TransNexus

OSP Toolkit

Device Enrollment

Release 2.5.5

09 February 2002

OSP Toolkit

Device Enrollment

Release 2.5.5

09 February 2002

Document 0300-1241-0200

TransNexus
1140 Hammond Drive, Building E
Suite 5250
Atlanta, GA 30328
USA

Phone: +1 770 671 1888
Fax: +1 770 671 1188

E-mail: support@transnexus.com

Contents

Introduction

This guide documents a sample device enrollment application included with release 2.5.5 of the Open Settlement Protocol (OSP) Toolkit. The Toolkit, freely available under license from TransNexus, contains an implementation of the standard settlement protocol endorsed by the European Telecommunications Standards Institute (ETSI) and the International Multimedia Teleconferencing Consortium's Voice over IP (VoIP) Forum. The Toolkit also implements, as an option, extensions to the standard that allow access to enhanced services.

The OSP Toolkit contains eleven separate documents, including this one. The documents are:

- *Introduction*
- *Implementation Guide*
- *How to Build and Test the OSP Toolkit*
- *Errorcode List*
- *Programming Interface*
- *Cisco Interoperability Example*
- *Device Enrollment*
- *Internal Architecture*
- *Porting Guide*
- *Protocol Extensions*
- *ETSI Technical Specification TS 101 321*

The *OSP Toolkit Introduction* includes a "Document Roadmap" section that summarizes the various documents and their application.The device enrollment application documented in this guide is not part of the Toolkit library; it is, instead, a standalone application that can be invoked from a command line. The application handles the certificate request and retrieval functions that are part of enrolling a device in the TransNexus service. Although the application is specific to TransNexus service, it employs industry standard cryptographic and networking protocols. It may, therefore, be used, or easily modified for use, with other service providers.

Six main sections comprise this document. The first describes the command line interface to the `enroll` application, including its options, inputs, and outputs. The second section documents the certificates used for the TransNexus service. It is followed by a section that describes the message formats used by the application to communicate with a TransNexus enrollment server. The fourth section outlines the software components that are needed to build `enroll`, but are not part of the provided source code. The fifth section provides an example of an enrollment session using OpenSSL and the enroll utility. The document concludes with examples of keys and certificate formats.

Note that the `enroll` source code re-uses several components from the OSP Toolkit library, including network input and output, HTTP protocol processing, SSL interface functions, and cryptographic interface functions. Additional documentation on these components may be found in the *Internal Architecture* and the *Porting Guide* documents.

Command Line Interface

As delivered, `enroll` is strictly a command line application. All input from the user is taken from the command line, and all output is delivered to `stdout`. As noted below, however, TransNexus recommends that at least some information be stored in and retrieved from local, non-volatile storage. Of course, implementers are encouraged to enhance the applications as appropriate for their environment, including, for example, the use of a graphical user interface.

The complete syntax for `enroll` may be found by executing "`enroll`" (without any parameters) from the command line. The following figure shows that output.

```
> enroll

enroll -function { getcacert | request | retrieve } [params]

  getcacert get Certificate Authority information
  request   start enrollment by issuing an enrollment request
  retrieve  retrieve a certificate (if available)

  [getcacert params]
    -caurl <URL>         http:// URL for retrieving CA info
    -fprint <fprint>     optional CA certificate fingerprint
                         (in hexadecimal)

  [request and retrieve params]
    -cacert <cert>       base64-encoded authority certificate
    -certreq <pkcs10>    base64-encoded certificate request
    -customer <custID>   TransNexus-assigned customer number
    -device <devID>      TransNexus-assigned device id
    -nonce <nonce>       random value to increase security
    -password <pwd>      password for TransNexus services
    -username <username> username for TransNexus services
    -sslurl <URL>        https:// URL for enrollment server


exit level:
    0  -  enrollment succeeded
    1  -  enrollment request pending
    2  -  user error

>
```

As the figure illustrates, the `enroll` command includes a function designator, followed by any parameters. Each of these elements is described below.

```
{ getcacert | request | retrieve }
```

The `enroll` application actually includes support for three different functions: obtaining certificate authority information, requesting a certificate, and retrieving a certificate. The first parameter to the command indicates which of these three functions is desired.

The `getcacert` function begins an enrollment request by retrieving the enrollment server's CA certificate. As output, the CA certificate is reported in a base64 encoded format. If the `–fprint` option is specified on the command line, then the CA's SHA1 fingerprint must match the hexadecimal fingerprint provided. Otherwise, the CA's fingerprint is not validated and the CA certificate returned from the enrollment server is assumed to be valid.

