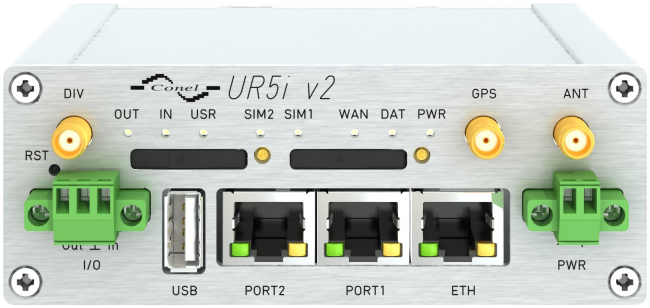
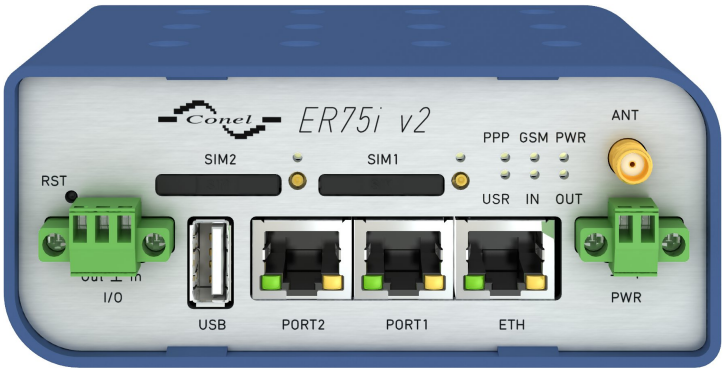
User Module

SCEP Client

APPLICATION NOTE





ADVANTECH

Used symbols

Danger – Information regarding user safety or potential damage to the router.

Attention – Problems that may arise in specific situations.

Information or notice – Useful tips or information of special interest.

Example – example of function, command or script.

Advantech B+B SmartWorx s.r.o., Sokolska 71, 562 04 Usti nad Orlici, Czech Republic
Document No. APP-0062-EN, revised on November 30, 2018. Released in the Czech Republic.

Contents

List of Tables

1. Basic information

User module *SCEP Client* is not contained in the standard router firmware. Uploading of any user module is described in the Configuration manual (see [1]).

The user module is v2 and v3 router platforms compatible.

1.1 What is SCEP?

SCEP (Cisco System's Simple Certificate Enrollment Protocol) is a PKI communication protocol which leverages existing technology by using PKCS#7 and PKCS#10. SCEP is the evolution of the enrollment protocol developed by Verisign, Inc. for Cisco Systems, Inc. It now enjoys wide support in both client and CA implementations.

The goal of SCEP is to support the secure issuance of certificates to network devices in a scalable manner, using existing technology whenever possible. The protocol supports the following operations:

- CA and RA public key distribution
- Certificate enrollment
- Certificate and CRL query

Certificate and CRL access can be achieved by using the LDAP protocol, or by using the query messages defined in SCEP.

2. How to use

Running the command *sscep* without any arguments should give you a list of arguments and command line options.

Usage: /opt/scepClient/bin/sscep OPERATION [OPTIONS]

Available OPERATIONs are:

OPERATION	Description
getca	Get CA/RA certificate(s)
enroll	Enroll certificate
getcert	Query certificate
getcrl	Query CRL

Table 1: Available OPERATIONs

General OPTIONS:

OPTION	Description
-u <url>	SCEP server URL
-p <host:port>	Use proxy server at host:port
-f <file>	Use configuration file
-c <file>	CA certificate file (write if OPERATION is getca)
-E <name>	PKCS#7 encryption algorithm (des|3des|blowfish)
-S <name>	PKCS#7 signature algorithm (md5|sha1)
-v	Verbose operation
-d	Debug (even more verbose operation)

Table 2: General OPTIONS

OPTIONS for OPERATION *getca* are:

OPTION	Description
-i <string>	CA identifier string
-F <name>	Fingerprint algorithm (md5|sha1)

Table 3: OPTIONS for OPERATION *getca*

OPTIONS for OPERATION *enroll* are:

