공인인증서 활용 가이드

2023년 02월 06일

(주)틸코블렛

help@tilko.net

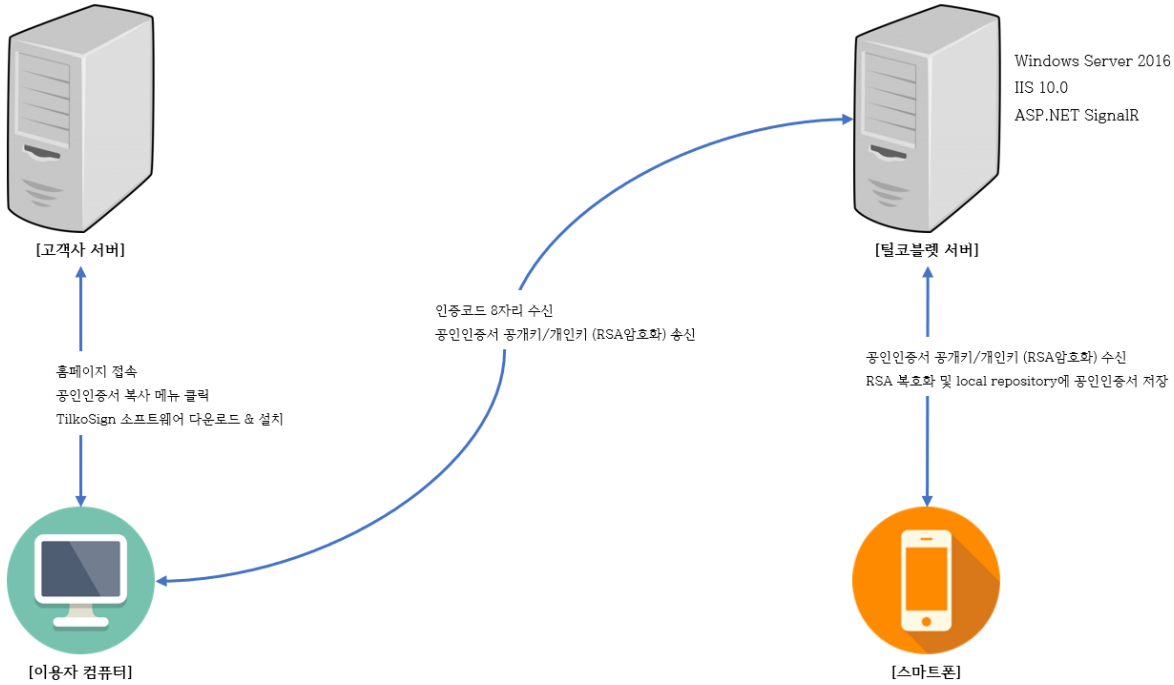
기술지원문의 : 1670-2822

개 정 이 력

순번	버전	작성자	날짜	내용
1	v1.0	손정민	2020-11-25	최초 작성
2	v1.1	서성원	2021-11-19	참조 이미지 수정
3	v1.2	서성원	2021-12-21	API Key Type 추가
4	v2.0	서성원	2022-01-14	Client Type 분기
5	v2.1	서성원	2022-01-17	인증서 추가 정보 전달
6	v2.2	서성원	2022-02-16	인증서 정보 반환 형태 수정
7	v2.3	서성원	2022-05-23	문서 제목 변경
8	v2.4	서성원	2022-06-21	MobileWebView Type 추가
9	v2.5	서성원	2022-06-28	Pc Type 에서 API 호출
10	v2.6	서성원	2023-02-06	API 호출 부분 설명 수정
11	v2.7	한인수	2023-02-06	TilkoSign 콜백 함수 파라미터 수정
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				

