Crosscert(DS) JSP Toolkit
Reference Manual

한국전자인증(주)

기술지원팀

프로젝트명	한국전자인증 툴킷-업체제공	프로젝트단계	진행
문서 번호		툴킷 버전	2004.8.27
문 서 명	CrossCert[DS] JSP Toolkit Reference Manual	최종작성일자	2004-8-27
문서 버전	1.3	작 성 자	기술지원팀

1. 개요 4

2. 환경설정 5

가) 라이브러리 패스 추가 5
나) 클래스 패스에 추가 5
다) CA 인증서와 라이센스 파일을 복사 5
라) 라이브러리 설치시 유의사항 5

3. 전자 서명 생성 클래스(CROSSCERT.SIGNER) 6

가) 전자 서명 생성 6
– PKCS#7 SignedData 6

4. 전자 서명 검증 클래스(CROSSCERT.VERIFIER) 8

나) 전자 서명 검증 8
– PKCS#7 SignedData 8

5. 암호화 클래스(CROSSCERT.ENCRYPT) 10

가) 비대칭키 암호화 (EncEnvelopedData) 10
– PKCS#7 EnvelopedData 10
나) 대칭키 암호화 (EncryptData) 11

6. 복호화 클래스(CROSSCERT.DECRYPT) 12

다) 비대칭키 복호화 (DecEnvelopedData) 12
– PKCS#7 EnvelopedData 12
라) 대칭키 복호화 (DecryptData) 14

7. 해쉬 클래스(CROSSCERT.HASH) 15

가) 해쉬함수 (GetHash) 15

8. 인증서 클래스(CROSSCERT.CERTIFICATE) 16

가) 인증서 검증 (ValidateCert) 16
나) 식별번호 검증 (VerifyVID) 18
다) 인증서 정보 추출 (ExtractCertInfo) 19
라) LDAP에서 인증서 가져오기 (GetCertFromLdap) 21

9. 개인키 추출 클래스 (CROSSCERT.PRIVATEKEY) 22

가) 개인키 추출 (DecryptPriKey) 22

10. BASE64 함수(CROSSCERT.BASE64) 24

가) base64 인코딩 (Encode) 24
나) base64 디코딩 (Decode) 25

1. 개요

Crosscert[DS] JSP 툴킷은 웹 기반의 응용 시스템에서 PKI 기반 라이브러리를 사용할 수 있도록 ATL/COM으로 작성된 ActiveX Control을 지원한다.

인증서 기반 전자 서명 및 암호화 모듈들에 대한 함수들을 설명한다. PKCS#7 관련 함수들은 PKI기반의 암호화 및 전자서명된 메시지의 표준 형식인 PKCS#7 Cryptographic Message Syntax에 맞는 결과들을 Base64 인코딩된 형식으로 변환하여 출력한다.

2. 환경설정

가) 라이브러리 패스 추가

: - 윈도우 : PATH
-. 리눅스 : LD_LIBRARY_PATH
-. 유닉스 : LD_LIBRARY_PATH
-. HP-UX : SHLIB_PATH

라이브러리 파일들을 /usr/local/CrossCertLIB에 복사한 후, 환경 파일에 다음을 추가한다.

export LD_LIBRARY_PATH=/usr/local/CrossCertLIB(리눅스,유닉스)
export SHLIB_PATH=/usr/local/CrossCertLIB(HP-UX)

나) 클래스 패스에 추가

: crosscert 폴더에 있는 class 파일들(Base64.class, Certificate.class, Decrypt.class, Encrypt.class, Hash.class, PrivateKey.class, Signer.class, Verifyer.class)을 classpath에 추가한다.

예) crosscert 디렉토리가 /usr/local/test/crosscert에 존재한다고 가정할 때 클래스패스는 /usr/local/test 까지 잡혀 있어야 한다.

다) CA 인증서와 라이센스 파일을 복사

: NPKI 폴더(윈도우는 C:₩Program Files₩NPKI, 유닉스는 /usr/local/NPKI)에 인증기관의 인증서 체인 파일(cpb)과 서버 툴킷 라이센스 파일인 DSToolkit32.lic을 복사한다.