OPTION	Description
-k <file>	Private key file
-r <file>	Certificate request file
-K <file>	Signature private key file
-O <file>	Signature certificate (used instead of self-signed)
-l <file>	Write enrolled certificate in file
-e <file>	Use different CA cert for encryption
-L <file>	Write selfsigned certificate in file
-t <secs>	Polling interval in seconds
-T <secs>	Max polling time in seconds
-n <count>	Max number of GetCertInitial requests
-R	Resume interrupted enrollment

Table 4: OPTIONS for OPERATION *enroll*

OPTIONS for OPERATION *getcert* are:

OPTION	Description
-k <file>	Private key file
-l <file>	Local certificate file
-s <number>	Certificate serial number (decimal)
-w <file>	Write certificate in file

Table 5: OPTIONS for OPERATION *getcert*

OPTIONS for OPERATION *getcrl* are:

OPTION	Description
-k <file>	Private key file
-l <file>	Local certificate file
-w <file>	Write CRL in file

Table 6: OPTIONS for OPERATION *getcrl*

SSCEP also supports configuration via a configuration file (-f). This is the recommended way to configure SSCEP and all the examples in below assume that you have done so.

All configuration options are key-value pairs separated with one or more space characters:

```
"Key"      [spaces]      "Value"
```

Quotation marks are optional – they are needed only if the value contains space characters (space or tab). Quotation marks inside the value string must be escaped using a backslash:

```
"Key"      [spaces]      "Value \"containing quotation marks\""
```

Comment lines (lines starting with '#') and empty lines are discarded.

2.1 Available configuration file keys and example values

Here are the available configuration file keys and example values:

- **CACertFile** – This is one is needed with all operations.

  Example: ./ca.crt
  Command line option: -c

- **CAIdentifier** – Some CAs require you to define this.

  Example: mydomain.com
  Command line option: -i

- **CertReqFile** – Certificate request file created with mkrequest.

  Example: ./local.csr
  Command line option: -r

- **Debug** – Debug? Answer "yes" or "no".

  Command line option: -d

- **EncAlgorithm** – PKCS#7 encryption algorithm. Available algorithms are des, 3des and blowfish. NOTE: this could be very misleading, current SCEP draft provides no mechanism to "negotiate" the algorithm – even if you send 3des, reply might be des (same thing applies to SigAlgorithm).

  Command line option: -E

- **EncCertFile** – If your CA/RA uses a different certificate for encyption and signing, define this. CACertFile is used for verifying the signature.

  Example: ./enc.crt
  Command line option: -e

- **SignCertFile** – Instead of creating a self-signed certificate from the new key pair use an already existing certficate/key to sign the SCEP request. If the "old" certificate and key is used, the CA can verify that the holder of the private key for an existing certificate re-enrolls for a renewal certificate, allowing for automatic approval of the request. Requires specification of the corresponding signature private key file (-K, SignKeyFile).

  Example: ./sig.crt
  Command line option: -O

- **SignKeyFile** – See SignCertFile. Specifies the corresponding private key.

  Example: ./sig.key
  Command line option: -K

- **FingerPrint** – Display fingerprint algorithm. Available algorithms are md5 and sha1. Default is md5.

  Command line option: -F

- **GetCertFile** – Write certificate asquired via getcert operation.

  Example: ./cert.crt
  Command line option: -w

- **GetCertSerial** – Certificate serial number. Define this for getcert. The value is defined as a decimal number.

  Example: 12
  Command line option: -s

- **GetCrlFile** – Write CRL to file.

  Example: ./crl.crl
  Command line option: -w

- **LocalCertFile** – Write successfully enrolled certificate.

  Example: ./local.crt
  Command line option: -l

- **MaxPollCount** – Max number of GetCertInitial requests.

  Example: 50
  Command line option: -n

- **MaxPollTime** – Max polling time in seconds.

  Example: 28800
  Command line option: -T

- **PollInterval** – Poll periodically for pending certificate.

  Example: 60
  Command line option: -t

- **PrivateKeyFile** – Private key created with mkrequest.

  Example: ./local.key
  Command line option: -k

- **Proxy** – Use HTTP proxy at host:port.