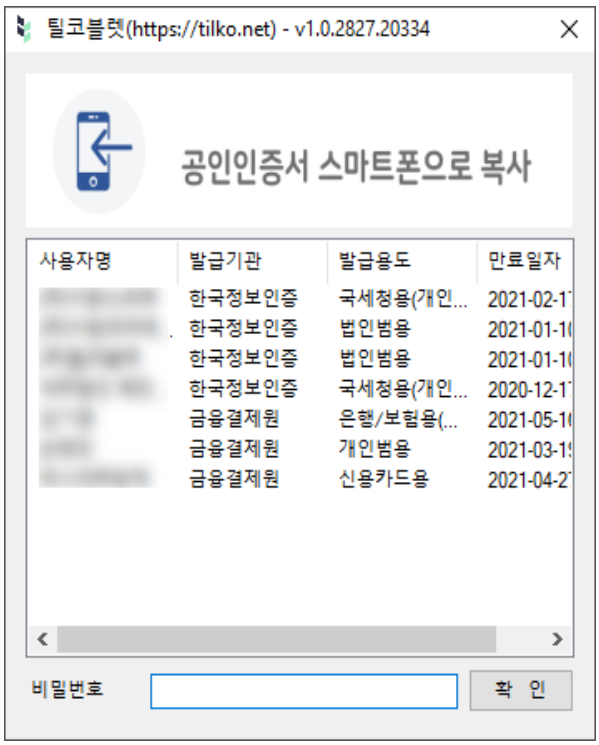
1 시스템 구성

1) 클라이언트가 모바일(웹뷰 포함)인 경우

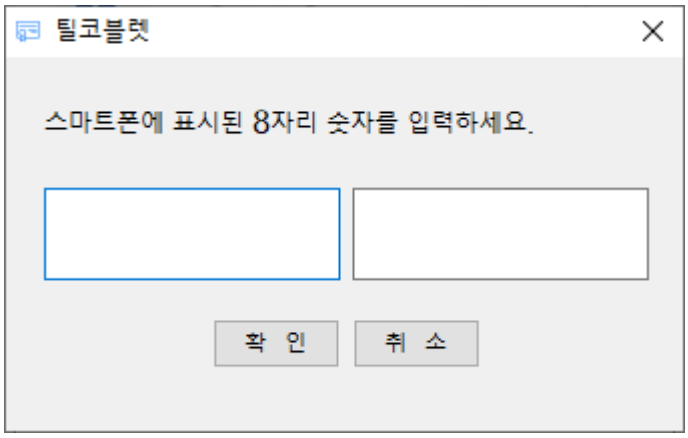


a. 이용자 컴퓨터에서 고객사 서버에 접속하여, "공인인증서를 스마트폰으로 복사하기" 버튼을 클릭하면, PC에 설치되어 있는 TilkoSign 소프트웨어를 자동으로 실행하게 됩니다.
이때, 설치된 TilkoSign이 없을 경우 이용자는 링크를 통하여 소프트웨어를 다운로드 받고 해당 소프트웨어를 설치할 필요가 있습니다.

b. 이용자 컴퓨터에 TilkoSign 소프트웨어가 실행되면, 아래 화면과 같이 컴퓨터에 있는 공인인증서 목록을 로드합니다.



