB (USER-CERTIFICATE)     *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
```

```
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION, DDNAME,
               CERTUSER, ADDR-CERT, DATALEN, RC.
           IF RC < 0
              DISPLAY "FILE 'XPSUSERC' NOT FOUND: RC = " RC
              GOBACK.
       * ------------------------------------------------------------ *
       * IMPORT X.509 CERTIFICATE               (USER-CERTIFICATE)    *
       * ------------------------------------------------------------ *
           MOVE IMPORT-CERTIFICATE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               ADDR-CERT, CERT-CTX, RC.
           IF RC < 0
              DISPLAY "ERROR IMPORT-CERTIFICATE: RC = " RC
              GOBACK.
       * ------------------------------------------------------------ *
       * VERIFY USER-CERTIFICATE WITH TRUSTED-SIGNER PUBLIC-KEY       *
       * ------------------------------------------------------------ *
           MOVE VERIFY-CERTIFICATE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               CERT-CTX, RSA-PUBLIC-KEY, RC.
           IF RC < 0
              DISPLAY "ERROR VERIFY: " RC
              GOBACK.
       * ------------------------------------------------------------ *
       * THE CERTIFICATE IS OKAY, SO WE CAN GET DATA FROM THE CERT    *
       * ------------------------------------------------------------ *
           MOVE GET-VERSION-INFO TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               CERT-CTX, OUT, OUTLEN, RC.
           IF RC < 0
              DISPLAY "ERROR GET-VERSION-INFO: RC = " RC
              GOBACK.
           MOVE RC TO DUMPLEN.
           MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               HEADER1, OUT, DUMPLEN, RC.
       *
           MOVE GET-SERIAL-NUMBER TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               CERT-CTX, OUT, OUTLEN, RC.
           IF RC < 0
              DISPLAY "ERROR GET-SERIAL-NUMBER: RC = " RC
              GOBACK.
           MOVE RC TO DUMPLEN.
           MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               HEADER2, OUT, DUMPLEN, RC.
       *
           MOVE GET-ISSUER-DN TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               CERT-CTX, OUT, OUTLEN, RC.
           IF RC < 0
              DISPLAY "ERROR GET-ISSUER-DN: RC = " RC
              GOBACK.
           MOVE RC TO DUMPLEN.
           MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               HEADER3, OUT, DUMPLEN, RC.
       *
           MOVE GET-START-DATE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               CERT-CTX, OUT, OUTLEN, RC.
           IF RC < 0
              DISPLAY "ERROR GET-START-DATE: RC = " RC
              GOBACK.
           MOVE RC TO DUMPLEN.
           MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               HEADER4, OUT, DUMPLEN, RC.
       *
           MOVE GET-END-DATE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               CERT-CTX, OUT, OUTLEN, RC.
           IF RC < 0
              DISPLAY "ERROR GET-END-DATE: RC = " RC
              GOBACK.
           MOVE RC TO DUMPLEN.
           MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               HEADER5, OUT, DUMPLEN, RC.
       *
           PERFORM VARYING IX
                   FROM 1 BY 1
                   UNTIL IX > 11
               MOVE GET-SUBJECT-DN-BY-TYPE TO CRYPT-FUNCTION
               CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
                   CERT-CTX, OUT, OUTLEN, DN-ID (IX), RC
               IF RC > 0
                 MOVE DN-NAME (IX) TO HEADER
                 MOVE RC TO DUMPLEN
```

```
           MOVE DUMP-STORAGE TO CRYPT-FUNCTION
           CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
               HEADER, OUT, DUMPLEN, RC
         END-IF
      END-PERFORM.
*
      MOVE GET-CRYPT-ALGO TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, OUT, OUTLEN, RC.
      IF RC < 0
         DISPLAY "ERROR GET-CRYPT-ALGO: RC = " RC
         GOBACK.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER6, OUT, DUMPLEN, RC.
*
      MOVE GET-CRYPT-KEYLEN TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, RC.
      IF RC < 0
         DISPLAY "ERROR GET-CRYPT-KEYLEN: RC = " RC
         GOBACK.
      MOVE 4 TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER7, RC, DUMPLEN, RC.
*
      MOVE GET-PUBLIC-KEY TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, OUT, OUTLEN, RC.
      IF RC < 0
         DISPLAY "ERROR GET-PUBLIC-KEY: RC = " RC
         GOBACK.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER8, OUT, DUMPLEN, RC.
*
      MOVE GET-SIGNATURE-ALGO TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, OUT, OUTLEN, RC.
      IF RC < 0
         DISPLAY "ERROR GET-SIGNATURE-ALGO: RC = " RC
         GOBACK.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER9, OUT, DUMPLEN, RC.
*
      MOVE GET-SIGNATURE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, OUT, OUTLEN, RC.
      IF RC < 0
         DISPLAY "ERROR GET-SIGNATURE: RC = " RC
         GOBACK.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER10, OUT, DUMPLEN, RC.
*
      MOVE GET-FIRST-EXTENSION TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, CERT-EXT, RC.
      PERFORM UNTIL RC <= ZEROES
        PERFORM GET-EXTENSION
      END-PERFORM.
*
      MOVE GET-EXTENSION-BY-OID TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, CERT-EXT, OID, RC.
      IF RC > ZEROES
        PERFORM GET-EXTENSION
      END-IF.
*
      MOVE GET-FINGERPRINT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, OUT, OUTLEN, HASHSHA, RC.
      IF RC < 0
         DISPLAY "ERROR GET-FINGERPRINT: RC = " RC
         GOBACK.
      MOVE RC TO DUMPLEN.
      MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          HEADER13, OUT, DUMPLEN, RC.
*
      MOVE GET-FINGERPRINT TO CRYPT-FUNCTION.
      CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
          CERT-CTX, OUT, OUTLEN, HASHMD5, RC.
      IF RC < 0
```