The `request` function sends the base64-encoded certificate request to the enrollment server for approval. It takes an optional `nonce` parameter that specifies a random value for increasing the security of the SSL transmission. If the `nonce` parameter isn't specified, then it is generated by the enrollment client. The enrollment request's success will be reported on. The request status was either successful, pending further approval, or rejected. If the enrollment request was approved, then a base64-encoded X.509v3 certificate is displayed to the user. If the request is still pending approval or if it failed for any reason, then the user is notified.

The `retrieve` function checks on a pending enrollment request. It queries the enrollment server for the status of the request and reports the results as output. Its parameters are the same as those for the `request` function.

All three `enroll` functions report their output by writing it to `stdout`. In most practical cases the output should be placed in non-volatile, secure storage. If the certificate request is lost, then the enrollment server will not allow the user to retrieve the device's certificate. Otherwise, the user may contact TransNexus for the CA certificate or the device's certificate at any time, in case of device failure. However, if the device's key is compromised or believed to be compromised, then the user should contact TransNexus immediately to revoke the certificate; this will require having the challenge phrase that was contained in the certificate request.

CA Parameters

```
-caurl http://…
```

The `–caurl` parameter is only used for requesting a CA certificate from the enrollment server. The CA certificate will be used for validating the certificate that is returned from the enrollment server and for validating the certificate that the enrollment server uses in the SSL handshake. It will be reported as base64-encoded text as follows:

```
CA certificate received:
MIIB9DCCAV2gAwIBAgIRANs4gtN4kbWXlwvw8YsAjxMwDQYJKoZIhvcNAQEEB
QAwFTETMBEGA1UEChMKVHJhbnNOZXh1czAeFw05OTAzMTgwMDAwMDBaFw0wOT
AzMTgyMzU5NTlaMBUxEzARBgNVBAoTClRyYW5zTmV4dXMwgZ8wDQYJKoZIhvc
NAQEBBQADgY0AMIGJAoGBAKuR4hI8P+g96Go7ihjfdQ+3VjA01pIqNjaSch+e
WWzbBG+q+aISa0sQM53elNuxMudoCFN27J7H4v0LuStDj+wSQzWjP4lBOQUXr
```

```
y1tRi+qwRaK5VhlwybHejOByURb4Qex5myhEbNWAxOimgCBIB2Exf4k5FJjOM
Us795rlUpXAgMBAAGjRDBCMCIGA1UdEQQbMBmkFzAVMRMwEQYDVQQDEwpPbnN
pdGUyLTYyMA8GA1UdEwQIMAYBAf8CAQAwCwYDVR0PBAQDAgEGMA0GCSqGSIb3
DQEBBAUAA4GBAEgeTxN56ztf2bzu2Zx1a/e0IWexTeEbjCQNNEZaFOLhp50kV
B6oQQkX726Oiv0Gx4IJdTv3YHYc7BOi1pU0jWlPc/DVkhHdlQ/gDSNFgwAqJC
x2nmlfr9TuEtAUWAxd/PN38//yDyXWgx5PKyU9+pyLPgCoAC8D17wMGdh+oTS
m
```

This base64-encoded output will also be used as the CA certificate for the request and retrieve functions, so it is recommended that it be saved to non-volatile storage.

```
-fprint 11D74881E8F2C3E998B0A63E47D50ACDDCED1E18
```

The `–fprint` parameter is a 40 character sequence of hexadecimal digits. If it is entered on the command line for the `ca` function, then it must match the SHA-1 digest of the CA certificate returned. If the two fingerprints do not match, then the certificate request fails.If the fingerprint is not entered on the command line, then the CA certificate is not validated. Therefore, it is recommended that this value be entered.

Request and Retrieve Parameters

The following command line parameters are only required for the `request` and `retrieve` parameters.

```
-cacert AZaz09+/…
```

The `–cacert` parameter specifies the base64-encoded CA certificate that will be used for validating the enrollment server in the SSL handshake and for signing the device's requested certificate. If this parameter is not specified on the command line, then the user will be notified and `enroll` will exit. Note that this certificate must contain a self-signed root certificate in order for it to be used by `enroll,` and that the CA certificate's contents may be collected by means outside of using the `getcacert` function.

```
-certreq AZaz09+/…
```

The -`certreq` parameter provides a previously generated PKCS#10 certificate request to the `request` and `retrieve` operations. Such requests are generated outside the scope of the enrollment client. It is ignored by the `getcacert` operation. In all cases the value is input as a base64-encoded character string. The certificate request is expected to contain the version of the certificate request (which should be 0 at the time of this writing), the subject name of the sequence, the public key, and the challenge phrase. The enrollment client does not require a challenge phrase, but it is highly recommended. If the challenge phrase is present in the certificate request, then it must be located in the optional attributes list of the PKCS#10 request as a PKCS#9-7 attribute.