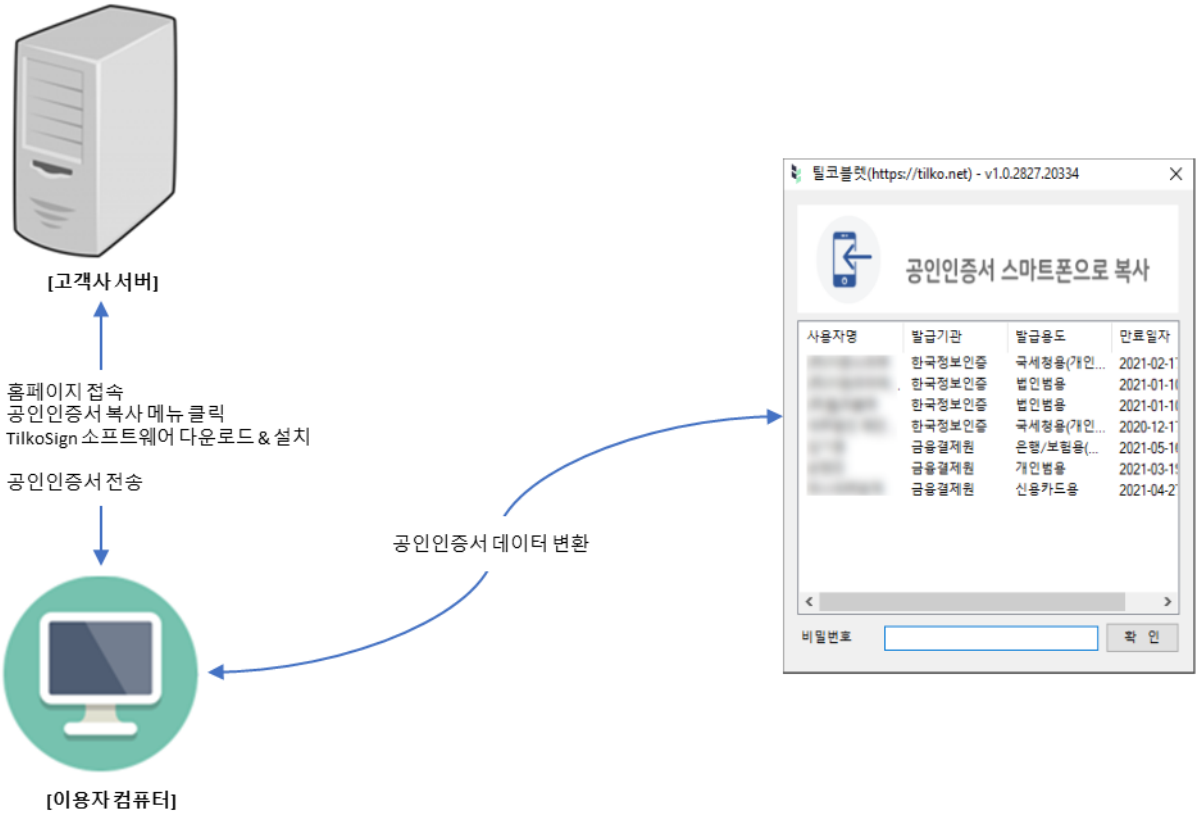
c. 스마트폰으로 복사할 공인인증서를 선택하고, 비밀번호를 입력한 후 확인 버튼을 클릭하면 스마트폰에 표시된 8자리 코드를 입력하는 창이 나타납니다.
입력되는 비밀번호는 키보드 보안모듈에 의해서 키 입력 값을 알 수 없게 처리합니다.



d. 스마트폰에 표시된 8자리 인증코드를 입력하면, 공인인증서의 공개키/개인키가 틸코블렛 서버를 통하여 스마트폰으로 전송됩니다.
이때, 틸코블렛은 이용자의 공인인증서 파일을 서버에 저장하지 않습니다.

e. 틸코블렛 서버, 이용자 컴퓨터 및 스마트폰과의 모든 통신은 암호화됩니다.
기본적으로 데이터는 AES-128에 의해서 암호화가 되면, AES 암호화에 사용된 키는 RSA-2048에 의해서 암호화됩니다.
전달되는 공개키만으로 암호화를 한 후, 데이터를 전송하며 보유하고 있는 개인키로만 복호화가 가능합니다.

2) 클라이언트가 PC인 경우



a. 이용자 컴퓨터에서 고객사 서버에 접속하여, “공인인증서를 스마트폰으로 복사하기” 버튼을 클릭하면, PC에 설치되어 있는 TilkoSign 소프트웨어를 자동으로 실행하게 됩니다.
이때, 설치된 TilkoSign이 없을 경우 이용자는 링크를 통하여 소프트웨어를 다운로드 받고 해당 소프트웨어를 설치할 필요가 있습니다.

b. 이용자 컴퓨터에 TilkoSign 소프트웨어가 실행되면, 아래 화면과 같이 컴퓨터에 있는 공인인증서 목록을 로드합니다.

c. 스마트폰으로 복사할 공인인증서를 선택하고, 비밀번호를 입력한 후 확인 버튼을 클릭하면 공인인증서의 정보가 자바스크립트의 지정된 콜백 함수로 반환됩니다.
이때, 틸코블렛은 이용자의 공인인증서 파일을 서버에 저장하지 않습니다.

d. 지정된 콜백 함수를 이용하여 고객사 서버로 전송할 수 있습니다. 이때, 고객사의 상황에 맞게 암호화하여 전송할 것을 권장합니다.

e. 인증서를 외부로 전송하는 것이 아닌, API 호출을 위해 이용자 PC에서 이용하는 경우에는, 틸코 API 암호화 규격을 따릅니다.

2 웹 페이지 구성 가이드

1) 클라이언트가 모바일인 경우

a. 고객사의 웹 서버에는 자바스크립트와 html 태그만으로 인증서 복사 프로그램을 구성할 수 있습니다.