```
           DISPLAY "ERROR GET-FINGERPRINT: RC = " RC
           GOBACK.
       MOVE RC TO DUMPLEN.
       MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
       CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           HEADER14, OUT, DUMPLEN, RC.
       STOP RUN.
*
 GET-EXTENSION SECTION.
       MOVE 32 TO DUMPLEN.
       MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
       CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           HEADER11, C-OIDCHR, DUMPLEN, RC.
       IF C-DLEN = ZEROES
          MOVE 4 TO DUMPLEN
          MOVE DUMP-STORAGE TO CRYPT-FUNCTION
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              HEADER12, C-VALUE, DUMPLEN, RC
       ELSE
          MOVE C-DLEN TO DUMPLEN
          SET ADDRESS OF EXTDATA TO C-DATA
          MOVE DUMP-STORAGE TO CRYPT-FUNCTION
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              HEADER12, EXTDATA, DUMPLEN, RC
       END-IF.
       MOVE GET-NEXT-EXTENSION TO CRYPT-FUNCTION.
       CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
           CERT-CTX, CERT-EXT, RC.
 GET-EXTENSION-END.
       EXIT.
 ENDRUN.
```

Chapter

8

S/MIME Objects (PKCS#7)

Common information

PKCS#7 also known as *Cryptographic-Message-Syntax-Standard* describes methods to secure data using cryptographic procedures such as digital signatures or encryption. CryptLib supports the following content types:

Data	Simply used to model data. This type offers no cryptographic functions.
Signed-data	Describes a format to ensure data integrity and sender authenticity by means of digital signatures and certificates.
Enveloped-data	Used to encrypt data in a receiver specific way to disable trespassers from reading the message (confidentiality).
Encrypted-data	Used to encrypt data.

Methods

IMPORT-PKCS7-DATA

Read a PKCS#7 data object and guarantee its formal correctness. The content type *Data* describes an arbitrary sequence of data bytes. Data may be retrieved using the methods *GET-FIRST-DATA* and *GET-NEXT-DATA*.

Syntax	**Cobol**	

	`MOVE IMPORT-PKCS7-DATA TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`APKCS7, PKCS7LEN, CTX, RC.`	

	Assembler	
	`CALL XPSCRYPT,(IMPORT_PKCS7_DATA,APKCS7,PKCS7LEN,CTX,RC),VL`	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
APKCS7	Storage address of the PKCS#7 data object. Binary, Base64 encrypted and S/MIME formats are supported.	Input
PKCS7LEN	Length of the PKCS#7 data object.	Input
CTX	Storage address of the imported PKCS#7 context. This object will be needed for subsequent processing.	Output

IMPORT-SIGNED-DATA

Read a PKCS#7 signed data object and guarantee its formal correctness. The content type *Signed-data* defines syntax for calculation and transport of digital signatures. The message may be signed by an arbitrary number of signers.

Data can be extracted using the methods *GET-FIRST-DATA* and *GET-NEXT-DATA* respectively. Signers can be leached using the methods *GET-FIRST-SIGNER* and *GET-NEXT-SIGNER* respectively. The methods *VERIFY-SIGNER* and *VERIFY-ALL-SIGNER* can be used to examine data integrity. Finally the *ADD-SIGNER-CERT* method can be used to add certificate issuers.

Syntax	**Cobol**	
	MOVE IMPORT-SIGNED-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **APKCS7, PKCS7LEN, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(IMPORT_SIGNED_DATA,APKCS7,PKCS7LEN,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
APKCS7	Storage address of the PKCS#7 data object. Binary, Base64 encrypted and S/MIME formats are supported.	Input
PKCS7LEN	Length of the PKCS#7 data object.	Input
CTX	Storage address of the imported PKCS#7 context. This object will be needed for subsequent processing.	Output

IMPORT-ENVELOPED-DATA

Read a PKCS#7 enveloped data object and guarantee its formal correctness. The content type *Enveloped-data* defines syntax for receiver specific message encryption. This means that information about the intended receiver may be part of the message. This is carried out using a technique called 'digital enveloping'.

Equivalent to the *Signed-data* type accepting an arbitrary number of signers the *Enveloped-data* type permits the inclusion of multiple message receivers. Information about specific receivers is included using the type 'RecipientInfo'.

Data can be extracted using the methods *GET-FIRST-DATA* and *GET-NEXT-DATA* respectively.

Syntax	**Cobol**	
	MOVE IMPORT-ENVELOPED-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **APKCS7, PKCS7LEN, APKCS12, PKCS12LEN, PWD, PWDLEN, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(IMPORT_ENVELOPED_DATA,APKCS7,PKCS7LEN,APKCS12, **PKCS12LEN,PWD,PWDLEN,CTX,RC),VL**	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
APKCS7	Storage address of the PKCS#7 data object. Binary, Base64 encrypted and S/MIME formats are supported.	Input
PKCS7LEN	Length of the PKCS#7 encrypted data object.	Input
APKCS12	Storage address of a PKCS#12 object. PKCS#12 objects define syntax	Input

	for the exchange of keys and certificates. Included in PKCS#12 objects are key-bags and certificate-bags. Using the certificate-bags it will be possible to find out if the PKCS#7 object contains a RecipientInfo for the user. Assuming a RecipientInfo is contained the users private key can be extracted from the key-bag. This private key is used to encrypt the *Content-Encryption* key which in turn can be used to decrypt the data.	
PKCS12LEN	Length of the PKCS#12 object.	Input
PWD	PKCS#12 objects are sealed with a password. Using this parameter the storage address of the password has to be made known.	Input
PWDLEN	Length of the password.	Input
CTX	Storage address of the imported PKCS#7 context. This object will be needed for subsequent processing.	Output

IMPORT-ENCRYPTED-DATA

Read a PKCS#7 encrypted data object and guarantee its formal correctness. The content type *Encrypted-data* defines syntax for message encryption. Unlike *Enveloped-data* processing it's assumed the receiver does already posses the *Content-Encryption* key thus making it superfluous to explicitly specify it.

This type mainly qualifies for storing encrypted data. As a prominent application the *Personal-Information-Syntax-Standard* PKCS#12 can be named.

Data can be extracted using the methods *GET-FIRST-DATA* and the *GET-NEXT-DATA* respectively.

Syntax	**Cobol**	
	MOVE IMPORT-ENCRYPTED-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **APKCS7, PKCS7LEN, PWD, PWDLEN, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(IMPORT_ENCRYPTED_DATA,APKCS7,PKCS7LEN,PWD,PWDLEN, **CTX,RC),VL**	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
APKCS7	Storage address of the PKCS#7 data object. Binary, Base64 encrypted and S/MIME formats are supported.	Input
PKCS7LEN	Length of the PKCS#7 encrypted data object.	Input
PWD	Address of storage area holding the password used to encrypt the *Content-Encryption* key.	Input
PWDLEN	Length of the password.	Input
CTX	Storage address of the imported PKCS#7 context. This object will be needed for subsequent processing.	Output

CREATE-PKCS7-DATA

Create a PKCS#7 data object. PKCS#7 data objects are used to model data and don't offer any cryptographic methods. Data can be added using the method *ADD-PKCS7-DATA*.

Syntax	**Cobol**

	MOVE CREATE-PKCS7-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, CTX, RC.**		
	Assembler		
	CALL XPSCRYPT,(CREATE_CTX_DATA,OPTION,CTX,RC),VL		
Return code	0 or error code (< 0).		
Parameter	**Description**		**Use**
OPTION	HEADER_INCLUDED	If this option is given the ContentType will be added.	Input
CTX	Storage address of the created PKCS#7 context.		Output

CREATE-SIGNED-DATA

Create a PKCS#7 signed data object. The content type *Signed-data* defines syntax for calculation and transport of digital signatures. The number of parties signing a message can be arbitrary. Signers can be added using the method *ADD-SIGNER*, using the method *ADD-PKCS7-DATA* data can be added. Additional certificates can be added using the method *ADD-SIGNER-CERT*.

Syntax	**Cobol**		
	MOVE CREATE-SIGNED-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, CTX, RC.**		
	Assembler		
	CALL XPSCRYPT,(CREATE_CTX_DATA,OPTION,CTX,RC),VL		
Return code	0 or error code (< 0).		
Parameter	**Description**		**Use**
OPTION	HEADER_INCLUDED	If this option is given the ContentType will be added.	Input
	DATA_IMPLICIT	If this option is specified the message text will be added to the signed-data object. This means that the content field will be available to include the message text. If this option is not specified the content field will be absent and the message text has to be transferred using a different way.	
	CERT_IMPLICIT	If this option is specified all signer certificates contained in the PKCS#12 file will be included in the signed-data object.	
CTX	Storage address of the created PKCS#7 context.		Output

CREATE-ENVELOPED-DATA

Create a PKCS#7 enveloped data object. The content type *Enveloped-data* defines syntax for receiver specific message encryption. This means that information about the intended receiver may be part of the message. This is carried out using a technique called 'digital enveloping'.

Equivalent to the *Signed-data* type accepting an arbitrary number of signers the *Enveloped-data* type permits the inclusion of multiple message receivers.

Receivers can be added using the method *ADD-RECIPIENT* while data can be added using the method *ADD-PKCS7-DATA*.

Syntax	Cobol		
	MOVE CREATE-ENVELOPED-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **OPTION, CTX, ENCRALGO, RC.**		
	Assembler		
	CALL XPSCRYPT,(CREATE_ENVELOPED_DATA,OPTION,CTX,ENCRALGO,RC),VL		
Return code	0 or error code (< 0).		
Parameter	**Description**		**Use**
OPTION	HEADER_INCLUDED	If this option is given the ContentType will be added.	Input
CTX	Storage address of the created PKCS#7 context.		Output
ENCRALGO	Algorithm to use for data encryption. CryptLib supports the following algorithms: DESEDE3CBC DESCBC RC2CBC_128 RC2CBC_64 RC2CBC_40 RC4_128 RC4_64 RC4_40	 Triple DES 168Bit DES 56Bit RC2 128Bit RC2 64Bit RC2 40Bit RC4 128Bit RC4 64Bit RC4 40Bit	Input

CREATE-ENCRYPTED-DATA

Create a PKCS#7 encrypted data object. The content type *Encrypted-data* defines syntax for message encryption. Unlike *Enveloped-data* processing it's assumed the receiver does already posses the *Content-Encryption* key thus making it superfluous to explicitly specify it.

This type mainly qualifies for storing encrypted data.

Data can be added using the method *ADD-PKCS7-DATA*.

Syntax	Cobol		
	MOVE CREATE-ENCRYPTED-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **OPTION, CTX, PBEALGO, PWD, PWDLEN, RC.**		
	Assembler		
	CALL XPSCRYPT,(CREATE_ENCRYPTED_DATA,OPTION,CTX,PBEALGO,PWD, **PWDLEN,RC),VL**		
Return code	0 or error code (< 0).		
Parameter	**Description**		**Use**
OPTION	HEADER_INCLUDED	If this option is given the ContentType will be added.	Input
CTX	Storage address of the created PKCS#7 context.		Output
PBEALGO	Algorithm to use for data encryption. CryptLib supports the following algorithms PBE3DES_3Key PBE3DES_2Key PBERC2_128 PBERC2_40 PBERC4_128	 Triple DES 168Bit Triple DES 112Bit RC2 128Bit RC2 40Bit RC4 128Bit	Input

	PBERC4_40 RC4 40Bit	
PWD	Storage address of the password used for key generation.	Input
PWDLEN	Length of the password.	Input

ADD-PKCS7-DATA

Import methods
Signed data objects are able to process data IMPLICIT or EXPLICT. Choosing IMPLICIT mode has the effect that data will be included in the signed data object. This means more precisely that the *content* field including the message content will be present. If EXPLICIT mode is chosen the *content* field will be absent meaning the message content has to be transferred any other way. This method offers the possibility to transmit data to the signed data object.

Create methods
Hereby the methods *CREATE-PKCS7-DATA, CREATE-SIGNED-DATA, CREATE-ENVELOPED-DATA* and *CREATE-ENCRYPTED-DATA* are used to add data to the PKCS#7 object.

Syntax	**Cobol**	
	MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, DATALEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ADD_PKCS7_DATA,CTX,DATA,DATALEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
DATA	Storage address of the data about to add.	Input
DATALEN	Length of data.	Input

ADD-MESSAGE-DIGEST

Signed data objects being processd in EXPLICIT mode can have a Message Digest (hash value calculated for the data) added. If an externally calculated Message Digest is added the method *ADD-PKCS7-DATA* must not be called.

Syntax	**Cobol**	
	MOVE ADD-MESSAGE-DIGEST TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, DATALEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ADD_MESSAGE_DIGEST,CTX,DATA,DATALEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	The storage address of the PKCS#7 context.	Input
DATA	The storage address of the Message Digest.	Input
DATALEN	The length of the Message Digest.	Input

ADD-SIGNER

A Signed-data object needs to be signed by one or more signers. Using this method based on a PKCS#12 file which includes the secret key as well as the signer's X.509 certificate a *SIGNERINFO* structure ready to sign the object will be created.

Syntax	**Cobol**	
	MOVE ADD-SIGNER TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, APKCS12, PKCS12LEN, PWD, PWDLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ADD_SIGNER,CTX,APKCS12,PKCS12LEN,PWD,PWDLEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
APKCS12	Storage address of the signers PKCS#12 object.	Input
PKCS12LEN	Length of the PKCS#12 object.	Input
PWD	Storage address of the password used to encrypt the PKCS#12 object.	Input
PWDLEN	Length of the password.	Input

ADD-RECIPIENT

While creating an *EnvelopedData* object, information about the intended receiver has to be added. This means the message has to be receiver specificly encrypted. Using this method the receiver's X.509 certificate is made known. The certificate includes the public key which will be used to encrypt the symmetrical *Content-Encryption* key.

Syntax	**Cobol**	
	MOVE ADD-RECIPIENT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, CERT, CERTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ADD_RECIPIENT,CTX,CERT,CERTLEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
CERT	Storage address of the receivers X.509 certificate.	Input
CERTLEN	The length of the certificate.	Input

ADD-SIGNER-CERT

Import methods
In order to verify a *Signed-data* object the hierarchy of signer certificates needs to be validated. If the PKCS#7 object doesn't contain signer certificates this methods provides the possibility to deliver signer certificates to the *Signed-data* object.

Create methods
When the *CREATE-SIGNED-DATA* method is executed all signer certificates stored in the PKCS#12 file will be added. Additional certificates can be added using this method.

Syntax	**Cobol**

	MOVE ADD-SIGNER-CERT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, CERT, CERTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ADD_SIGNER_CERT,CTX,CERT,CERTLEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
CERT	Storage address of the signers X.509 certificate.	Input
CERTLEN	Length of the certificate.	Input

ADD-TRUSTED-SIGNER

During the process of verification of a Signed-data object the signer's certificate can be examined for a trusted signer. Using this method, certificates of trusted signers can be added.

Syntax	**Cobol**	
	MOVE ADD-TRUSTED-SIGNER TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, CERT, CERTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ADD_TRUSTED_SIGNER,CTX,CERT,CERTLEN,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
CERT	Storage address of the trusted signers X.509 certificate.	Input
CERTLEN	The length of the certificate.	Input

FORCE-TRUSTED-SIGNER

Using this function prior to the verification of a *Signed-data* object it's possible to overwrite a certificate with an equal identity (issuer- and subject-blob) that might be already contained in the the PKCS#7 object.

Syntax	**Cobol**	
	MOVE FORCE-TRUSTED-SIGNER TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, CERT, CERTLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(FORCE_TRUSTED_SIGNER,CTX,CERT,CERTLEN,RC),VL	
Returncode (RC)	0 oder Fehlercode (< 0).	
Parameter	**Beschreibung**	**Verwendung**
CTX	Speicheradresse des PKCS#7-Kontexts.	Eingabe
CERT	Speicheradresse des X.509 Zertifikates des vertrauenswürdigen Ausstellers.	Eingabe
CERTLEN	Länge des Zertifikates.	Eingabe

GET-FIRST-SIGNER

Signed-data objects may be examined for information about the signers (SIGNERINFOS). Using this method, the first SIGNERINFO structure can be extracted.

Syntax	Cobol	

	`MOVE GET-FIRST-SIGNER TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, SIGNER, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_FIRST_SIGNER,CTX,SIGNER,RC),VL`	
Return code	0 if a signer is available else < 0.	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
SIGNER	Storage address of the extracted SIGNERINFO. This structure includes information about the certificate, the serial number, the issuer, the attributes, the Message Digest, the digest algorithm and the signature. The structures description named SIGNINFO can be found in the copy books *XPSCLCOB* (COBOL) and *XPSCLASM* (Assembler) respectively.	Output

GET-NEXT-SIGNER

Signed-data objects may be examined for information about the signers (SIGNERINFOS). Using this method, subsequent SIGNERINFO structures can be extracted.

Syntax	Cobol	

	`MOVE GET-NEXT-SIGNER TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, SIGNER, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_NEXT_SIGNER,CTX,SIGNER,RC),VL`	
Return code	0 if a signer is available else < 0.	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
SIGNER	Storage address of the extracted SIGNERINFO. This structure includes information about the certificate, the serial number, the issuer, the attributes, the Message Digest, the digest algorithm and the signature. The structures description named SIGNINFO can be found in the copy books *XPSCLCOB* (COBOL) and *XPSCLASM* (Assembler) respectively.	Output

GET-SIGNING-ALGO

Signed-data objects may be examined for information about the used encryption- and hash-algorithm. This method needs a SingerInfo structure for input which will be returned from calling *GetFirstSigner* and *GetNextSigner*.

Syntax	Cobol

	MOVE GET-SIGNING-ALGO TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, SIGNER, SIGALGO, ALGOLEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_SIGNING_ALGO,CTX,SIGNER,SIGALGO,ALGOLEN,RC),VL	
Return code	Length of the extracted *AlgoInfo*.	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
SIGNER	Storage address of the SIGNERINFO.	Input
SIGALGO	Address of storage area to use to store the required information. This information will be returned as a string containing the encryption algorithm and the hash algorithm separated by a slash character e.g "rsaEncryption/sha-1".	Input/Output
ALGOLEN	Length of the storage area available for the *AlgoInfo*.	Input

GET-SIGNING-TIME

Signed-data objects may be examined for information about the signing time. This method needs a SingerInfo structure for input which will be returned from calling *GetFirstSigner* and *GetNextSigner*.

Syntax	**Cobol**	
	MOVE GET-SIGNING-TIME TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, SIGNER, SIGTIME, TIMELEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_SIGNING_TIME,CTX,SIGNER,SIGTIME,TIMELEN,RC),VL	
Returnccode	Stringlänge von *SIGTIME*.	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
SIGNER	Storage address of the SIGNERINFO.	Input
SIGTIME	Address of storage area to use to store the required information. The signing time will be returned as a string with the following format: "YYMMDDHHMMZ". The single characters have the following meaning: YY year (00 – 99) MM month (01 - 12) DD day (01 - 31) HH hour (00 - 23) MM minute (00 - 59) Z The "Z" character indicates Greenwich Mean Time (GMT).	Input/Output
TIMELEN	Length of the storage area available for the *SigningTime*.	Input

GET-NEXT-SIGNER-CERT

Using this method the current signer's next signer certificate will be extracted.

Syntax	Cobol	
	MOVE GET-NEXT-SIGNER-CERT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, CERT, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_NEXT_SIGNER_CERT,CTX,CERT,RC),VL	
Return code	0 if no more certificates are available, else the length of the certificate.	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
CERT	Storage address of the extracted signer certificate.	Output

VERIFY-SIGNER

Using this method the *Signed-data* object will be checked for validity regarding a specific signer. The required SIGNERINFO strucuture has to be previously extracted calling one of the *GET-FIRST-SIGNER* and *GET-NEXT-SIGNER* methods.

Syntax	Cobol	
	MOVE VERIFY-SIGNER TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, SIGNER, CHECKCERT, RC.**	
	Assembler	
	CALL XPSCRYPT,(VERIFY_SIGNER,CTX,SIGNER,CHECKCERT,RC),VL	
Returncode (RC)	0 if signer can be verified, else error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
SIGNER	Storage address of the SIGNERINFO structure about to verify.	Input
CHECKCERT	If set to 0 the certificate paths of the signer and the trusted signers won't be checked. If set to 1 the path of certificates will be checked up to the root. In this case the issuer's certificate has to be loaded in advance using the ADD-TRUSTED-SIGNER method.	Input

VERIFY-ALL-SIGNER

All signers of the Signed-data object will be checked for validity.

Syntax	Cobol	
	MOVE VERIFY-ALL-SIGNER TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, CHECKCERT, RC.**	
	Assembler	
	CALL XPSCRYPT,(VERIFY_ALL_SIGNER,CTX,CHECKCERT,RC),VL	
Return code	0 if all signers can be verified, else error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input

CHECKCERT	If set to 0 the certificate paths of the signer and the trusted signers won't be checked. If set to 1 the path of certificates will be checked up to the root. In this case the issuer's certificate has to be loaded in advance using the ADD-TRUSTED-SIGNER method.	Input

GET-FIRST-PKCS7-DATA

The PKCS#7 objects data will be extracted. This method is available for all supported PKCS#7 types.

Syntax	**Cobol**	
	MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_FIRST_PKS7_DATA,CTX,DATA,RC),VL	
Return code	Length of data. If 0, no data is available, else error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
DATA	Storage address of the extracted data.	Output

GET-NEXT-PKCS7-DATA

The next data from the PKCS#7 object will be extracted. This method is available for all supported PKCS#7 types.

Syntax	**Cobol**	
	MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, DATA, RC.**	
	Assembler	
	CALL XPSCRYPT,(GET_NEXT_PKCS7_DATA,CTX,DATA,RC),VL	
Return code	Length of data. If 0, no data is available, else error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
DATA	Storage address of the extracted data.	Output

CREATE-OBJECT

This method finalizes the creation of a PKCS#7 object.

Syntax	**Cobol**
	MOVE CREATE-OBJECT TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, OBJECT, RC.**
	Assembler
	CALL XPSCRYPT,(CREATE_OBJECT,CTX,OBJECT,RC),VL

Return ode	Length of the PKCS#7 object or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input
OBJECT	Storage address of the created PKCS#7 object.	Output

CLEANUP-PKCS7

Deallocate storage areas previously reserved by diverse PKCS#7 methods.

Syntax	Cobol	
	MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(CLEANUP_PKCS7,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#7 context.	Input

COBOL example (create PKCS7Data):