라) 라이브러리 설치시 유의사항

- 모든 라이브러리와 NPKI디렉토리는 해당 서버에 FTP를 사용해서 올릴 때 전송 모드를 바이너리로 한다.

확인방법 : ldd DSCrossCert2.so
명령 수행시 참조하는 것들이 정상적으로 출력이 되면 정상이고 그렇지 않으면 다시 바이너리 모드로 올려야 한다.

- 모든 설정이 완료되면 웹서버를 restart 시킨다.

3. 전자 서명 생성 클래스(crosscert.Signer)

가) 전자 서명 생성

– PKCS#7 SignedData

● *함수 원형*

Int GetSignedData(byte[] deckeybuf, int deckeylen, byte[] certbuf, int certlen, byte contentbuf, int contentlen)

int GetSignedDataNoContent(byte[] deckeybuf, int deckeylen, byte[] certbuf, int certlen, byte contentbuf, int contentlen)

● *기능*

선택된 서명용 인증서와 비밀키, 인증경로 정보를 이용해 평문에 대한 전자 서명을 수행하여 Base64 인코딩된 형태로 반환한다. GetSignedData()는 원문을 포함한 전자서명이고, GetSignedDataNoContent()는 원문이 없는 전사서명이다.

● ***Parameters***

deckeybuf – 복호화된 개인키
deckeylen – 복호화된 개인키의 길이
certbuf – 인증서
certlen – 인증서의 길이
contentbuf – 서명할 원본 데이터
contentlen – 서명할 원본 데이터의 길이

● ***Return***

String : 실패시 – 0이 아님
성공시 – 0

● ***Example***

```
InputStream inPri = new FileInputStream(new File(CertPath + "kmPri.key"));
InputStream inCert = new FileInputStream(new File(CertPath + "kmCert.der"));
```

```
int nPrilen = inPri.available();
Prifilebuf = new byte[nPrilen];
int nRet = inPri.read(Prifilebuf);

PrivateKey privateKey = new PrivateKey();
nRet=privateKey.DecryptPriKey( "88888888", Prifilebuf, nPrilen);
int certLen = inCert.available();
Certfilebuf = new byte[nCertlen];

Signer signer = new Signer();
nRet=signer.GetSignedData(privateKey.prikeybuf, privateKey.prikeylen,
Certfilebuf, nCertlen, srcData.getBytes("KSC5601"),
srcData.getBytes ("KSC5601").length);
String sSignData = new String(signer.signedbuf, "KSC5601");
out.println("전자서명 데이터 : " + sSignData);
out.println("전자서명 길이 : " + signer.signedlen);
```

4. 전자 서명 검증 클래스(crosscert.Verifier)

나) 전자 서명 검증

– PKCS#7 SignedData

● *함수 원형*

int VerifySignedData (byte[] signedbuf, int signedlen)

int VerifySignedDataNoContent (byte[] signedbuf, int signedlen,
byte[] contentbuf, int contentlen)

● *기능*

전자서명을 선택된 인증서의 공개키를 사용하여 전자서명 검증을 확인한다. 또한 전자 서명을 생성한 인증서의 유효성 검증도 수행된다. VerifySignedData ()는 원문을 포함한 전자서명 검증이고, VerifySignedDataNoContent ()는 원문이 없는 전사서명 검증이다.