Because of the complexity of its value, implementers are strongly encouraged to place the certificate request in some form of non-volatile storage. If the certificate request is lost, then the user will not be able to retrieve the certificate for the device.

```
-customer 01234567…
```

The `–customerid` parameter is the TransNexus-assigned customer number for the operator. This number should be at least 8 digits long and is required for only the request and retrieve functions. If not present on the command line, `enroll` will prompt the user for its value. Note that the customer number includes built-in check digits. If the user provides an invalid customer number, `enroll` will fail.

```
-device 01234567…
```

The -device parameter is the TransNexus-assigned device number to the operator for the particular device being enrolled. This is decimal number that is at least eight characters long. If not present on the command line, `enroll` will notify the user and exit.

```
-certreq AZaz09+/…
```

The `–certreq` parameter specifies the base64-encoded request that is to be transmitted to the enrollment server. The certificate request should be a PKCS#10 CertificateRequest that contains the Version of the certificate request (0 at the time of this writing), the subject name of the device, the subjectPublicKeyInfo of the device, and the challenge phrase for the certificate. The challenge phrase should be contained in the optional attributes section of the certificate request, and its attribute type should be a PKCS#9-7 challengePhrase.

```
-username AZaz09…
```

The `-username` parameter is the ASCII user name corresponding to the operator's login account with TransNexus. If not present on the command line, `enroll` will notify the user and exit. Note that the username is never transmitted in unencrypted form across the IP network, which is part of the reason why the username and password are not transmitted when requesting the CA certificate in cleartext.

```
-password AZaz09…
```

The `-password` parameter is the ASCII password corresponding to the operator's login account with TransNexus. If not present on the command line, `enroll` notify the user and exit. Note that the password is never transmitted in unencrypted form across the IP network, which is part of the reason why the username and password are not transmitted when requesting the CA certificate in cleartext.

```
-sslurl https:...
```

The `-sslurl` parameter identifies the uniform resource locator value for the enrollment server. If it is omitted from the command line or if it is unreachable, then `enroll` will exit. Note that it is different from the `–caurl` in that it must specify a host and port for a service that is capable of handling SSL connectivity, while the `–caurl` parameter specifies a server that is incapable of handling SSL connections.

```
-nonce ABCDEF0123456789…
```

The `–nonce` parameter optionally specifies a value that increases the security of the messages transmitted to the enrollment server. Without this parameter, there is an increased risk that known-plaintext attacks can be launched on messages to the enrollment server. If this parameter is not specified on input, then it is generated by `enroll`.

Message Formats

All messages sent to the enrollment server are carried in HTTP (version 1.1) POST messages. All replies are returned in responses to the POST. Each POST request contains a series of ASCII *variable=value* pairs, encoded as given in RFC 1738. Any response also consists of variable/value pairs. These pairs are, for the most, identical to command line parameters described above, with the exception that non-alphanumeric characters are encoded as a "%" and their corresponding two hexadecimal digits (as specified in RFC 1738.)

The names of the parameters passed to the enrollment server are similar to the names used on the command line, with the main exception being that the function is specified by the "operation" parameter instead of "function".

The following subsections show complete example messages for all three functions.

Retrieving the CA Certificate

The following figure shows a sample CA certificate request message. In it, the device asks for the enrollment server's CA certificate in cleartext:

```
POST  HTTP/1.1
Host: osptestserver.transnexus.com
content-type: text/plain
Content-Length: 19
Connection: Keep-Alive
```

```
operation=getcacert
```