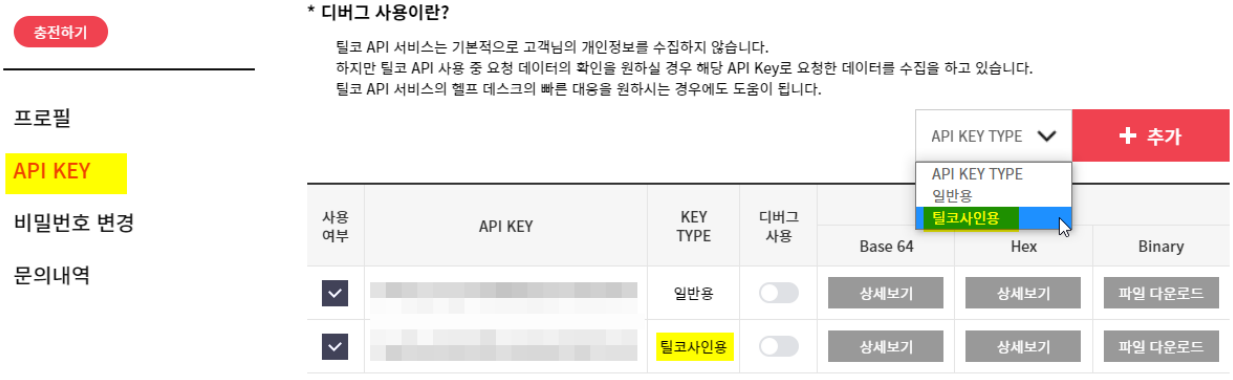
b. 아래는 자바스크립트 코드입니다. (https://github.com/Tilkoblet/TilkoSign/blob/main/index.html)

```
<script type="text/javascript">
    let _tksReq = new TilkoSignRequest("general", "ping");

    function callScheme(api_key, hubServerUrl, bannerUrl, clientType) {
        try {
            var siteUrl = window.location.host;

            _tksReq.Add("apiKey=" + api_key);
            _tksReq.Add("hubServerUrl=" + hubServerUrl);
            _tksReq.Add("bannerUrl=" + bannerUrl);
            _tksReq.Add("siteUrl=" + siteUrl);
            _tksReq.Add("clientType=" + clientType);

            // TilkoSign 설치여부 확인
            _tksReq.Send();

            var _scheme     = api_key + "|" + hubServerUrl + "|" + bannerUrl + "|" + siteUrl + "|" + clientType;
            _scheme         = _scheme.hexEncode();
            location.href   = "tilkosign://" + _scheme;
        }
        catch (e) {
            alert(e);
        }
    }
</script>
```

c. 아래는 “공인인증서 복사하기” 버튼에 처리할 링크입니다.

```
<a href="javascript:callScheme('틸코사인용 API KEY', 'https://cert.tilko.net/', '', 'Mobile');">
    <h3 class="guide-title">공동인증서 복사하기</h3>
</a>
```

d. callScheme 함수의 첫 번째 파라미터는 틸코 API 홈페이지에서 발급받은 API키(틸코사인용) 값입니다.
필히 실제 API키 값을 입력해 주세요. 샘플 페이지의 API키 값을 입력하는 경우에는 프로그램이 정상적으로 동작되지 않습니다.



e. callScheme 함수의 두 번째 파라미터는 https://cert.tilko.net을 고정해 주세요. 별도의 인증서 복사 서버를 구성하지 않는 경우에는 틸코블렛의 인증서 복사 서버를 이용하시게 됩니다.

f. callScheme 함수의 세 번째 파라미터는 이용자 컴퓨터의 TilkoSign 소프트웨어의 상단 배너에 표시될 이미지의 url입니다. http://abc.com/def.png 등의 전체 url을 입력해 주세요.
배너 이미지의 크기는 768 * 160픽셀이 가장 적합합니다.

g. callScheme 함수의 네 번째 파라미터는 클라이언트의 타입입니다.