● ***Parameters***

signedbuf – 전자 서명 값
signedlen – 전자 서명의 길이
signedbuf – 원문 데이터
signedlen – 원문 데이터의 길이

● ***Return***

int : 실패시 – 0이 아닌 값
성공시 – 0

● ***Example***

– 원문이 있는 전자서명 검증 예

```
Verifier CVerifier = new Verifier();
nRet = CVerifier.VerSignedData(CBase64.contentbuf, CBase64.contentlen);
if (nRet == 0) {
 strBuf = CVerifier.contentbuf;
 nLen = CVerifier.contentlen;
 String sOrgData = new String(strBuf,"KSC5601");
 out.println("전자서명 검증 결과 : 성공₩n");
```

```
 out.println("원문 : " + sOrgData + "₩n");
}
else {
 out.println("전자서명 검증 결과 : 실패₩n");
 out.println("에러내용 : " + CVerifier.errmessage + "₩n");
 out.println("에러코드 : " + CVerifier.errcode + "₩n");
 return;
}
```

- 원문이 없는 전자서명 검증 예

```
Verifier CVerifier = new Verifier();
nRet = CVerifier.VerSignedDataNoContent(signer.signedbuf,
signer.signedlen, srcData.getBytes("KSC5601"),
srcData.getBytes("KSC5601").length);
if (nRet == 0) {
 strBuf = CVerifier.contentbuf;
 nLen = CVerifier.contentlen;
 String sOrgData = new String(strBuf,"KSC5601");
 out.println("전자서명 검증 결과 : 성공₩n");
 out.println("원문 : " + sOrgData + "₩n");
}
else {
 out.println("전자서명 검증 결과 : 실패₩n");
 out.println("에러내용 : " + CVerifier.errmessage + "₩n");
 out.println("에러코드 : " + CVerifier.errcode + "₩n");
 return;
}
```

5. 암호화 클래스(crosscert.Encrypt)

가) 비대칭키 암호화 (EncEnvelopedData)

– PKCS#7 EnvelopedData

● *함수 원형*

int EncEnvelopedData(byte[] certbuf, int certlen, byte[] contentbuf,
int contentlen)

● *기능*

입력된 평문을 인증서의 비공개키를 사용하여 PKCS#7의 EnvelopedData로 생성한 후 Base64 인코딩된 형태로 반환한다. 암호화에 사용 할 수신자의 암호화용 인증서는 로컬 디렉토리에 있을 경우에는 로컬에서 찾아오고, 없을 경우에는 LDAP에서 인증서를 얻어서 사용한다.

● ***Parameters***

certbuf – 수신자 암호화용 인증서.
certlen – 수신자 암호화용 인증서 길이
contentbuf – 암호화할 원문 데이터.
contentlen – 암호화할 원문 데이터 길이

● ***Return***

int : 실패시 – 0이 아님
성공시 – 0

● ***Example***

```
Encrypt encrypt = new Encrypt();
int nRet = encrypt.EncEnvelopedData(Certfilebuf, nCertlen,
                                    srcData.getBytes("KSC5601"),
                                    srcData.getBytes("KSC5601").length);
String sEncData = new String(encrypt.envelopedbuf, "KSC5601");
out.println("암호문(PKCS) 데이터 : " + sEncData);
out.println("암호문(PKCS) 길이 : " + encrypt.envelopedlen);
```

나) 대칭키 암호화 (EncryptData)

- *함수 원형*

int EncryptData(byte[] contentbuf, int contentlen, String password)

- *기능*

입력된 평문을 인증서의 대칭키를 사용하여 암호화 생성한 후 Base64 인코딩 된 형태로 반환한다.

- ***Parameters***

contentbuf – 암호화할 원문 데이터
contentlen – 암호화할 원문 데이터 길이
password – 암호화할 패스워드

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Encrypt encrypt = new Encrypt();
int nRet = encrypt.EncryptData(srcData.getBytes("KSC5601"),
srcData.getBytes("KSC5601").length, "88888888");
String sEncData = new String(encrypt.envelopedbuf, "KSC5601");
out.println("암호문(PKCS) 데이터 : " + sEncData);
out.println("암호문(PKCS) 길이 : " + encrypt.envelopedlen);
```

6. 복호화 클래스(crosscert.Decrypt)

다) 비대칭키 복호화 (DecEnvelopedData)

– PKCS#7 EnvelopedData

● *함수 원형*

int DecEnvelopedData (byte[] deckeybuf, int deckeylen, byte[] certbuf,
int certlen, byte[] envelopedbuf, int envelopedlen)

● *기능*

암호화된 EnvelopedData를 비공개키로 풀어서 원문을 생성한다.

● ***Parameters***

deckeybuf – 복호화에 사용할 개인키
deckeylen – 복호화에 사용할 개인키의 길이
certbuf – 복호화에 사용할 인증서
certlen – 복호화에 사용할 인증서의 길이
envelopedbuf – 암호화된 데이터
envelopedlen – 암호화된 데이터의 길이

● ***Return***

int : 실패시 – 0이 아님
성공시 – 0

● ***Example***