  Example: localhost:8080
  Command line option: -p

- **SelfSignedFile** – Write optionally the selfsigned certificate in file (needed in SCEP transaction).

  Example: ./selfsigned.crt
  Command line option: -L

- **SigAlgorithm** – PKCS#7 signature algorithm. Available algorithms are md5 and sha1. Default is md5.

  Command line option: -E

- **URL** – URL of the SCEP server.

  Example: http://localhost/cgi-bin/pkiclient.exe
  Command line option: -u

- **Verbose** – Verbose? Answer "yes" or "no".

  Command line option: -v

2.2 Actual enrollment

The actual enrollment is done with the following procedure:

2.2.1 STEP 1 – Gather information

- **CA server identification string**
  If your SCEP server requires you to use a specific identification string in the initial CA certificate access (step 3), write it down.

- **CA server http URL**
  You must know the *complete* url, with http:// and cgi-program path and everything.

  Example: http://pkiserver.company.com/cgi-bin/pkiclient.exe

- **CA naming policy**
  You need to know what kind of DN you request. Some may require you to use unstructuredName naming, some may require a CN with localityName, etc.

2.2.2 STEP 2 – Make certificate request and key

Before the enrollment can take place, sscep needs a private key file and the corresponding X.509 certificate request in PKCS#10 format. Edit the DN variables in the file mkrequest (it's a shell script) if you need. When ready, make the request:

```
$ mkrequest -ip 172.30.0.1
Generating RSA private key, 1024 bit long modulus
.............++++++
...++++++
e is 65537 (0x10001)
Using configuration from .4018client.cnf
```

This writes key and request named local.key and local.csr (you can change the "local" with variable PREFIX in mkrequest).

If the CA supports automatic enrollment, you may supply the password in cert request:

```
$ mkrequest -ip 172.30.0.1 password
```

2.2.3 STEP 3 – Get CA certificate

Configure the URL and CACertFile in configuration file (sscep.conf) and run the command:

```
$ ./sscep getca -f sscep.conf
./sscep: requesting CA certificate
./sscep: valid response from server
./sscep: MD5 fingerprint: 1D:3C:4C:DF:99:73:B8:FB:B4:EE:C4:56:A9:7C:37:A3
./sscep: CA certificate written as ca.crt
```

NOTE: it is very important to make sure that the CA certificate is really what you think it is. The security of the whole protocol depends on that!! This is why the fingerprint is printed on terminal – you should check that from your CA. You can check the fingerprint any time with command:

```
$ openssl x509 -in ca.crt -noout -fingerprint
```

If the CA sends a certificate chain, sscep writes all certificates in the order it founds them in reply and names them with an integer prefix (-number) appended to CACertFile.

```
$ ./sscep getca -f sscep.conf
./sscep: requesting CA certificate
./sscep: valid response from server

./sscep: found certificate with
  subject: /C=FI/O=klake.org/CN=klake.org VPN RA
  issuer: /C=FI/O=klake.org/CN=klake.org VPN CA
  usage: Digital Signature, Non Repudiation
  MD5 fingerprint: 7A:92:84:2A:6F:EE:28:14:F9:69:D8:9D:61:34:B5:67
./sscep: certificate written as ca.crt-0

./sscep: found certificate with
  subject: /C=FI/O=klake.org/CN=klake.org VPN CA
  issuer: /C=FI/O=klake.org/CN=klake.org VPN CA
  usage: Digital Signature, Non Repudiation, Certificate Sign, CRL Sign
  MD5 fingerprint: A5:CE:94:5C:96:77:94:E8:F5:31:AB:D5:31:18:1D:E1
./sscep: certificate written as ca.crt-1
```

SSCEP prints out issuer, subject, key usage and md5/sha1 fingerprint for each certificate it founds. This information might help you to decide what certificate to use. Some CAs may give you a three (or more) certificates, the root CA(s) plus different RA certificates for encryption and signing. If that's your case, you have to define encryption certificate with command line option (-e) or with conf file keyword EncCertFile. Probably it is the certificate with key usage "Key Encipherment".