The response received from the enrollment server should look like:

```
HTTP/1.1 200 OK
Server: TNS/1.0
Connection: Keep-Alive
Content-Type: text/plain
Content-Length: 693

status=0&certificate=MIIB9DCCAV2gAwIBAgIRANs4gtN4kbWXlwvw8YsA
jxMwDQYJKoZIhvcNAQEEBQAwFTETMBEGA1UEChMKVHJhbnNOZXh1czAeFw05O
TAzMTgwMDAwMDBaFw0wOTAzMTgyMzU5NTlaMBUxEzARBgNVBAoTClRyYW5zTm
V4dXMwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKuR4hI8P+g96Go7ihj
fdQ+3VjA01pIqNjaSch+eWWzbBG+q+aISa0sQM53elNuxMudoCFN27J7H4v0L
uStDj+wSQzWjP4lBOQUXry1tRi+qwRaK5VhlwybHejOByURb4Qex5myhEbNWA
xOimgCBIB2Exf4k5FJjOMUs795rlUpXAgMBAAGjRDBCMCIGA1UdEQQbMBmkFz
AVMRMwEQYDVQQDEwpPbnNpdGUyLTYyMA8GA1UdEwQIMAYBAf8CAQAwCwYDVR0
PBAQDAgEGMA0GCSqGSIb3DQEBBAUAA4GBAEgeTxN56ztf2bzu2Zx1a/e0IWex
TeEbjCQNNEZaFOLhp50kVB6oQQkX726Oiv0Gx4IJdTv3YHYc7BOi1pU0jWlPc
/DVkhHdlQ/gDSNFgwAqJCx2nmlfr9TuEtAUWAxd/PN38//yDyXWgx5PKyU9+p
yLPgCoAC8D17wMGdh+oTSm
```

Sending an Enrollment Request and Retrieving the Certificate

Once the CA certificate is retrieved, the certificate request is encrypted and transmitted to the enrollment server for approval. The initial request (before it is encrypted) looks like this:

```
POST  HTTP/1.1
Host: enroll.transnexus.com
content-type: text/html
Content-Length: 714
Connection: Keep-Alive

operation=request&nonce=1502767911902931&username=mcmanus&pas
sword=01234567&device=134217728&customer=0&request=MIIBtTCCAR
4CAQAwWzELMAkGA1UEBhMCVVMxEDAOBgNVBAgTB0dlb3JnaWExGDAWBgNVBAo
TD1RyYW5zTmV4dXMsIExMQzEgMB4GA1UEAxMXdGVzdHRlcDQudHJhbnNuZXh1
cy5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALhYeWbF8HrVIRVMW
4p2H2DZhs9tEisHe1ynyUEIcC4n9CLW104HW0zeSzNMtYBQrqJ6qZMhc0RKZ%
2BMQA9E1S9hvN8TLo4KDBPXmQWEQg6R9f3TokpIhOJ4bOwpj9WeXAiyNyTq7h
TgQdtPYN65xq92t5CkHpWBWEya9v2Ux9I27AgMBAAGgGjAYBgkqhkiG9w0BCQ
cxCxMJcGFzc3dvcmQAMA0GCSqGSIb3DQEBBAUAA4GBAFC7sCjCbmVgUYenJR8
XgMtLilQFSSq4YJ9BcmiYsZZ6KOxFxNgEf84wyRscdrP9LV9EhQM%2BS3gEAE
w%2FLxCRHGGgyS1%2FYpNmavs51thGeplH%2BAFW%2Blnds9CYUwyKx%2F8ve
FJFC6y6pYhD7RyZxyKNnzBhgxAxU3rUgr3Cm78RbT1G
```

The retrieve function only differs in the “operation” parameter, in which the “request” value is replaced by “retrieve”. Otherwise, all parameters have the same names and values.

If the enrollment request is pending further approval, then the enrollment server is only required to send the status of the certificate request. It may send a nonce along with the response, but this value is not guaranteed. The response should look like this:

```
HTTP/1.1 200 OK
content-type: text/plain
content-length: 31

status=1&nonce=A1F0765F71C9E6AD
```

If the enrollment request has been processed and accepted by the server, it will return a response such as the following. Note that a status of 0 indicates that the certificate is now ready for retrieval.

```
HTTP/1.1 200 OK
content-type: text/plain
content-length: 694

status=0&cert=MIICfjCCAeegAwIBAgIQARAm+prL9zmocfkRWNN0fjANBgk
qhkiG9w0BAQQFADAV…
```

Additional Components

In addition to the included source code, a complete implementation of the `enroll` application requires two additional software components. Those components are necessary to provide Secure Socket Layer (SSL) services and cryptographic algorithms. TransNexus does not provide that software as part of the application. These same services are required for the Open Settlement Protocol Toolkit library, and both the library and the `enroll` application have a common interface into each component. The following subsections both components, and they identify potential sources.

Secure Sockets Layer

Secure Sockets Layer (SSL) is a protocol that provides a secure channel for communications between clients and servers. SSL defines mechanisms to authenticate both parties in a communication and to encrypt and decrypt their exchanges. The `enroll` application requires SSL version 3.0, and an SSL implementation is a required addition to the Toolkit. At the time this document was written, TransNexus was aware of the following sources for SSL implementations:

- **Baltimore**. Baltimore offers both C and Java implementations of SSL in its C/SSL and J/SSL Developer Toolkits. Information is available at http://www.baltimore.com.