```
*-----------------------------------------------------------------*
*      CREATE PKCS-7 DATA OBJECT                                  *
*-----------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     PK7WR1C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7DA".
 01  DATA1           PIC X(32)      VALUE "XPS Software GmbH, Haar
-                                         "/Muenchen".
 01  DATA2           PIC X(21)      VALUE "Muenchener Strasse 17".
 01  PKCS7-CTX       POINTER.
 01  ADDR-PKCS7-OBJ  POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  OPTION          PIC X.
 77  OBJECT-LENGTH   PIC 9(8)   COMP  VALUE ZEROES.
 77  DATALEN1        PIC 9(8)   COMP  VALUE 32.
 77  DATALEN2        PIC 9(8)   COMP  VALUE 21.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
*
 COPY XPSCLCOB.
*****************************************************************
**            PROCEDURE DIVISION                               **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* CREATE PKCS-7 DATA-OBJECT                                    *
* ------------------------------------------------------------ *
     MOVE HEADER-INCLUDED TO OPTION.
     MOVE CREATE-PKCS7-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-PKCS7-DATA: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* ADD DATA TO PKCS7 DATA-OBJECT                                *
* ------------------------------------------------------------ *
```

```
     MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, DATA1, DATALEN1, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, DATA2, DATALEN2, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CREATE PKCS-7 DATA-OBJECT                                    *
* ------------------------------------------------------------ *
     MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, ADDR-PKCS7-OBJ RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
        GOBACK.
     MOVE RC TO OBJECT-LENGTH.
* ------------------------------------------------------------ *
* WRITE PKCS-7 DATA-OBJECT TO MACLIB                           *
* ------------------------------------------------------------ *
     MOVE WRITE-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PKCS7-OBJ, OBJECT-LENGTH, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR PUT-FILE: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CLEANUP PKCS-7 CONTEXT                                       *
* ------------------------------------------------------------ *
     MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PKCS7-CTX, RC.
     STOP RUN.
 ENDRUN.
```

COBOL example (read PKCS7Data):