2) 클라이언트가 PC인 경우

a. 고객사의 웹 서버에는 자바스크립트와 html 태그만으로 인증서 복사 프로그램을 구성할 수 있습니다.

b. 아래는 자바스크립트 코드입니다. (https://github.com/Tilkoblet/TilkoSign/blob/main/index.html)

```
<script type="text/javascript">
    let _tksReq = new TilkoSignRequest("general", "ping", getCert);

    function callScheme(api_key, hubServerUrl, bannerUrl, clientType) {
        try {
            var siteUrl = window.location.host;

            _tksReq.Add("apiKey=" + api_key);
            _tksReq.Add("hubServerUrl=" + hubServerUrl);
            _tksReq.Add("bannerUrl=" + bannerUrl);
            _tksReq.Add("siteUrl=" + siteUrl);
            _tksReq.Add("clientType=" + clientType);

            // TilkoSign 설치여부 확인
            _tksReq.Send();

            var _scheme     = api_key + "|" + hubServerUrl + "|" + bannerUrl + "|" + siteUrl + "|" + clientType;
            _scheme         = _scheme.hexEncode();
            location.href   = "tilkosign://" + _scheme;
        }
        catch (e) {
            alert(e);
        }
    }

    function getCert(obj) {
        console.log(obj);
    }
</script>
```

TilkoSignRequest의 마지막 파라미터에는 인증서를 받을 콜백함수를 넣어주시면 됩니다. 이 콜백함수를 통해 인증서를 고객사 서버에 전송하는 로직을 넣으시면 됩니다.
콜백함수로 들어오는 파라미터는 다음과 같습니다.

{

PubFile	: 공개키 파일명
PriFile	: 개인키 파일명
PubKey	: 공개키 파일의 바이너리 데이터를 Base64 String으로 변환한 값
PriKey	: 개인키 파일의 바이너리 데이터를 Base64 String으로 변환한 값
Pwd	: 인증서 비밀번호의 아스키값을 Base64로 변환한 값
Name	: 인증서 이름,
Issuer	: 발행기관,
Purpose	: 용도,
Expire	: 만료일,
Dn	: DN
IsPwdCorrect	: 인증서 비밀번호 일치여부(bool)

```
}
```

c. 아래는 “공인인증서 복사하기” 버튼에 처리할 링크입니다.

```
<a href="javascript:callScheme('틸코사인용 API KEY', 'https://cert.tilko.net/', '', 'Pc');">
    <h3 class="guide-title">공동인증서 복사하기</h3>
</a>
```

d. 나머지 부분은 클라이언트가 모바일인 경우와 동일합니다.

e. 이용자 PC 자체적으로 API 호출을 위해 틸코사인을 이용하는 경우에는, 다음의 스크립트를 참조하시기 바랍니다. (https://github.com/Tilkoblet/TilkoSign/blob/main/PcSample.html)

```
const apiKey = "일반용 API KEY"
const hubUrl = "https://cert.tilko.net/";
const apiHost = "https://api.tilko.net/";
```

```
function getCert(obj) {
    console.log(obj);
    callApi(obj);
}
```

```
async function callApi(cert) {
    // AES Key를 RSA Public Key로 암호화
    const rsaPublicKey = await getPublicKey().then(function (data) {
        // AES Secret Key 및 IV 생성
        const aesKey = CryptoJS.enc.Utf8.parse('1234567890123456');                 // 16 바이트 랜덤값 사용 가능
        const aesIv = CryptoJS.enc.Hex.parse("00000000000000000000000000000000");  // 고정값

        const rsaPublicKey = data.PublicKey;

        const aesCipherKey = rsaEncrypt(rsaPublicKey, aesKey, "pkcs1");

        const uri = apiHost + "/api/v1.0/Gov/AA090UserJuminCheckResApp";

        const headers = {
            "Content-Type": "application/json",
            "API-KEY": apiKey,
            "ENC-KEY": aesCipherKey
        };

        const params = {
            "CertFile": aesEncryptCert(aesKey, aesIv, cert.PubKey),
            "KeyFile": aesEncryptCert(aesKey, aesIv, cert.PriKey),
            "CertPassword": aesEncrypt(aesKey, aesIv, CryptoJS.enc.Base64.parse(cert.Pwd).toString(CryptoJS.enc.Utf8)),
            "PersonName": "홍길동",
            "IdentityNumber": aesEncrypt(aesKey, aesIv, "8801011234567"),
            "PublishDate": "20200101",
        };

        $.ajax({
            type: "POST",
            url: uri,
            headers: headers,
            data: JSON.stringify(params),
            contentType: "application/json",
            success: function (data) {
                console.log(data);
            },
            error: function (jqXHR, textStatus, errorThrown) {
                console.error(textStatus);
                console.error(jqXHR);
            }
        });
    });
}
```

단, API Key가 외부에 노출될 위험이 있는 경우에는 위의 스크립트대로 진행하시기 보다는, 서버로 인증서를 보낸 후, 서버단에서 API를 호출하는 것이 보안상 안전합니다. 각 언어별 API 호출 샘플 코드 또한 깃허브를 통해 제공해드리고 있습니다. (https://github.com/Tilkoblet?tab=repositories)

3) 클라이언트가 웹뷰용 모바일인 경우

a. 고객사의 웹 서버에는 자바스크립트와 html 태그만으로 인증서 복사 프로그램을 구성할 수 있습니다.