```
PrivateKey CPrivateKey = new PrivateKey();
int nRet = CPrivateKey.DecryptPriKey("1111111111111111", Prifilebuf,
nPrilen);

Base64 CBase64 = new Base64();
nRet = CBase64.Decode(EncryptData.getBytes("KSC5601"),
                      EncryptData.getBytes("KSC5601").length);

Decrypt CDecrypt = new Decrypt();
nRet = CDecrypt.DecEnvelopedData(CPrivateKey.prikeybuf,
```

```
                                        CPrivateKey.prikeylen, strCertBuf, certLen,
                                        CBase64.contentbuf, CBase64.contentlen);
if (nRet != 0) {
 out.println("DecryptPriKey : 실패₩n");
 out.println("에러내용 :" + CDecrypt.errmessage + "₩n");
 out.println("에러코드 :" + CDecrypt.errcode + "₩n");
 return;
}
String sDecData = new String(CDecrypt.contentbuf, "KSC5601");
out.println("암호화된 값 복호화 :" + sDecData + "₩n");
```

라) 대칭키 복호화 (DecryptData)

- *함수 원형*

int DecryptData (byte[] encrypteddata, int encrypteddatalen,
String password)

- *기능*

암호화된 문을 대칭키로 풀어서 원문을 생성한다.

- ***Parameters***

encrypteddata – 암호화 데이터
encrypteddatalen – 암호화 데이터의 길이
password – 패스워드

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Base64 CBase64 = new Base64();
nRet = CBase64.Decode(EncryptData.getBytes("KSC5601"),
                      EncryptData.getBytes("KSC5601").length);

Decrypt CDecrypt = new Decrypt();
nRet = CDecrypt.DecryptData(CBase64.contentbuf, CBase64.contentlen,
                            "123456789");
if (nRet != 0) {
 out.println("DecryptPriKey : 실패₩n");
 out.println("에러내용 : " + CDecrypt.errmessage + "₩n");
 out.println("에러코드 : " + CDecrypt.errcode + "₩n");
 return;
}
String sDecData = new String(CDecrypt.contentbuf, "KSC5601");
out.println("암호화된 값 복호화 : " + sDecData + "₩n");
```

7. 해쉬 클래스(crosscert.Hash)

가) 해쉬함수 (GetHash)

- *함수 원형*

int GetHash (byte[] contentbuf, int contentlen)

- *기능*

원문 데이터에 대한 해쉬 데이터를 생성한다.

- ***Parameters***

contentbuf – 해쉬할 데이터
contentlen – 해쉬할 데이터의 길이

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Hash CHash = new Hash();
int nRet = CHash.GetHash(data.getBytes(), data.length());
if (nRet != 0) {
 out.println("CHash.GetHash : 실패₩n");
 out.println("에러내용 : " + CHash.errmessage + "₩n");
 out.println("에러코드 : " + CHash.errcode + "₩n");
 return;
}
out.println("해쉬된 값 : " + new String(CHash.contentbuf) + "₩n");
```

8. 인증서 클래스(crosscert.Certificate)

가) 인증서 검증 (ValidateCert)

함수 원형

int ValidateCert (byte[] certbuf, int certlen, String policy, int crlflag)

기능

인증서 유효성 검증을 수행한다.

- ***Parameters***

certbuf – 유효성을 검증할 인증서
certlen – 유효성을 검증할 인증서의 길이
policy – 인증서 정책 – 특정 보안 정책(oid)에 대해서 신뢰
crlflag – 인증서 폐기 여부 검증

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Certificate CCertificate = new Certificate();
String Policies = "1.2.410.200004.5.4.1.1|1.2.410.200004.5.4.1.2|
                   1.2.410.200004.5.4.1.3|1.2.410.200004.5.4.1.4|
                   1.2.410.200004.5.4.1.5";
Int nRet = CCertificate.ValidateCert(CVerifier.certbuf, CVerifier.certlen,
           Policies, 1);
if (nRet == 0) {
 out.println("인증서 유효성 검증 성공 ₩n");
}
else {
 out.println("인증서 유효성 검증 실패₩n");
 out.println("에러내용 : " + CCertificate.errmessage + "₩n");
 out.println("에러코드 : " + CCertificate.errcode + "₩n");
return;
```

```
}
```