```
*------------------------------------------------------------*
*      TEST CHECK PKCS-7 DATA OBJECT                         *
*------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     PK7RD1C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7DA".
 01  HEADER1         PIC X(20)      VALUE "DATA:".
 01  PKCS7-CTX       POINTER.
 01  ADDR-PKCS7      POINTER.
 01  ADDR-DATA       POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  PKCS7LEN        PIC 9(8)   COMP  VALUE ZEROES.
 77  DUMPLEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
*
 COPY XPSCLCOB.
 01  PKCS7-DATA      PIC X(1).
************************************************************
**            PROCEDURE DIVISION                          **
************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* READ PKCS-7 DATA OBJECT "PK7DA" FROM MACLIB                  *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
     IF RC < 0
        MOVE RC TO RCC
```

```
       DISPLAY "FILE 'PK7DA' NOT FOUND  RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* IMPORT PKCS-7 DATA OBJECT                                    *
* ------------------------------------------------------------ *
    MOVE IMPORT-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
       PKCS7-CTX, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR IMPORT-PKCS-DATA: RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* GET ALL DATA                                                 *
* ------------------------------------------------------------ *
    MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, ADDR-DATA, RC.
    PERFORM UNTIL RC <= ZEROES
      PERFORM GET-DATA
    END-PERFORM.
* ------------------------------------------------------------ *
* CLEANUP PKCS-7 CONTEXT                                       *
* ------------------------------------------------------------ *
    MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, PKCS7-CTX, RC.
* ------------------------------------------------------------ *
* RELEASE FILE-STORAGE "PK7DA"                                 *
* ------------------------------------------------------------ *
    MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        ADDR-PKCS7, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR RELEASE-FILE: RC = " RCC
       GOBACK.
    STOP RUN.
*
 GET-DATA SECTION.
    SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER1, PKCS7-DATA, DUMPLEN, RC.
    MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, ADDR-DATA, RC.
 GET-DATA-END.
    EXIT.
 ENDRUN.
```

COBOL example (create SignedData):

```
*-------------------------------------------------------------*
*      CREATE PKCS-7 SIGNED DATA OBJECT                       *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
    PK7WR2C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7SD".
 01  PKCS12-FILE     PIC X(8)       VALUE "XPSUSERP".
 01  PWD             PIC X(8)       VALUE "xpsuser1".
 01  DATA1           PIC X(32)      VALUE "XPS Software GmbH, Haar
-                                         "/Muenchen".
 01  DATA2           PIC X(21)      VALUE "Muenchener Strasse 17".
 01  DATA3           PIC X(10)      VALUE "85540 Haar".
 01  ADDR-PKCS12     POINTER.
 01  PKCS7-CTX       POINTER.
 01  PEM-CTX         POINTER.
 01  ADDR-PKCS7-OBJ  POINTER.
 01  ADDR-PEM-OBJ    POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  OPTION          PIC X.
 77  PKCS12-LENGTH   PIC 9(8)   COMP  VALUE ZEROES.
 77  PWDLEN          PIC 9(8)   COMP  VALUE 8.
 77  OBJECT-LENGTH   PIC 9(8)   COMP  VALUE ZEROES.
 77  PEM-LENGTH      PIC 9(8)   COMP  VALUE ZEROES.
 77  DATALEN1        PIC 9(8)   COMP  VALUE 32.
```

```
 77  DATALEN2        PIC 9(8)   COMP  VALUE 21.
 77  DATALEN3        PIC 9(8)   COMP  VALUE 10.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
****************************************************************
**             PROCEDURE DIVISION                             **
****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* READ PKCS-12 FILE "XPSUSERP" FROM MACLIB                     *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS12-FILE, ADDR-PKCS12, PKCS12-LENGTH, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'XPSUSERP' NOT FOUND  RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CONVERT PASSWORD/DATA FROM EBCDIC TO ASCII                   *
* ------------------------------------------------------------ *
     MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, DATA1, DATALEN1, RC.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, DATA2, DATALEN2, RC.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, DATA3, DATALEN3, RC.
* ------------------------------------------------------------ *
* CREATE PKCS-7 SIGNED-DATA-OBJECT                             *
* ------------------------------------------------------------ *
     MOVE DATA-CERT-HEADER TO OPTION.
     MOVE CREATE-SIGNED-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-SIGNED-DATA: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* ADD DATA TO PKCS7 SIGNED-DATA-OBJECT                         *
* ------------------------------------------------------------ *
     MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, DATA1, DATALEN1, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, DATA2, DATALEN2, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, DATA3, DATALEN3, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* ADD SIGNER TO PKCS-7 SIGNED-DATA-OBJECT                      *
* ------------------------------------------------------------ *
     MOVE ADD-SIGNER TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, PKCS7-CTX,
         ADDR-PKCS12, PKCS12-LENGTH, PWD, PWDLEN, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ADD-SIGNER: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CREATE PKCS-7 SIGNED-DATA-OBJECT                             *
* ------------------------------------------------------------ *
     MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, ADDR-PKCS7-OBJ RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
        GOBACK.
     MOVE RC TO OBJECT-LENGTH.
* ------------------------------------------------------------ *
* CONVERT ASN1-FORMAT TO PEM-FORMAT                            *
* ------------------------------------------------------------ *
```

```
     MOVE ASN-2-PEM TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7-OBJ,
        OBJECT-LENGTH, PKCS7-FILE, ADDR-PEM-OBJ, PEM-CTX, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ASN-2-PEM: RC = " RCC
        GOBACK.
     MOVE RC TO PEM-LENGTH.
* ------------------------------------------------------------ *
* WRITE PKCS-7 SIGNED-DATA-OBJECT TO MACLIB                    *
* ------------------------------------------------------------ *
     MOVE WRITE-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PEM-OBJ, PEM-LENGTH, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR PUT-FILE: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CLEANUP PKCS-7 CONTEXT                                       *
* ------------------------------------------------------------ *
     MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PKCS7-CTX, RC.
     MOVE CLEANUP-PEM TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PEM-CTX, RC.
* ------------------------------------------------------------ *
* RELEASE FILE-STORAGES                                        *
* ------------------------------------------------------------ *
     MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         ADDR-PKCS12, RC.
     STOP RUN.
 ENDRUN.
```

COBOL example (read SignedData):

```
*-------------------------------------------------------------*
*      TEST CHECK PKCS-7 SIGNED-DATA OBJECT                   *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     PK7RD2C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7SD".
 01  CERT-FILE       PIC X(8)       VALUE "XPSTESTC".
 01  HEADER1         PIC X(20)      VALUE "SIGNER-CERTIFICATE:".
 01  HEADER2         PIC X(20)      VALUE "DATA:".
 01  PKCS7-CTX       POINTER.
 01  CERT-CTX        POINTER.
 01  ADDR-PKCS7      POINTER.
 01  ADDR-CERT       POINTER.
 01  ADDR-DATA       POINTER.
 01  ADDR-SIGNER     POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DUMPLEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  PKCS7LEN        PIC 9(8)   COMP  VALUE ZEROES.
 77  CERTLEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  BOOLEAN-TRUE    PIC 9(8)   COMP  VALUE 1.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
 01  PKCS7-DATA      PIC X(1).
 01  SIGNER-CERT     PIC X(1).
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* READ PKCS-7 SIGNED DATA OBJECT "PK7SD" FROM MACLIB           *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
     IF RC < 0
        MOVE RC TO RCC
```

```
           DISPLAY "FILE 'PK7SD' NOT FOUND  RC = " RCC
           GOBACK.
      * ------------------------------------------------------------ *
      * READ TRUSTED-SIGNER CERTIFICATE "XPSTESTC"                   *
      * ------------------------------------------------------------ *
          MOVE READ-FILE TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              DDNAME, CERT-FILE, ADDR-CERT, CERTLEN, RC.
          IF RC < 0
             MOVE RC TO RCC
             DISPLAY "FILE 'PK7SD' NOT FOUND  RC = " RCC
             GOBACK.
      * ------------------------------------------------------------ *
      * IMPORT SIGNED DATA OBJECT                                    *
      * ------------------------------------------------------------ *
          MOVE IMPORT-SIGNED-DATA TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
             PKCS7-CTX, RC.
          IF RC < 0
             MOVE RC TO RCC
             DISPLAY "ERROR IMPORT-SIGNED-DATA: RC = " RCC
             GOBACK.
      * ------------------------------------------------------------ *
      * ADD TRUSTED SIGNER CERTIFICATE                               *
      * ------------------------------------------------------------ *
          MOVE ADD-TRUSTED-SIGNER TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              PKCS7-CTX, ADDR-CERT, CERTLEN, RC.
          IF RC < 0
             MOVE RC TO RCC
             DISPLAY "ERROR ADD-TRUSTED-SIGNER: RC = " RCC
             GOBACK.
      * ------------------------------------------------------------ *
      * GET ALL SIGNERS                                              *
      * ------------------------------------------------------------ *
          MOVE GET-FIRST-SIGNER TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              PKCS7-CTX, ADDR-SIGNER, RC.
          IF RC < 0
             MOVE RC TO RCC
             DISPLAY "ERROR GET-FIRST-SIGNER: RC = " RCC
             GOBACK.
          PERFORM UNTIL RC < ZEROES
            PERFORM GET-SIGNERS
          END-PERFORM.
      * ------------------------------------------------------------ *
      * VERIFY SIGNER                                                *
      * ------------------------------------------------------------ *
          MOVE VERIFY-SIGNER TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              PKCS7-CTX, ADDR-SIGNER, BOOLEAN-TRUE, RC.
          IF RC < 0
             MOVE RC TO RCC
             DISPLAY "ERROR VERIFY-SIGNER: RC = " RCC
             GOBACK.
      * ------------------------------------------------------------ *
      * GET ALL DATA                                                 *
      * ------------------------------------------------------------ *
          MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              PKCS7-CTX, ADDR-DATA, RC.
          PERFORM UNTIL RC <= ZEROES
            PERFORM GET-DATA
          END-PERFORM.
      * ------------------------------------------------------------ *
      * CLEANUP PKCS-7 CONTEXT                                       *
      * ------------------------------------------------------------ *
          MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT'
              USING CRYPT-FUNCTION, PKCS7-CTX, RC.
      * ------------------------------------------------------------ *
      * RELEASE FILE-STORAGES                                        *
      * ------------------------------------------------------------ *
          MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              ADDR-PKCS7, RC.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              ADDR-CERT, RC.
          STOP RUN.
      * ------------------------------------------------------------ *
      * PERFORM ROUTINES                                             *
      * ------------------------------------------------------------ *
       GET-SIGNERS SECTION.
          SET ADDRESS OF SIGNINFO TO ADDR-SIGNER.
          SET ADDRESS OF SIGNER-CERT TO S-CERT.
          MOVE S-LCERT TO DUMPLEN.
          MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
          CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
              HEADER1, SIGNER-CERT, DUMPLEN, RC.
          MOVE GET-NEXT-SIGNER TO CRYPT-FUNCTION.
```