- **Certicom**. Certicom Corporation offers a commercial implementation of SSL known as SSL Plus. Licensing information is available at http://www.certicom.com.

- **Eric Young**. Eric Young has developed an implementation of SSL known as SSLeay that, in most circumstances, may be licensed free of charge. As of this writing, SSLeay was available at ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL.

- **Microsoft Corporation**. Microsoft includes an SSL implementation as part of its WinInet control. As of this writing, more information on this is available from Microsoft at http://msdn.microsoft.com/workshop/networking/wininet/wininet.asp.

  In addition, Microsoft has unofficially indicated that it will soon make available a public interface to SCHANNEL.DLL, the SSL implementation used by Microsoft applications such as IIS.

- **OpenSSL**. OpenSSL is an open source implementation of the SSL protocol. More information is available at http://www.openssl.org.

- **RSA Data Security**. RSA Data Security offers a commercial implementation of SSL known as SSL-C. Licensing information is available from RSA at http://www.rsa.com.

- **Spyrus/Terisa**. Spyrus/Terisa offers commercial implementations of SSL (and HTTP) in its TLS Gold SSL Toolkit, and Device SSL Toolkit for embedded systems. For details, refer to http://www.spyrus.com.

As delivered, the TransNexus `enroll` application supports the programming interface of OpenSSL. The Toolkit's SSL interface is modularized; however, to permit easy porting to other SSL libraries.

Cryptographic Algorithms

The `enroll` application relies on special cryptographic algorithms for message digesting, digital signature generation and verification, and encryption and decryption. At the time this document was written, sources for those cryptographic algorithms include the following.

- **Baltimore**. Baltimore offers cryptographic algorithms in its J/Crypto and Crypto Systems Toolkit products, in Java and C respectively. Licensing information is available at http://www.baltimore.com.

- **Microsoft Corporation**. Microsoft includes a complete implementation of the required cryptographic algorithms as part of Windows 98, Windows NT 4.0 (beginning with SP3) and Internet Explorer 4.0 for Windows NT, Windows 98, and Windows 95. These algorithms are exposed by version 2.0 of the CryptoAPI interface, which is described at http://www.microsoft.com/security.

- **RSA Data Security**. RSA Data Security provides a reference implementation of some cryptographic algorithms (RSARef) and a commercial software library of cryptographic algorithms (BSAFE). Availability and licensing information for these products is available at http://www.rsa.com.

- **OpenSSL**. OpenSSL is an open source implementation of the SSL protocol. More information is available at http://www.openssl.org.

As delivered, the TransNexus `enroll` application supports the BSAFE version 3.0 programming interface. Cryptographic interfaces are modularized, however, to permit easy porting to other libraries.

Example TEP Enrollment Session

This section provides an example of the enroll application using the TransNexus OSP Test Server.

In order to access the TransNexus OSP Test Server for free, your device(s) must be added to the test server configuration. This is accomplished by sending an e-mail to support@transnexus.com with the following information:

- The name of at least one person we can contact from your company and his/her e-mail address.

- If the machine running test_app and/or using the OSP Toolkit is outside all firewalls, then we need the IP address of this machine.

- If the machine running test_app and/or using the OSP Toolkit is behind at least one firewall, then we need the IP address of the OUTERMOST firewall which the OSP messages will travel through (our test server must be able to see the machine with that address).

Within one business day of receiving your e-mail, we will send you an e-mail reply indicating that your device is configured with the TransNexus OSP Test Server. Included in this e-mail will be certificates to use with the TransNexus OSP Test Server. These certificates (private key, local certificate, and self-signed CA certificate) should be placed in the "bin" directory within the OSP Toolkit directory. If you wish to use or generate your own certificates, see the next section for details. For first time users of the OSP Toolkit, we recommend starting with the default configuration and use the certificates we give you, then working toward more complexity.

NOTE: The OSP Test Server is configured only for sending and receiving non-SSL messages, and issuing signed tokens.

Using the enroll program on Unix and Windows NT

Go to the bin directory within the OSP Toolkit directory.

```
cd bin
```

1) Generate a public and private key pair as well as a certificate request.