나) 식별번호 검증 (VerifyVID)

함수 원형

int VerifyVID (byte[] certbuf, int certlen, byte[] randombuf, int randomlen,
String idn)

기능

식별번호 검증을 수행한다.

- ***Parameters***

certbuf – 식별번호 검증할 인증서
certlen – 식별번호 검증할 인증서의 길이
randombuf – 식별번호 확인 랜덤 값
randomlen – 식별번호 확인 랜덤 값의 길이
idn – 인증서 폐기 여부 검증

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

다) 인증서 정보 추출 (ExtractCertInfo)

함수 원형

int ExtractCertInfo (byte[] certbuf, int certlen)

기능

인증서 정보(기본 필드 및 확장 필드) 내용을 추출한다.

- ***Parameters***

certbuf – 필드 정보를 추출할 인증서
certlen – 필드 정보를 추출할 인증서의 길이

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Verifier CVerifier = new Verifier();
...(서명 검증 - 중략)...
Certificate CCertificate = new Certificate();
nRet = CCertificate.ExtractCertInfo(CVerifier.certbuf, CVerifier.certlen);
if (nRet != 0) {
 out.println("인증서 정보 추출 실패₩n");
 out.println("에러내용 : " + CCertificate.errmessage + "₩n");
 out.println("에러코드 : " + CCertificate.errcode + "₩n");
 return;
}
out.println("인증서 정보 추출 결과 : " + nRet + "₩n");
out.println("버전 : " + CCertificate.version + "₩n");
out.println("일련번호 : " + CCertificate.serial + "₩n");
out.println("발급자 DN : " + CCertificate.issuer + "₩n");
out.println("주체 DN : " + CCertificate.subject + "₩n");
out.println("공개키 알고리즘 : " + CCertificate.subjectAlgId + "₩n");
out.println("유효기간 시작 : " + CCertificate.from + "₩n");
out.println("유효기간 끝 : " + CCertificate.to + "₩n");
```

```
out.println("서명 알고리즘 : " + CCertificate.signatureAlgId + "₩n");
out.println("공개키 : " + CCertificate.pubkey + "₩n");
out.println("서명값 : " + CCertificate.signature + "₩n");
out.println("발급자 대체 이름 : " + CCertificate.issuerAltName + "₩n");
out.println("주체 대체 이름 : " + CCertificate.subjectAltName + "₩n");
out.println("키 사용 용도 : " + CCertificate.keyusage + "₩n");
out.println("보안 정책 : " + CCertificate.policy + "₩n");
out.println("기본 제한 : " + CCertificate.basicConstraint + "₩n");
out.println("정책 제한 : " + CCertificate.policyConstraint + "₩n");
out.println("CRL 배포 지점 : " + CCertificate.distributionPoint + "₩n");
out.println("발급자 키 식별자 : " + CCertificate.authorityKeyId + "₩n");
out.println("주체 키 식별자 : " + CCertificate.subjectKeyId + "₩n");
```

라) LDAP에서 인증서 가져오기 (GetCertFromLdap)

함수 원형

int GetCertFromLdap (String subjectDn, int certType)

기능

LDAP에서 인증서를 가져온다.

- ***Parameters***

subjectDn – LDAP에서 가져올 인증서 DN 값

certType –

- ***Return***

int : 실패시 – 0이 아님

성공시 – 0

- ***Example***