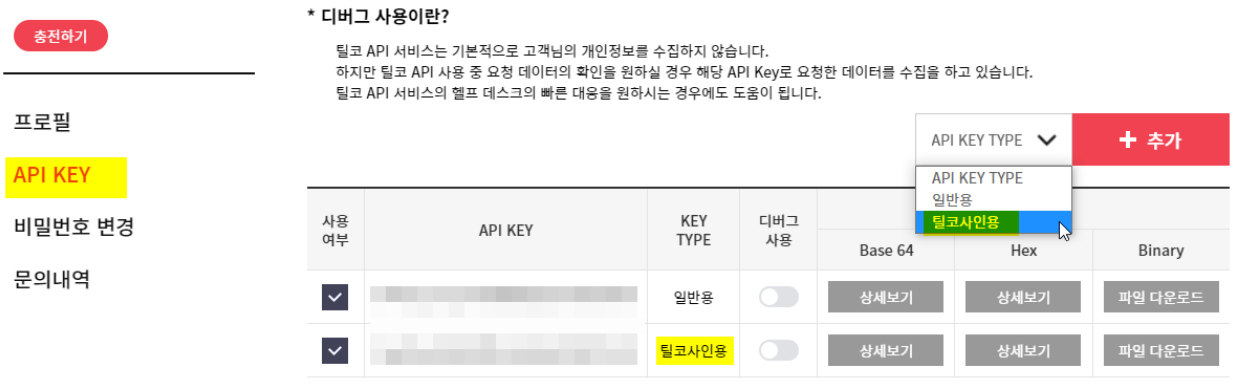
b. 아래는 자바스크립트 코드입니다. (https://github.com/Tilkoblet/TilkoSign/blob/main/index.html)

```
<script type="text/javascript">
    let _tksReq = new TilkoSignRequest("general", "ping");

    function callScheme(api_key, hubServerUrl, bannerUrl, clientType) {
        try {
            var siteUrl = window.location.host;

            _tksReq.Add("apiKey=" + api_key);
            _tksReq.Add("hubServerUrl=" + hubServerUrl);
            _tksReq.Add("bannerUrl=" + bannerUrl);
            _tksReq.Add("siteUrl=" + siteUrl);
            _tksReq.Add("clientType=" + clientType);

            // TilkoSign 설치여부 확인
            _tksReq.Send();

            var _scheme     = api_key + "|" + hubServerUrl + "|" + bannerUrl + "|" + siteUrl + "|" + clientType;
            _scheme         = _scheme.hexEncode();
            location.href   = "tilkosign://" + _scheme;
        }
        catch (e) {
            alert(e);
        }
    }
</script>
```

c. 아래는 “공인인증서 복사하기” 버튼에 처리할 링크입니다.

```
<a href="javascript:callScheme('틸코사인용 API KEY', 'https://cert.tilko.net/', '', 'MobileWebView');">
    <h3 class="guide-title">공동인증서 복사하기</h3>
</a>
```

d. callScheme 함수의 첫 번째 파라미터는 틸코 API 홈페이지에서 발급받은 API키(틸코사인용) 값입니다.
필히 실제 API키 값을 입력해 주세요. 샘플 페이지의 API키 값을 입력하는 경우에는 프로그램이 정상적으로 동작되지 않습니다.



e. callScheme 함수의 두 번째 파라미터는 https://cert.tilko.net을 고정해 주세요. 별도의 인증서 복사 서버를 구성하지 않는 경우에는 틸코블렛의 인증서 복사 서버를 이용하시게 됩니다.

f. callScheme 함수의 세 번째 파라미터는 이용자 컴퓨터의 TilkoSign 소프트웨어의 상단 배너에 표

시될 이미지의 url입니다. http://abc.com/def.png 등의 전체 url을 입력해 주세요.
배너 이미지의 크기는 768 * 160픽셀이 가장 적합합니다.

g. callScheme 함수의 네 번째 파라미터는 클라이언트의 타입입니다.

h. 모바일 페이지 구성은 아래의 깃허브 샘플 코드를 참조하시면 됩니다.
(https://github.com/Tilkoblet/TilkoSign/blob/main/WebViewSample.html)

i. 인증서 복사시, 인증서는 웹 브라우저의 LocalStorage에 저장됩니다.