Execute the following command at the prompt:

```
openssl req -outform PEM -nodes -newkey rsa:512 -md5 -new
      -out certreq.b64 -keyform PEM -keyout pkey.b64
```

NOTE: For Windows NT, you must append the following text to the above command:

```
-config <openssl_path>\apps\openssl.cnf
```

where <openssl_path> is the location of your OpenSSL directory. This is the same openssl.cnf file you edited earlier for Windows NT.

NOTE: On Windows NT, even if the configuration file name might be "openssl" instead of openssl.cnf, you MUST use "openssl.cnf" with the "-config" option.

If all goes well, the following text will be displayed. The gray boxes indicate required input.

```
Using configuration from /usr/local/bin/openssl.cnf
Loading 'screen' into random state - done
Generating a 512 bit RSA private key
....+++++
...........+++++
writing new private key to 'pkey.b64'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

The two files generated are base64 encoded files: the certificate request "certreq.b64" and the private key "pkey.b64". The public key is contained in the certificate request.

2) Isolate the encoded certificates by manually editing the files:

A) Edit the certreq.b64 file within a text editor (e.g. vi, emacs, WordPad, NotePad, etc.).

B) Within certreq.b64, remove everything above and including the text

```
-----BEGIN CERTIFICATE REQUEST-----
```

and everything below and including the text

```
-----END CERTIFICATE REQUEST-----
```

Also, remove all newline characters (in other words, put all the text on one line).

C) Save the changes using the same filename (replacing the original file).

D) Edit the pkey.b64 file within a text editor (e.g. vi, emacs, WordPad, NotePad, etc.).

E) Within pkey.b64, remove everything above and including the text

```
-----BEGIN RSA PRIVATE KEY-----
```

and everything below and including the text

```
-----END RSA PRIVATE KEY-----
```

Also, remove all newline characters (in other words, put all the text on one line).

F) Save the changes using the same filename (replacing the original).

Check to make sure the certificates are in the proper format. The contents of the files should be comprised only of the characters from "a" through "z", "A" through "Z", "0" through "9", "+", and "/" (sometimes with "=" at the end of the file) all on on line. For example:

```
MIIBy9b68yjCAuG2aNrPkVSpBnTRQvcRS9yGdS7/G1v+hhjBP1DVfAXl4XwbJ
xYkMCyIKtgwIDAQABmVBAoTCU9TUFNlcnZlcjAeFw0wMDEyMDQxMzU5MThaFw
0wMTEyMDUxMzU5MThaMeiC9XIgZ+17/WzwDQYJKoZIhvc+AQE/BQAwJzERMA8
GA1UEAxMId2lsbGlhbXMxEjAQBgN70GCSqFSIf3DQEBAQUAt4GNAeCKiQKBgQ
C40wqtMdP4MrQrrCLNan2xE4NV0t49w58m4EuQMm1C3Fjnd1BilmHnJYWdsKm
oBpAGf9o402/vLGrSePwftx3bFZ94aQKLcA15OtIUhwKhYrjwiYsmk85YXHD/
8Vl6RjOxGQxCzAJBgNVBAYTAlVtRAwDgYDooQIEwdHZW9yZ2lhqRAwDgHDV2Q
sEwdBdGxhbnRhMqlwHwYDVQQKExhJbnRlcm5ldCBXaWR2EwZdqQcTuZaJV/Bn
aXRzIFB0eSBcdGQxDjAMBgNVBvMTBiRldjAxMIofMA0GPSqGrIb3DQEPBAUAA
0EAvvImOBao8PjgYSVmsvRy00bJoGHI7znEqEC2+PLoUIH6PzNEJlzYqs7s8r
6Bl==
```

Do this step again if unsuccessful.

3) Get the Certificate Authority (CA) certificate from the server:

Execute the following command at the prompt:

```
enroll -function getcacert -caurl http://osptestserver.transnexus.com:80/tep
```

If you are using a different OSP server that supports TEP enrollment, you would put the server name in place of osptestserver.transnexus.com.

If there are no errors, copy the resulting certificate into a new file with the filename cacert.b64. From this file, isolate the resulting encoded certificate by manually editing the files:

A) Edit the cacert.b64 file within a text editor (e.g. vi, emacs, WordPad, NotePad, etc.).

B) Within cacert.b64, remove everything above and including the text

```
CA certificate received
```

and everything below and including the text

```
Press any key to continue.
```

Also, remove all newline characters (in other words, put all the text on one line).

C) Save the changes using the same filename (replacing the original file).

Check to make sure the certificate is in the proper format. The contents of the file should be comprised only of the characters from "a" through "z", "A" through "Z", "0" through "9", "+", and "/" all on on line (see the earlier example). Do this step again if unsuccessful.

4a) Enroll the device and receive a local certificate using UNIX.

For Unix, execute the following command at the prompt:

```
enroll -function request -username trans -password nexus
    -customer 1000 -device 1000 -cacert `cat cacert.b64`
    -certreq `cat certreq.b64` -sslurl
    https://osptestserver.transnexus.com:443/tep
```

If you are using a different OSP server that supports TEP enrollment, put the server name in place of osptestserver.transnexus.com.

NOTE: The ` is a backward quote. The command will not work otherwise.

When you run this command, you may see the following errors which may disregarded.