```
Certificate CCertificate = new Certificate();
int nRet = CCertificate.GetCertFromLdap("cn=황민구,ou=개인,ou=비씨카드,ou=licensedCA,o=CrossCert,c=KR", 0);
if (nRet == 0) {
 out.println("LDAP에서 인증서 가져오기 성공 ₩n");
}
else {
 out.println("LDAP에서 인증서 가져오기 실패₩n");
 out.println("에러내용 : " + CCertificate.errmessage + "₩n");
 out.println("에러코드 : " + CCertificate.errcode + "₩n");
return;
}
out.println("LDAP에서 가져온 인증서길이 : " + CCertificate.contentlen );
```

9. 개인키 추출 클래스 (crosscert.PrivateKey)

가) 개인키 추출 (DecryptPriKey)

함수 원형

int DecryptPriKey (String passwd, byte encprikeybuf, int encprikeylen)

기능

서버 인증서의 복호용 개인키를 추출한다.

- ***Parameters***

passwd – 개인키 접근 패스워드
encprikeybuf – 암호화된 개인키
encprikeylen – 암호화된 개인키 길이

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
InputStream inPri = new FileInputStream(new File(CertPath +  "kmPri.key"));
InputStream inCert = new FileInputStream(new File(CertPath +
                                                  "kmCert.der"));
int nPrilen = inPri.available();
Prifilebuf = new byte[nPrilen];
int nRet = inPri.read(Prifilebuf);
int certLen = inCert.available();
byte[] strCertBuf = new byte[certLen];
nRet = inCert.read(strCertBuf);
PrivateKey CPrivateKey = new PrivateKey();
nRet = CPrivateKey.DecryptPriKey("1111111111111111", Prifilebuf,
                                                  nPrilen);
if (nRet != 0) {
 out.println("DecryptPriKey : 실패₩n");
 out.println("에러내용 : " + CPrivateKey.errmessage + "₩n");
```

```
 out.println("에러코드 : " + CPrivateKey.errcode + "₩n");
 return;
}
out.println("서버인증서개인키 길이 : " + CPrivateKey.prikeylen + "₩n");
out.println("서버인증서 길이 : " + certLen + "₩n");
```

10. base64 함수(crosscert.Base64)

가) base64 인코딩 (Encode)

함수 원형

int Encode (byte[] decodedbuf, int decodedlen)

기능

원문을 Base64로 인코딩한다.

- ***Parameters***

decodedbuf: 원문 데이터
decodedlen: 원문 데이터의 길이

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Base64 CBase64 = new Base64();
int nRet = CBase64.Encode(srcData.getBytes(), srcData.length());
if (nRet != 0) {
 out.println("CBase64 : 실패₩n");
 out.println("에러내용 : " + CBase64.errmessage+ "₩n");
 out.println("에러코드 : " + CBase64.errcode + "₩n");
 return;
}
out.println("CBase64 : 성공₩n"+ new String(CBase64.contentbuf));
```

나) base64 디코딩 (Decode)

함수 원형

int Decode (byte[] encodedbuf, int encodedlen)

기능

Base64로 인코딩된 문을 디코딩한다.

- ***Parameters***

encodedbuf: base64 인코딩 데이터
encodedlen: base64 인코딩 데이터의 길이

- ***Return***

int : 실패시 – 0이 아님
성공시 – 0

- ***Example***

```
Base64 CBase64 = new Base64();
int nRet = CBase64.Decode(sBase64EncData.getBytes(),
                          sBase64EncData.length());
if (nRet != 0) {
  out.println("CBase64 : 실패₩n");
  out.println("에러내용 : " + CBase64.errmessage+ "₩n");
  out.println("에러코드 : " + CBase64.errcode + "₩n");
  return;
}
out.println("CBase64 : 성공₩n"+ new String(CBase64.contentbuf));
```