```
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, ADDR-SIGNER, RC.
 GET-SIGNERS-END.
    EXIT.
*
 GET-DATA SECTION.
    SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER2, PKCS7-DATA, DUMPLEN, RC.
    MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, ADDR-DATA, RC.
 GET-DATA-END.
    EXIT.
 ENDRUN.
```

COBOL example (create EnvelopedData):

```
*-------------------------------------------------------------*
*      CREATE PKCS-7 ENVELOPED DATA OBJECT                    *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
    PK7WR3C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7EV".
 01  CERT-FILE       PIC X(8)       VALUE "XPSUSERC".
 01  DATA1           PIC X(32)      VALUE "XPS Software GmbH, Haar
-                                         "/Muenchen".
 01  ADDR-CERT       POINTER.
 01  PKCS7-CTX       POINTER.
 01  PEM-CTX         POINTER.
 01  ADDR-PKCS7-OBJ  POINTER.
 01  ADDR-PEM-OBJ    POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  OPTION          PIC X.
 77  CRYPT-TYPE      PIC X.
 77  CERT-LENGTH     PIC 9(8)   COMP  VALUE ZEROES.
 77  OBJECT-LENGTH   PIC 9(8)   COMP  VALUE ZEROES.
 77  PEM-LENGTH      PIC 9(8)   COMP  VALUE ZEROES.
 77  DATALEN1        PIC 9(8)   COMP  VALUE 32.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* READ RECIPIENT CERTIFICATE-FILE 'XPSUSERC' FROM MACLIB       *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, CERT-FILE, ADDR-CERT, CERT-LENGTH, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'XPSUSERC' NOT FOUND  RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CONVERT DATA FROM EBCDIC TO ASCII                            *
* ------------------------------------------------------------ *
     MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, DATA1, DATALEN1, RC.
* ------------------------------------------------------------ *
* CREATE PKCS-7 ENVELOPED-DATA-OBJECT                          *
* ------------------------------------------------------------ *
     MOVE HEADER-INCLUDED TO OPTION.
     MOVE DESEDE3CBC      TO CRYPT-TYPE.
     MOVE CREATE-ENVELOPED-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX,
        CRYPT-TYPE, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-ENVELOPED-DATA: RC = " RCC
```

```
       GOBACK.
* ------------------------------------------------------------ *
* ADD DATA TO PKCS7 ENVELOPED-DATA-OBJECT                      *
* ------------------------------------------------------------ *
    MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, DATA1, DATALEN1, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* ADD RECIPIENT TO PKCS-7 ENVELOPED-DATA-OBJECT                *
* ------------------------------------------------------------ *
    MOVE ADD-RECIPIENT TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION, PKCS7-CTX,
        ADDR-CERT, CERT-LENGTH, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR ADD-RECIPIENT: RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* CREATE PKCS-7 ENVELOPED-DATA-OBJECT                          *
* ------------------------------------------------------------ *
    MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, ADDR-PKCS7-OBJ RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
       GOBACK.
    MOVE RC TO OBJECT-LENGTH.
* ------------------------------------------------------------ *
* CONVERT ASN1-FORMAT TO PEM-FORMAT                            *
* ------------------------------------------------------------ *
    MOVE ASN-2-PEM TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7-OBJ,
       OBJECT-LENGTH, PKCS7-FILE, ADDR-PEM-OBJ, PEM-CTX, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR ASN-2-PEM: RC = " RCC
       GOBACK.
    MOVE RC TO PEM-LENGTH.
* ------------------------------------------------------------ *
* WRITE PKCS-7 ENVELOPED-DATA-OBJECT TO MACLIB                 *
* ------------------------------------------------------------ *
    MOVE WRITE-FILE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        DDNAME, PKCS7-FILE, ADDR-PEM-OBJ, PEM-LENGTH, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR PUT-FILE: RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* CLEANUP PKCS-7 CONTEXT                                       *
* ------------------------------------------------------------ *
    MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, PKCS7-CTX, RC.
    MOVE CLEANUP-PEM TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, PEM-CTX, RC.
* ------------------------------------------------------------ *
* RELEASE FILE-STORAGES                                        *
* ------------------------------------------------------------ *
    MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        ADDR-CERT, RC.
    STOP RUN.
 ENDRUN.
```

COBOL example (read EnvelopedData):

```
*-------------------------------------------------------------*
*      TEST CHECK PKCS-7 ENVELOPED-DATA OBJECT                *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
    PK7RD3C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7EV".
 01  PKCS12-FILE     PIC X(8)       VALUE "XPSUSERP".
 01  PWD             PIC X(8)       VALUE "xpsuser1".
```

```
 01  HEADER1         PIC X(20)      VALUE "DATA:".
 01  PKCS7-CTX       POINTER.
 01  ADDR-PKCS7      POINTER.
 01  ADDR-PKCS12     POINTER.
 01  ADDR-DATA       POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DUMPLEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  PKCS7LEN        PIC 9(8)   COMP  VALUE ZEROES.
 77  PKCS12LEN       PIC 9(8)   COMP  VALUE ZEROES.
 77  PWDLEN          PIC 9(8)   COMP  VALUE 8.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
 01  PKCS7-DATA      PIC X(1).
*****************************************************************
**            PROCEDURE DIVISION                               **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------- *
* CONVERT PASSWORD FROM EBCDIC TO ASCII  (PASSWORD LOWER CASE|) *
* ------------------------------------------------------------- *
     MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
* ------------------------------------------------------------- *
* READ PKCS-7 ENVELOPED DATA OBJECT "PK7EV" FROM MACLIB         *
* ------------------------------------------------------------- *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'PK7EV' NOT FOUND  RC = " RCC
        GOBACK.
* ------------------------------------------------------------- *
* READ RECIPIENT PKCS12-FILE "XPSUSERP" FROM MACLIB             *
* ------------------------------------------------------------- *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS12-FILE, ADDR-PKCS12, PKCS12LEN, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'XPSUSERP' NOT FOUND  RC = " RCC
        GOBACK.
* ------------------------------------------------------------- *
* IMPORT ENVELOPED DATA OBJECT                                  *
* ------------------------------------------------------------- *
     MOVE IMPORT-ENVELOPED-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
        ADDR-PKCS12, PKCS12LEN, PWD, PWDLEN, PKCS7-CTX, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR IMPORT-ENVELOPED-DATA: RC = " RCC
        GOBACK.
* ------------------------------------------------------------- *
* GET ALL DATA                                                  *
* ------------------------------------------------------------- *
     MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, ADDR-DATA, RC.
     PERFORM UNTIL RC <= ZEROES
       PERFORM GET-DATA
     END-PERFORM.
* ------------------------------------------------------------- *
* CLEANUP PKCS-7 CONTEXT                                        *
* ------------------------------------------------------------- *
     MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PKCS7-CTX, RC.
* ------------------------------------------------------------- *
* RELEASE FILE-STORAGES                                         *
* ------------------------------------------------------------- *
     MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         ADDR-PKCS7, RC.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         ADDR-PKCS12, RC.
     STOP RUN.
*
* ------------------------------------------------------------- *
* PERFORM ROUTINES                                              *
* ------------------------------------------------------------- *
 GET-DATA SECTION.
     SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
```

```
    MOVE RC TO DUMPLEN.
    MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        HEADER1, PKCS7-DATA, DUMPLEN, RC.
    MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, ADDR-DATA, RC.
GET-DATA-END.
    EXIT.
ENDRUN.
```

COBOL example (create EncryptedData):

```
*-------------------------------------------------------------*
*      CREATE PKCS-7 ENCRYPTED DATA OBJECT                    *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
    PK7WR4C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7EC".
 01  DATA1           PIC X(32)      VALUE "XPS Software GmbH, Haar
-                                         "/Muenchen".
 01  PWD             PIC X(12)      VALUE "testpassword".
 01  PKCS7-CTX       POINTER.
 01  PEM-CTX         POINTER.
 01  ADDR-PKCS7-OBJ  POINTER.
 01  ADDR-PEM-OBJ    POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  OPTION          PIC X.
 77  CRYPT-TYPE      PIC X.
 77  OBJECT-LENGTH   PIC 9(8)   COMP  VALUE ZEROES.
 77  PEM-LENGTH      PIC 9(8)   COMP  VALUE ZEROES.
 77  DATALEN1        PIC 9(8)   COMP  VALUE 32.
 77  PWDLEN          PIC 9(8)   COMP  VALUE 12.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* CONVERT PWD/DATA FROM EBCDIC TO ASCII                        *
* ------------------------------------------------------------ *
    MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
    CALL 'XPSCRYPT'
        USING CRYPT-FUNCTION, DATA1, DATALEN1, RC.
    CALL 'XPSCRYPT'
* ------------------------------------------------------------ *
* CREATE PKCS-7 ENCRYPTED-DATA-OBJECT                          *
* ------------------------------------------------------------ *
    MOVE HEADER-INCLUDED TO OPTION.
    MOVE PBE3DES-3KEY    TO CRYPT-TYPE.
    MOVE CREATE-ENCRYPTED-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OPTION, PKCS7-CTX,
       CRYPT-TYPE, PWD, PWDLEN, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR CREATE-ENCRYPTED-DATA: RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* ADD DATA TO PKCS7 ENCRYPTED-DATA-OBJECT                      *
* ------------------------------------------------------------ *
    MOVE ADD-PKCS7-DATA TO CRYPT-FUNCTION.
    CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        PKCS7-CTX, DATA1, DATALEN1, RC.
    IF RC < 0
       MOVE RC TO RCC
       DISPLAY "ERROR ADD-PKCS7-DATA: RC = " RCC
       GOBACK.
* ------------------------------------------------------------ *
* CREATE PKCS-7 ENCRYPTED-DATA-OBJECT                          *
* ------------------------------------------------------------ *
    MOVE CREATE-OBJECT TO CRYPT-FUNCTION.
```