```
ERROR 16030: X509 CertInfo context is null pointer
      File: ospx509.c  line: 634
ERROR 12080: Unable to get Local Certificate
      File: ospopenssl.c  line: 491
```

If there are no errors, copy the resulting certificate into a file and save it as localcert.b64. With this file, isolate the resulting encoded certificate by manually editing the files:

1) Edit the localcert.b64 file within a text editor (e.g. vi, emacs, NotePad, WordPad, etc.).

2) Within localcert.b64, remove everything above and including the text

   ```
   The certificate request was successful.
   ```

   and everything below and including the text

   ```
   Press any key to continue.
   ```

   Also, remove all newline characters (in other words, put all the text on one line).

3) Save the changes using the same filename (replacing the original).

Check to make sure the certificate is in the proper format. The contents of the file should be comprised only of the characters from "a" through "z", "A" through "Z", "0" through "9", "+", and "/" all on on line. Do this step again if unsuccessful.

4b) Enroll the device and receive a local certificate using NT.

For NT, create a new text file (in NotePad, WordPad, etc.) and copy this command into this file (the command all on one line). Replace "`cat cacert.b64`" with the CONTENTS of the cacert.b64 file, and replace "`cat certreq.b64`" with the CONTENTS of the certreq.b64 file. Save this file as "enroll.bat"; this becomes a DOS batch file. For example, the file might look like the following:

```
enroll -function request -username trans -password nexus
-customer 1000 -device 1000 -cacert MIIBy9b68yjCAuG2aNrP
kVSpBnTRQvcRS9yGdS7/G1+hhjBP1DVfAXl4XwbJxYkMCyIKtgwIDAQA
BmVBAoTCU9TUFNlcnZlcjAeFw0wMDEyMDQxMzU5MThaFw0wMTEyMDUxM
zU5MThaMeiC9XIgZ+17/WzwDQYJKoZIhvc+AQE/BQAwJzERMA8GA1UEA
xMId2lsbGlhbXMxEjAQBgN70GCSqFSIf3DQEBAQUAt4GNAeCKiQKBgQC
40wqtMdP4MrQrrCLNan2xE4NV0t49w58m4EuQMm1C3Fjnd1BilmHnJYW
dsKmoBpAGf9o402/vLGrSePwftx3bFZ94aQKLcA15OtIUhwKhYrjwiYs
mk85YXHD/8Vl6RjOxGQxCzAJBgNVBAYTAlVtRAwDgYDooQIEwdHZW9yZ
2lhqRAwDgHDV2QsEwdBdGxhbnRhMqlwHwYDVQQKExhJbnRlcm5ldCBXa
-certreq Nan2xE4NV0t49w58m4EuQMm1C3Fjnd1BilmHnJYWdsKmoBp
AGf9o402/vLGrSePwftx3bFZ94aQKLcA15OtIUhwKhYrjwiYsmk85YXH
D/8Vl6RjOxGQxCzAJBgNVBAYTAlVtRAwDgYDooQIEwdHZW9yZ2lhqRAw
DgHDV2QsEwdBdGxhbnRhMqlwHwYDVQQKExhJbnRlcm5ldCBXaWR2EwZd
qQcTuZaJV/BnaXRzIFB0eSBcdGQxDjAMBgNVBvMTBiRldjAxMIofMA0G
PSqGrIb3DQEPBAUAA0EAvvImOBao8PjgYSVmsvRy00bJoGHI7znEqEC2
+PLoUIH6PzNEJlzYqs7s8r6Bl== -sslurl https://osptestserve
r.transnexus.com:443/tep
```

Run it by typing in the name of the batch file at the prompt.

If there are no errors, copy the resulting certificate into a file and save it as localcert.b64. With this file, isolate the resulting encoded certificate by manually editing the files:

1) Edit the localcert.b64 file within a text editor (e.g. vi, emacs, NotePad, WordPad, etc.).

2) Within localcert.b64, remove everything above and including the text