단, API Key가 외부에 노출될 위험이 있는 경우에는 위의 스크립트대로 진행하시기 보다는, 서버로 인증서를 보낸 후, 서버단에서 API를 호출하는 것이 보안상 안전합니다. 각 언어별 API 호출 샘플코드 또한 깃허브를 통해 제공해드리고 있습니다. (https://github.com/Tilkoblet?tab=repositories)

3 Android 코틀린 개발 가이드

a. 접속할 server url은 웹 페이지 구성 가이드의 callScheme 두 번째 파라미터와 동일해야 합니다.

```
@RequiresApi(Build.VERSION_CODES.O)
class MainActivity : AppCompatActivity() {
    private var rsa:RSA? = null
    private val server_url = "https://cert.tilko.net/"
    private var pem:String = ""
    private var hubConnection:HubConnection? = null

    val dateFormat = SimpleDateFormat( pattern: "yyyy.MM.dd")


    // 서버 RSA Public Key 불러오기
    suspend fun getRSAPublcKey() {

        val (request, response, result) = Fuel.get( path: pubUrl + apiKey)
            .header( header: "Content-Type",  value: "application/json")
            .awaitStringResponseResult()

        val (json, error) = result

        error.let { it: FuelError?
            println(it?.localizedMessage)
        }
```

b. 틸코 API에 사용할 공인인증서의 주민등록번호 및 인증서 비밀번호를 설정합니다.

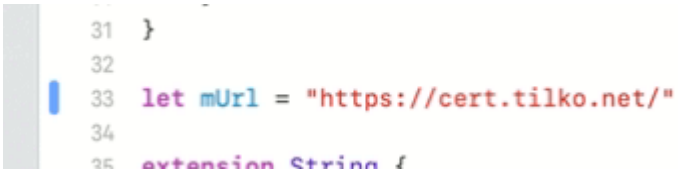
```
154
155         val ssn = ""            // 주민번호
156         val carrier = ""                    // 통신사
157         val phoneNumber = ""      // 휴대폰
158         val certPass = ""          // 인증서암호
159
```

c. API의 성격에 따라 필요한 body 정보들을 추가해 주시면 됩니다.

```
182         val body = JSONObject()
183
184         body.put( name: "CertFile", Base64.encodeToString(certCipherBytes!!, Base64.NO_WRAP))
185         body.put( name: "KeyFile", Base64.encodeToString(keyCipherBytes!!, Base64.NO_WRAP))
186         body.put( name: "IdentityNumber", Base64.encodeToString(ssnCipherBytes!!, Base64.NO_WRAP))
187         body.put( name: "CertPassword", Base64.encodeToString(certPassCipherBytes!!, Base64.NO_WRAP))
188         body.put( name: "TelecomCompany", carrier)
189         body.put( name: "CellphoneNumber", Base64.encodeToString(phoneNumberCipherBytes!!, Base64.NO_WRAP))
190
```

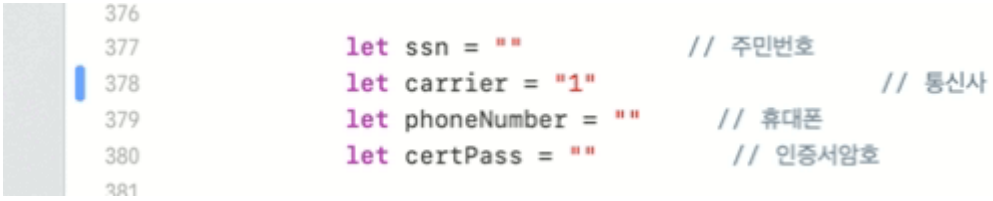
4 iOS 스위프트 개발 가이드

a. 접속할 server url은 웹 페이지 구성 가이드의 callScheme 두 번째 파라미터와 동일해야 합니다.