```
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, ADDR-PKCS7-OBJ RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR CREATE-OBJECT: RC = " RCC
        GOBACK.
     MOVE RC TO OBJECT-LENGTH.
* ------------------------------------------------------------ *
* CONVERT ASN1-FORMAT TO PEM-FORMAT                            *
* ------------------------------------------------------------ *
     MOVE ASN-2-PEM TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7-OBJ,
        OBJECT-LENGTH, PKCS7-FILE, ADDR-PEM-OBJ, PEM-CTX, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR ASN-2-PEM: RC = " RCC
        GOBACK.
     MOVE RC TO PEM-LENGTH.
* ------------------------------------------------------------ *
* WRITE PKCS-7 ENCRYPTED-DATA-OBJECT TO MACLIB                 *
* ------------------------------------------------------------ *
     MOVE WRITE-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PEM-OBJ, PEM-LENGTH, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR PUT-FILE: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* CLEANUP PKCS-7 CONTEXT                                       *
* ------------------------------------------------------------ *
     MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PKCS7-CTX, RC.
     MOVE CLEANUP-PEM TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PEM-CTX, RC.
     STOP RUN.
 ENDRUN.
```

COBOL example (read EncryptedData):

```
*-------------------------------------------------------------*
*      TEST CHECK PKCS-7 ENCRYPTED-DATA OBJECT                *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     PK7RD4C.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DDNAME          PIC X(8)       VALUE "XPSDATA".
 01  PKCS7-FILE      PIC X(8)       VALUE "PK7EC".
 01  PWD             PIC X(12)      VALUE "testpassword".
 01  HEADER1         PIC X(20)      VALUE "DATA:".
 01  PKCS7-CTX       POINTER.
 01  ADDR-PKCS7      POINTER.
 01  ADDR-DATA       POINTER.
*
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DUMPLEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  PKCS7LEN        PIC 9(8)   COMP  VALUE ZEROES.
 77  PWDLEN          PIC 9(8)   COMP  VALUE 12.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
 01  PKCS7-DATA      PIC X(1).
*************************************************************
**             PROCEDURE DIVISION                          **
*************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------ *
* CONVERT PASSWORD FROM EBCDIC TO ASCII  (PASSWORD LOWER CASE|) *
* ------------------------------------------------------------ *
     MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PWD, PWDLEN, RC.
* ------------------------------------------------------------ *
* READ PKCS-7 ENCRYPTED DATA OBJECT "PK7EC" FROM MACLIB        *
* ------------------------------------------------------------ *
     MOVE READ-FILE TO CRYPT-FUNCTION.
```

```
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS7-FILE, ADDR-PKCS7, PKCS7LEN, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "FILE 'PK7EC' NOT FOUND  RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* IMPORT ENCRYPTED DATA OBJECT                                 *
* ------------------------------------------------------------ *
     MOVE IMPORT-ENCRYPTED-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS7, PKCS7LEN,
        PWD, PWDLEN, PKCS7-CTX, RC.
*
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR IMPORT-ENCRYPTED-DATA: RC = " RCC
        GOBACK.
* ------------------------------------------------------------ *
* GET ALL DATA                                                 *
* ------------------------------------------------------------ *
     MOVE GET-FIRST-PKCS7-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, ADDR-DATA, RC.
     PERFORM UNTIL RC <= ZEROES
       PERFORM GET-DATA
     END-PERFORM.
* ------------------------------------------------------------ *
* CLEANUP PKCS-7 CONTEXT                                       *
* ------------------------------------------------------------ *
     MOVE CLEANUP-PKCS7 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PKCS7-CTX, RC.
* ------------------------------------------------------------ *
* RELEASE FILE-STORAGES                                        *
* ------------------------------------------------------------ *
     MOVE CLEANUP-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         ADDR-PKCS7, RC.
     STOP RUN.
*
* ------------------------------------------------------------ *
* PERFORM ROUTINES                                             *
* ------------------------------------------------------------ *
 GET-DATA SECTION.
     SET ADDRESS OF PKCS7-DATA TO ADDR-DATA.
     MOVE RC TO DUMPLEN.
     MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         HEADER1, PKCS7-DATA, DUMPLEN, RC.
     MOVE GET-NEXT-PKCS7-DATA TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS7-CTX, ADDR-DATA, RC.
 GET-DATA-END.
     EXIT.
 ENDRUN.
```

Chapter

9

PKCS#12 private key

Common information

PKCS#12 objects (*Personal-Information-Exchange-Syntax-Standard*) define syntax for keys and certificates exchange. PKCS#12 objects contain key-bags and certificate-bags. PKCS#12 is accounted standard for securely storing private keys and certificates. Internet browser programs such as Netscape (.p12) and Microsoft Internet Explorer (.pfx) support PKCS#12.

Methods

IMPORT-PKCS12

Reading a PKCS#12 object and checking its formal correctness.

Syntax	**Cobol**	

	`MOVE IMPORT-PKCS12 TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`APKCS12, PKCS12LEN, PWD, PWDLEN, CTX, RC.`	

	Assembler	
	`CALL XPSCRYPT,(IMPORT_PKCS12,APKCS12,PKCS12LEN,PWD,PWDLEN,CTX,` `RC),VL`	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
APKCS12	Storage address of the PKCS#12 object.	Input
PKCS12LEN	Length of the PKCS#12 object.	Input
PWD	Storage address of the password that has been used to encrypt the PKCS#12 object.	Input
PWDLEN	The length of the password.	Input
CTX	Storage address of the created PKCS#12 context. This object will be needed for subsequent processing.	Output

GET-PRIVATE-KEY

Extract the private key from the PKCS#12 object.

Syntax	Cobol	

	`MOVE GET-PRIVATE-KEY TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, PRIVKEY, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_PRIVATE_KEY,CTX,PRIVKEY,RC),VL`	
Return code	Length of the private key structure or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#12 context.	Input
PRIVKEY	Strorage address to be used to store the extracted private key. The description of the extracted strcuture can be found in the copy books XPSCLRSA (COBOL) and XPSCLASM (Assembler) respectively.	Output

GET-FIRST-CERT

Extract the first user certificate from the PKCS#12 object.

Syntax	Cobol	

	`MOVE GET-FIRST-CERT TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, CERT, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_FIRST_CERT,CTX,CERT,RC),VL`	
Return code	Length of the X.509 certificate or 0 if no certificate is available.	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#12 context.	Input
CERT	Storage address of the extracted X.509 certificate.	Output

GET-NEXT-CERT

Extract the next certificate from the PKCS#12 object. Besides the user's certificate a PKCS#12 object can contain all signer certificates up to the root certificate.

Syntax	Cobol	

	`MOVE GET-NEXT-CERT TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`CTX, CERT, RC.`	

	Assembler	
	`CALL XPSCRYPT,(GET_NEXT_CERT,CTX,CERT,RC),VL`	
Return code	Length of the X.509 certificate or 0 if no certificate is available.	
Parameter	**Description**	**Verwendung**
CTX	Storage address of the PKCS#12 context.	Input
CERT	Storage address of the extracted X.509 certificate.	Output

CLEANUP-PKCS12

Deallocate storage areas previously allocated by diverse PKCS#12 methods.

Syntax	**Cobol**	
	MOVE CLEANUP-PKCS12 TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(CLEANUP_PKCS12,CTX,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the PKCS#12 context.	Input

COBOL example:

```
*-----------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM: GET PRIVATE-KEY               *
*-----------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     PK12TSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  PASSWRD          PIC X(32)        VALUE "xpsuser1".
 01  PKCS12-FILE      PIC X(8)         VALUE "XPSUSERP".
 01  DDNAME           PIC X(8)         VALUE "XPSDATA".
 01  PKCS12-CTX       POINTER.
 01  ADDR-PKCS12-FILE POINTER.
 01  ADDR-CERTIFICATE POINTER.
 COPY XPSCLRSA.
*
 77  CRYPT-FUNCTION  PIC X.
 77  DATALEN         PIC 9(8)   COMP  VALUE ZEROES.
 77  PWDLEN          PIC 9(8)   COMP  VALUE 8.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
*
 COPY XPSCLCOB.
*****************************************************************
**             PROCEDURE DIVISION                              **
*****************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ------------------------------------------------------------- *
* READ PKCS-12 FILE "XPSUSERP" FROM MACLIB                      *
* ------------------------------------------------------------- *
     MOVE READ-FILE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         DDNAME, PKCS12-FILE, ADDR-PKCS12-FILE, DATALEN, RC.
     IF RC < 0
        DISPLAY "FILE 'XPSUSERP' NOT FOUND  RC = " RC
        GOBACK.
* ------------------------------------------------------------- *
* CONVERT PASSWORD FROM EBCDIC TO ASCII  (PASSWORD LOWER CASE|) *
* ------------------------------------------------------------- *
     MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PASSWRD, PWDLEN, RC.
* ------------------------------------------------------------- *
* IMPORT PKCS-12 FILE                                           *
* ------------------------------------------------------------- *
     MOVE IMPORT-PKCS12 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, ADDR-PKCS12-FILE,
        DATALEN, PASSWRD, PWDLEN, PKCS12-CTX, RC.
     IF RC < 0
        DISPLAY "ERROR IMPORT-FILE: RC = " RC
        GOBACK.
* ------------------------------------------------------------- *
* GET PRIVATE-KEY FROM PKCS-12 FILE                             *
```