Currently, SSCEP doesn't verify the CA/RA certificate chain. You can do it manually with OpenSSL:

```
$ openssl verify -CAfile ca.crt-1 ca.crt-0
ca.crt-0: OK
```

NOTE: In case of multiple CA/RA certificates, the actual CA (the one who signs your certificate) might not be the same as the CA/RA you are dealing with. Keep this in mind when installing the CA cert in /etc/isakmpd/ca.

2.2.4 STEP 4 – Make enrollment

You need to supply configuration file keys URL, CACertFile, PrivateKeyFile, LocalCertFile and CertReqFile. PrivateKeyFile is the key generated in step 2 (local.key), CertReqFile is the request (local.csr) and LocalCertFile is where the enrolled certificate will be written once ready.

If your CA/RA have different certificates for encryption and signing, you must also provide the encryption certificate (EncCertFile).

Normally, the enrollment looks like this:

```
$ ./sscep enroll -f sscep.conf
./sscep: sending certificate request
./sscep: valid response from server
./sscep: pkistatus: PENDING
./sscep: requesting certificate (\#1)
./sscep: valid response from server
./sscep: pkistatus: PENDING
./sscep: requesting certificate (\#2)
./sscep: valid response from server
./sscep: pkistatus: PENDING
....
./sscep: requesting certificate (\#NNN)
./sscep: valid response from server
./sscep: pkistatus: SUCCESS
./sscep: certificate written as ./local.crt
```

First message sent is PKCSReq, that's where your request goes. Then the CA writes request down and sends reply PENDING, indicating that the certificate is not signed yet. SSCEP polls periodically for the certificate by sending GetCertInitial messages until the CA returns SUCCESS. The polling interval can be adjusted with PollInterval, or command line option (-t). You can interrupt the process any time and start again using "sscep enroll ..". You should use the command line option (-R) when you continue the interrupted enrollment.

If the CA is configured for automatic enrollment (and your request includes the challenge password), it returns SUCCESS as a first reply. Otherwise, the enrollment requires manual signing and authentication (perhaps a phone call).

Newer SCEP draft versions allow to use the existing certificate (issued by the CA) to authenticate a renewal request. In this context, the SCEP request with the new public key is signed with the old certificate and key (instead of using a self-signed certificate created from the new key pair).

To use this feature, use the command line options -O and -K to specify the old certificate and private key (SignCertFile and SignCertKey in the configuration file).

The actual behaviour of the SCEP server depends on the CA policy and on the capabilities of the SCEP server (not all servers implement this feature, using the existing certificate with an older SCEP server may or may not work, depending on implementation).

Note: Newer versions of OpenCA (http://www.openca.info/) support an SCEP server that is capable of automatically approving SCEP requests signed with the already existing key pair.

2.2.5 STEP 5 – Use certificate

Install local.key, local.crt and ca.crt in the isakmpd default locations and you are ready to go! Default locations are:

- Private key /etc/isakmpd/private/local.key
- Certificate /etc/isakmpd/certs/local.crt
- CA certificate /etc/isakmpd/ca/ca.crt

And pay attention to CA certificate if your enrollment was done via RA server. "openssl verify -CAfile ca.crt local.crt" is your friend here.

2.2.6 STEP 6 – Check out revocation list (optional)

You need your enrolled certificate for this step.

```
$ ./sscep getcrl -f sscep.conf
./sscep: requesting crl
./sscep: valid response from server
./sscep: pkistatus: SUCCESS
./sscep: CRL written as ./crl.crl
```

3. Recommended literature

[1] Advantech B+B SmartWorx: **v2 Routers Configuration Manual** (MAN-0021-EN)
[2] Advantech B+B SmartWorx: **SmartFlex Configuration Manual** (MAN-0023-EN)
[3] Advantech B+B SmartWorx: **SmartMotion Configuration Manual** (MAN-0024-EN)
[4] Advantech B+B SmartWorx: **SmartStart Configuration Manual** (MAN-0022-EN)
[5] Advantech B+B SmartWorx: **ICR-3200 Configuration Manual** (MAN-0042-EN)

Product related documents can be obtained on *Engineering Portal* at https://ep.advantech-bb.cz/ address.