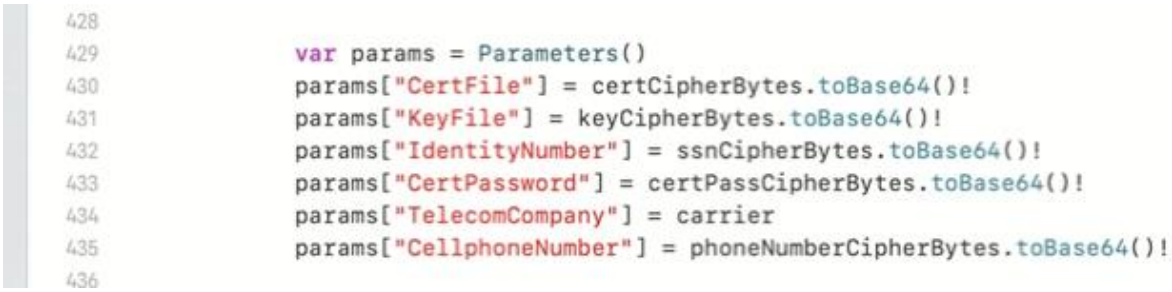
```
31 }
32
33 let mUrl = "https://cert.tilko.net/"
34
35 extension String {
```

b. 틸코 API에 사용할 공인인증서의 주민등록번호 및 인증서 비밀번호를 설정합니다.



```
376
377             let ssn = ""            // 주민번호
378             let carrier = "1"                    // 통신사
379             let phoneNumber = ""     // 휴대폰
380             let certPass = ""         // 인증서암호
381
```

c. API의 성격에 따라 필요한 body 정보들을 추가해 주시면 됩니다.



```
428
429             var params = Parameters()
430             params["CertFile"] = certCipherBytes.toBase64()!
431             params["KeyFile"] = keyCipherBytes.toBase64()!
432             params["IdentityNumber"] = ssnCipherBytes.toBase64()!
433             params["CertPassword"] = certPassCipherBytes.toBase64()!
434             params["TelecomCompany"] = carrier
435             params["CellphoneNumber"] = phoneNumberCipherBytes.toBase64()!
436
```

5 스마트폰 API 호출 로직

a. 필수 암호화 파라미터

틸코 API 홈페이지에 [암호화]라고 표시되어 있는 항목들은 필수적으로 암호화 단계를 거친 후, base64 인코딩 처리를 하고 전송해야 합니다.

BODY

Name	Type	Description	Required
CertFile	String	[암호화] 인증서 공개키(Base64 인코딩)	O
KeyFile	String	[암호화] 인증서 개인키(Base64 인코딩)	O
CertPassword	String	[암호화] 인증서 암호(Base64 인코딩)	O
AgentId	String	[암호화] 세무대리인 ID(세무대리 관리번호가 있는 경우 / Base64 인코딩)	
AgentPassword	String	[암호화] 세무대리인 암호(세무대리 관리번호가 있는 경우 / Base64 인코딩)	
BusinessNumber	String	[암호화] 검색 할 사업자등록번호 또는 주민등록번호(xxxxxxxxxx 또는 xxxxxxxxxxxx / Base64 인코딩)	O
Year	String	검색년도(yyyy) 공백일 경우 검색 기준 해	

b. 기본 암호화 로직 : AES-128-CBC

기본적으로 AES-128-CBC 암호화를 수행합니다. 이때 암호화 키 16 바이트는 RSA-2048 공개키로 암호화한 후, API 서버로 헤더(ENC-KEY 항목)에 실어서 전송합니다.

```
177         val header = hashMapOf<String, Any>()
178         header["Content-Type"] = "application/json"
179         header["API-Key"] = apiKey
180         header["ENC-KEY"] = Base64.encodeToString(aesCipherKey!!, Base64.NO_WRAP)
```

c. RSA 공개키는 API 키 별로 별도로 생성되며, 틸코 API 홈페이지에서 “서버용 공개키” 항목을 통하여 확인할 수 있습니다.

사용 여부	API KEY	KEY TYPE	디버그 사용	서버용 공개키		
				Base 64	Hex	Binary
✓	[redacted]	일반용		상세보기	상세보기	파일 다운로드
✓	[redacted]	틸코사인용		상세보기	상세보기	파일 다운로드

d. 또한, 아래 RESTful 페이지를 통하여 API 키에 따른 RSA(서버용 공개키) 공개키 값을 수신할 수도 있습니다.

https://api.tilko.net/api/Auth/GetPublicKey

- END OF DOCUMENT -