```
* ------------------------------------------------------------ *
     MOVE GET-PRIVATE-KEY TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS12-CTX, RSA-PRIVATE-KEY, RC.
     IF RC < 0
        DISPLAY "ERROR GET-PRIVATE-KEY: RC = " RC
        GOBACK.
* ------------------------------------------------------------ *
* GET ALL CERTIFICATES FROM PKCS-12 FILE                       *
* ------------------------------------------------------------ *
     MOVE GET-FIRST-CERT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS12-CTX, ADDR-CERTIFICATE, RC.
     PERFORM UNTIL RC <= ZEROES
       PERFORM GET-CERTS
     END-PERFORM.
* ------------------------------------------------------------ *
* CLEANUP PKCS-12 CONTEXT                                      *
* ------------------------------------------------------------ *
     MOVE CLEANUP-PKCS12 TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, PKCS12-CTX, RC.
     STOP RUN.
*
 GET-CERTS SECTION.
     MOVE GET-NEXT-CERT TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
         PKCS12-CTX, ADDR-CERTIFICATE, RC.
 GET-CERTS-END.
     EXIT.
 ENDRUN.
```

Chapter

10

GZIP

Common information

CryptLib offers the possibility to compress and decompress data using *GZIP* and *GUNZIP*.

Methods

GZIP

Data will be compressed according to the gzip format.

Syntax	**Cobol**	
	MOVE GZIP TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **INPUT, INPUTLEN, AOUTPUT, OUTLEN, FILENAME, RC.**	
	Assembler	
	CALL XPSCRYPT,(GZIP,INPUT,INPUTLEN,AOUTPUT,OUTLEN,FILENAME,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
INPUT	Storage address of the data to be compressed.	Input
INPUTLEN	Length of the data about to compress.	Input
AOUTPUT	Address pointer to receive the storage address of the compressed data.	Output
OUTLEN	Storage address of a field about to receive the length of the compressed data.	Output
FILENAME	File name to be stored in the ZIP header. If given the name must be low value terminated (x'00').	Input

GUNZIP

Decompress data previously compressed according to the gzip format.

Syntax	**Cobol**
	MOVE GUNZIP TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **INPUT, INPUTLEN, AOUTPUT, OUTLEN, AFILENAME, RC.**
	Assembler

	CALL XPSCRYPT,(GUNZIP,INPUT,INPUTLEN,AOUTPUT,OUTLEN,AFILENAME, RC),VL	
Returncode (RC)	0 or error code (< 0).	
Parameter	**Description**	**Use**
INPUT	Storage address of the data to be decompressed.	Input
INPUTLEN	Length of the data about to decompress.	Input
AOUTPUT	Address pointer to receive the storage address of the decompressed data.	Output
OUTLEN	Storage address of a field about to receive the length of the decompressed data.	Output
FILENAME	Storage address to receive the file name stored in the ZIP header. If no file name is stored in the ZIP header this address is set to NULL otherwise the file name will be low value terminated (x'00').	Output

COBOL example:

```
*-------------------------------------------------------------*
*      XPS-CRYPTLIB SAMPLE PROGRAM ZIP                        *
*-------------------------------------------------------------*
 ID DIVISION.
 PROGRAM-ID.
     AESTSTC.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  XPS             PIC X(160)       VALUE "XPS Software GmbH
-                                           "Muenchener Str. 17
-                                           "85540 Haar/Muenchen
-                                           "Tel. 0049-89-456989-0
-                                           "
-                                           "Internet: www.xps.biz
-                                           "".
 01  FILENAME        PIC X(16)        VALUE "XPSHOME.TXT".
 01  HEADER1         PIC X(20)        VALUE "ZIP-DATA:".
 01  HEADER2         PIC X(20)        VALUE SPACES.
 01  ADDR-ZIPDATA    POINTER.
 01  ADDR-UNZIPDATA  POINTER.
 01  ADDR-FILENAME   POINTER          VALUE NULL.
*
 COPY XPSCLCTX.
*
 77  CRYPT-FUNCTION  PIC X.
 77  ZIPLEN          PIC 9(8)   COMP  VALUE ZEROES.
 77  UNZIPLEN        PIC 9(8)   COMP  VALUE ZEROES.
 77  XPSLEN          PIC 9(8)   COMP  VALUE 160.
 77  RC              PIC 9(8)   COMP  VALUE ZEROES.
 77  RCC             PIC 9(8)         VALUE ZEROES.
 77  NULL-PARM       PIC 9(8)   COMP  VALUE ZEROES.
*
 LINKAGE SECTION.
 COPY XPSCLCOB.
 01  OUT-DATA        PIC X(1).
 01  OUT-FILENAME    PIC X(12).
*************************************************************
**             PROCEDURE DIVISION                          **
*************************************************************
 PROCEDURE DIVISION.
 MAIN SECTION.
* ---------------------------------------------------------- *
* ZIP XPS                                                    *
* ---------------------------------------------------------- *
     MOVE GZIP TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION,
        XPS, XPSLEN, ADDR-ZIPDATA, ZIPLEN, FILENAME, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR GZIP: RC = " RCC
        GOBACK.
     SET ADDRESS OF OUT-DATA TO ADDR-ZIPDATA.
     MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, HEADER1, OUT-DATA, ZIPLEN, RC.
* ---------------------------------------------------------- *
* UNZIP XPS                                                  *
* ---------------------------------------------------------- *
```

```
     MOVE GUNZIP TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT' USING CRYPT-FUNCTION, OUT-DATA,
        ZIPLEN, ADDR-UNZIPDATA, UNZIPLEN, ADDR-FILENAME, RC.
     IF RC < 0
        MOVE RC TO RCC
        DISPLAY "ERROR GUNZIP: RC = " RCC
        GOBACK.
     SET ADDRESS OF OUT-DATA TO ADDR-UNZIPDATA.
     IF ADDR-FILENAME NOT = NULL
       SET ADDRESS OF OUT-FILENAME TO ADDR-FILENAME
       MOVE OUT-FILENAME TO HEADER2
     END-IF.
     MOVE DUMP-STORAGE TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, HEADER2, OUT-DATA, UNZIPLEN, RC.
* ------------------------------------------------------------ *
* CLEANUP CONTEXT                                              *
* ------------------------------------------------------------ *
     MOVE CLEANUP-GZIP TO CRYPT-FUNCTION.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, ADDR-ZIPDATA, RC.
     CALL 'XPSCRYPT'
         USING CRYPT-FUNCTION, ADDR-UNZIPDATA, RC.
     STOP RUN.
 ENDRUN.
```

Kapitel

11

Additional methods

Common information

CryptLib offers the programmer a number of usefull methods for the development of cryptographic applications. Among these are methods for the conversion of ASN.1 objects from binary to US-ASCII and vice versa, methods to read and write files and methods for data conversion from EBCDIC to ASCII and vice versa.

Methods

ASN2PEM

BER/DER encoded ASN.1 objects are available in binary format. Because of the fact that some transmission protocols don't support transmission of binary data the need for translation into US-ASCII representation rises. This is carried out using the Base64 method as layed out in RFC 1521. Using this method a binary object can be translated into a Base64 object.

Syntax	**Cobol**	
	MOVE ASN2PEM TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **INPUT, INPUTLEN, SMIME, AOUTPUT, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(ASN2PEM,INPUT,INPUTLEN,SMIME,AOUTPUT,CTX,RC),VL	
Return code	Length of the Base64 object or error code (< 0).	
Parameter	**Description**	**Use**
INPUT	Storage address of the binary ASN.1 object.	Input
INPUTLEN	Length of the ASN.1 object.	Input

SMIME	If a S/MIME name is provided the following S/MIME header will be included in front of the PEM object:
	
	`Content-Disposition: attachment; filename="smime.p7m"`
	`Content-Type: application/x-pkcs7-mime; name="smime.p7m"`
	`Content-Transfer-Encoding: base64`
		Input

AOUTPUT	Storage address of the created Base64 object.	Output
CTX	Storage address of the created context. This will be required in order to deallocate any reserved working storage.	Output

PEM2ASN

Using this method a Base64 object can be re-translated into a binary object.

Syntax	Cobol	
	MOVE PEM2ASN TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **INPUT, INPUTLEN, AOUTPUT, CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(PEM2ASN,INPUT,INPUTLEN,AOUTPUT,CTX,RC),VL	
Return code	Length of the binary object or error code (< 0).	
Parameter	**Description**	**Use**
INPUT	Storage address of the Base64 object.	Input
INPUTLEN	Length of the Base64 object.	Input
AOUTPUT	Storage address of the created binary object.	Output
CTX	Storage address of the created context. This will be required in order to deallocate any reserved working storage.	Output

CLEANUP-PEM

Deallocation of the storage used by the Base64 routines.

Syntax	Cobol	
	MOVE CLEANUP-PEM TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **CTX, RC.**	
	Assembler	
	CALL XPSCRYPT,(CLEANUP_PEM,RC),VL	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
CTX	Storage address of the context.	Input

READ-FILE

Using this function a file can be read. DDNAME specifies the MACLIB (MVS) or the Library-Sublib (VSE) respectively where the required file is located. Under MVS the DD statement and under VSE the LIBDEF SEARCH statement has to be listed in Job Control.

Syntax	Cobol	
	MOVE READ-FILE TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **DDNAME, FILENAME, AFILE, FILELEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(READ_FILE,DDNAME,FILENAME,AFILE,FILELEN,RC),VL	
Retur ncode	Length of the file or error code (< 0).	
Parameter	**Description**	**Use**
DDNAME	Name of the Job Control DD statement (MVS) or Library-Sublib-Name (VSE) respectively.	Input

FILENAME	Name of the file to read.	Input
AFILE	Storage address of the read file.	Output
FILELEN	Storage address of a field used to return the length of the read file.	Output

WRITE-FILE

Using this function a file can be written. DDNAME specifies the MACLIB (MVS) or the Library-Sublib (VSE) respectively to use to store the file. Under MVS the DD statement and under VSE the LIBDEF SEARCH statement has to be listed in Job Control.

Syntax	**Cobol**	

	`MOVE WRITE-FILE TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`DDNAME, FILENAME, AFILE, FILELEN, RC.`	

	Assembler	
	`CALL XPSCRYPT,(WRITE_FILE,DDNAME,FILENAME,AFILE,FILELEN,RC),VL`	
Return code	0 or error code (< 0).	
Parameter	**Description**	**Use**
DDNAME	Name of the Job Control DD statement (MVS) or Library-Sublib-Name (VSE) respectively.	Input
FILENAME	Name of the file to create.	Input
AFILE	Storage address of the file data to write.	Input
FILELEN	Length of the file data to write.	Input

CLEANUP-FILE

Deallocation of the storage reserved for read file data.

Syntax	**Cobol**	

	`MOVE CLEANUP-FILE TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`AFILE, RC.`	

	Assembler	
	`CALL XPSCRYPT,(CLEANUP_FILE,AFILE,RC),VL`	
Return code	None.	
Parameter	**Description**	**Use**
AFILE	Address of the storage to deallocate.	Input

EBCDIC-TO-ASCII

Using this method data can be converted from EBCDIC to ASCII.

Syntax	**Cobol**

	`MOVE EBCDIC-TO-ASCII TO CRYPT-FUNCTION.`
	`CALL 'XPSCRYPT' USING CRYPT-FUNCTION,`
	`DATA, DATALEN, RC.`

	Assembler	
	CALL XPSCRYPT,(EBCDIC_TO_ASCII,DATA,DATALEN,RC),VL	
Return code	None.	
Parameter	**Description**	**Use**
DATA	Storage address of the EBCDIC data about to convert.	Input/Output
DATALEN	Length of the EBCDIC data.	Input

ASCII-TO-EBCDIC

Using this method data can be converted from ASCII to EBCDIC.

Syntax	**Cobol**	
	MOVE ASCII-TO-EBCDIC TO CRYPT-FUNCTION. **CALL 'XPSCRYPT' USING CRYPT-FUNCTION,** **DATA, DATALEN, RC.**	
	Assembler	
	CALL XPSCRYPT,(ASCII_TO_EBCDIC,DATA,DATALEN,RC),VL	
Return code	None.	
Parameter	**Description**	**Use**
DATA	Storage address of the ASCII data about to convert.	Input/Output
DATALEN	Length of the ASCII data.	Input

Chapter

12

Error codes

ERR_TOOMANYPARMS -001

Description: Too many parameters have been specified for a method call.

ERR_UNSUPPFUNC -002

Description: The selected function transmitted in the Crypt Function field is invalid.

ERR_NOSTORAGE -003

Description: No more virtual storage available.

ERR_INVPARMS -004

Description: Too few parameters have been specified for a method call.

ERR_ALGORITHM -100

Description: The algorithm transmitted to the INIT-CTX method is not supported. Supported algorithms are AES, DES, RC2, RC4, Blowfish and RSA.

ERR_KEYLENGTH -101

Description: The key length transmitted to the INIT-CTX method is not supported.

ERR_MODE -102

Description: The mode transmitted to the INIT-CTX method is not supported. Supported modes are ECB and CBC for symmetrical encryption, PUBLIC and PRIVATE for asymmetrical encryption.

ERR_BUILDKEY **-103**

Description: An error occurred when the INIT-CTX method tried to initialize the encryption algorithm.

ERR_BUILDIV **-104**

Description: An error occurred when the INIT-CTX method tried to build the initialization vector.

ERR_CTX **-105**

Description: The transmitted context is invalid or not properly initialized.

ERR_OUTLEN **-106**

Description: The storage area provided for the method is too small.

ERR_LICENSE **-150**

Description: No valid license file is available. Please contact XPS.

ERR_CONTENTENC **-200**

Description: An error occurred when converting Base64 data to PEM.

ERR_DATA **-201**

Description: Data transmitted to a RSA method is invalid.

ERR_DIGALGO **-202**

Description: The hash type selected for the RSA method is not supported.

ERR_ENCODING **-203**

Description: An error occurred when converting PEM data to Base64.

ERR_RSAKEY **-204**

Description: The RSA key transmitted to the method is invalid.

ERR_RSALENGTH **-205**

Description: The length of the data transmitted to the RSA method is invalid.

ERR_MODULUS **-206**

Description: The modulus of the transmitted RSA key is incorrect.

ERR_RANDOM **-207**

Description: A random structure couldn't be initialized.

ERR_PRIVKEY **-208**

Description: The private RSA key transmitted to the method is invalid.

ERR_PUBKEY **-209**

Description: The public RSA key transmitted to the method is invalid.

ERR_SIGNATURE **-210**

Description: The signature transmitted for verification doesn't match the original.

ERR_ENCRALGO **-211**

Description: The encryption algorithm specified for the RSA method is unknown.

ERR_CERTPARM **-300**

Description: An invalid parameter has been transmitted to the IMPORT-CERTIFICATE method.

ERR_CERTIMPORT **-301**

Description: An unsupported X.509 certificate has been transmitted to the IMPORT-CERTIFICATE method.

ERR_CERTLENGTH **-302**

Description: The storage area provided for the extraction of a certificate is too small.

ERR_CERTALGO **-303**

Description: The only supported encryption algorithm for X.509 certificates is RSA.

ERR_CERTHASH **-304**

Description: The only supported hash methods for X.509 certificates are MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160.

ERR_CERTSTART **-305**

Description: Within the validation process of a X.509 certificate a future certificate begin date has been detected. The signature of the certificate is correct.

ERR_CERTEND **-306**

Description: Within the validation process of a X.509 certificate an elapsed certificate end data has been detected. The signature of the certificate is correct.

ERR_CERTOID **-307**

Description: The certificate extension searched with the GET-EXTENSION-BY-OID method is inexistent.

ERR_ASN1 **-400**

Description: The object currently being imported has a faulty or unsupported ASN.1 structure.

ERR_ASN1TABLE **-401**

Description: The imported ASN.1 object is not compatible with the called method.

ERR_HASHOID **-402**

Description: The imported PKCS object uses an unsupported hash type.

ERR_CONTENTINF **-403**

Description: The imported PKCS object contains unsupported content information.

ERR_HMAC **-404**

Description: The HMAC calculated for the imported PKCS#12 object is invalid. Possibly the transmitted password is incorrect.

ERR_AUTHSAFE **-405**

Description: The imported PKCS#12 object contains an unsupported authenticated safe.

ERR_PBETYPE **-406**

Description: The imported PKCS#12 object contains an unsupported PBE type (PBE = password based encryption). CryptLib supports the following algorithms: pbeWithSHAAnd128BitRC4, pbeWithSHAAnd40BitRc4, pbeWithSHAAnd3KeyTripleDES-CBC, pbeWithSHAAnd2KeyTripleDES-CBC, pbeWithSHAAnd128BitRC2-CBC and pbeWithSHAAnd40BitRC2-CBC.

ERR_CERTBAG **-407**

Description: The imported PKCS#12 object contains an unsupported certificate-bag.

ERR_CERTBAGTYPE **-408**

Description: The certificate-bag found in the imported PKCS#12 object contains an unspported certificate format. CryptLib supports the following formats: x509Certificate and sdsiCertificate.

ERR_NOAUTHSAFE **-409**

Description: The imported PKCS#12 object doesn't contain an authenticated safe.

ERR_SAFEBAG **-410**

Description: The imported PKCS#12 object doesn't contain a safe bag type 'pkcs8-shroudedKeybag'.

ERR_PRIVKEY **-411**

Description: The imported PKCS#12 object doesn't contain a private key.

ERR_RSAENC **-412**

Description: CryptLib only supports the RSA encryption algorithm for PKCS#12 objects.

ERR_NOKEYBAG **-413**

Description: The imported PKCS#12 object doesn't contain a key-bag.

ERR_HMACALGO **-414**

Description: The imported PKCS#12 object contains an unsupported HMAC algorithm. CryptLib supports the following algorithms: MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160..

ERR_ALGO **-415**

Description: The imported PKCS#7 object contains an unsupported hash type. CryptLib supports the following hash types: MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RipeMD160..

ERR_EALGO **-416**

Description: The imported PKCS#7 object contains an unsupported encryption algorithm. CryptLib only supports the RSA algorithm.

ERR_NOSIGNER -417

Description: No trusted signer could be found for the signer specified in the imported PKCS#7 object.

ERR_NOSIGNERCERT -418

Description: No certificate could be found for the signer specified in the imported PKCS#7 object.

ERR_MESSAGEDIG -419

Description: The Message Digest for the signer specified in the imported PKCS#7 signed-data object is invalid.

ERR_VERIFY -420

Description: Verification of the signature of the signer of the imported PKCS#7 signed-data object failed.

ERR_UNKNOWNSIGNER -421

Description: The signer transmitted to the method *VERIFY-SIGNER* isn't contained in the imported PKCS#7 signed-data object.

ERR_NODATA -422

Description: The imported PKCS#7 object doesn't contain data.

ERR_NOCERT -423

Description: The imported PKCS#7 object doesn't contain a certificate.

ERR_NORECIPIENT -424

Description: The imported PKCS#7 enveloped-data object doesn't contain a recipient.

ERR_P12NOTVALID -425

Description: The PKCS#12 object tansmitted to the *IMPORT-ENVELOPED-DATA* method is invalid.

ERR_INVOPT -426

Description: The option transmitted to the *CREATE…DATA* method is invalid.

ERR_NOTRUSTEDSIGN -427

Description: No trusted signer could be located for the signer transmitted to the *VERIFY-SIGNER* method.

ERR_MAXDATA **-428**

Description: Executing the ADD-PKCS7-DATA method the maximum acceptable data size has been exceeded.