```
The certificate request was successful.
```

and everything below and including the text

```
Press any key to continue.
```

Also, remove all newline characters (in other words, put all the text on one line).

3) Save the changes using the same filename (replacing the original).

Check to make sure the certificate is in the proper format. The contents of the file should be comprised only of the characters from "a" through "z", "A" through "Z", "0" through "9", "+", and "/" all on on line. Do this step again if unsuccessful.

5) Convert the files from base64 to binary

Execute the following commands at the command prompt:

```
openssl enc -base64 -in cacert.b64 -out cert_00001.dat -d -a -A

openssl enc -base64 -in pkey.b64 -out pkey_00001.dat -d -a -A

openssl enc -base64 -in localcert.b64 -out localcert_00001.dat -d -a -A
```

After following the instructions for enrolling a TEP device, three files should exist: a private key, a local certificate, and a CA certificate. In order to use these files with test_app (a client simulator included with the TransNexus OSP Toolkit), they must be located in the bin directory within the OSP Toolkit directory.

File Type	File Name for use with test_app
Private Key	pkey_00001.dat
Local Certificate	localcert_00001.dat
CA Certificate	cert_00001.dat

Example Keys and Certificates

This section contains sample keys and certificates used with the OSP Test Server. All of these examples were obtained by using a freeware program called "dumpasn1" on certificates AFTER they were base64 decoded (the examples earlier in this document are all base64 encoded).

Certificate Authority (CA) Certificate:

This certificate contains the public key of the OSP Test Server. It is the result after executing an "enroll -function getcacert..." command. The following is a breakdown of the data that is returned:

```
  0 30  309: SEQUENCE {
  4 30  224:   SEQUENCE {
  7 02    1:     INTEGER 1
 10 30   13:     SEQUENCE {
 12 06    9:       OBJECT IDENTIFIER
           :         md5withRSAEncryption (1 2 840 113549 1 1 4)
 23 05    0:       NULL
           :       }
 25 30   38:     SEQUENCE {
 27 31   16:       SET {
 29 30   14:         SEQUENCE {
 31 06    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
 36 13    7:           PrintableString 'testosp'
           :           }
           :         }
 45 31   18:       SET {
 47 30   16:         SEQUENCE {
 49 06    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
 54 13    9:           PrintableString 'OSPServer'
           :           }
           :         }
           :       }
 65 30   30:     SEQUENCE {
 67 17   13:       UTCTime '010213212341Z'
 82 17   13:       UTCTime '030214212341Z'
           :       }
 97 30   38:     SEQUENCE {
 99 31   16:       SET {
101 30   14:         SEQUENCE {
103 06    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
108 13    7:           PrintableString 'testosp'
           :           }
           :         }
117 31   18:       SET {
119 30   16:         SEQUENCE {
121 06    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
126 13    9:           PrintableString 'OSPServer'
           :           }
           :         }
           :       }
137 30   92:     SEQUENCE {
139 30   13:       SEQUENCE {
141 06    9:         OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
152 05    0:         NULL
           :         }
154 03   75:       BIT STRING 0 unused bits
           :         30 48 02 41 00 F1 9E 1B 05 78 D4 42 21 5F 08 C4
           :         0C 52 D1 5D 08 43 12 BE 74 39 44 0F AB E7 F9 E5
           :         5C 10 77 44 D0 36 BD 34 E8 13 F6 F7 E5 22 35 D4
           :         56 DD CD 8C 47 7F 59 70 97 42 D4 28 3A F0 C1 3D
           :         72 07 AA D2 1F 02 03 01 00 01
           :       }
           :     }
231 30   13:   SEQUENCE {
233 06    9:     OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
244 05    0:     NULL
           :     }
246 03   65:   BIT STRING 0 unused bits
           :     D0 D4 E0 7C 30 C0 AF 93 44 1D 96 4E BD BE 74 DD
           :     69 7A E7 13 AF 50 27 FC 89 6A 51 A3 E4 74 FE F7
```

```
            :     A8 CB 5B 2A 91 A3 2C BA 75 4B 09 24 E4 A1 3F 7C
            :     2F C9 63 12 76 52 F4 B4 22 B7 94 E1 C0 13 1D A2
            :   }

0 warnings, 0 errors.
```

Local Certificate:

This certificate is returned by the OSP Test Server. It is the result after executing an "enroll -function request..." command. The following is a breakdown of the data that is returned:

```
   0 30  456: SEQUENCE {
   4 30  370:   SEQUENCE {
   8 02   17:     INTEGER
            :       00 AB A3 29 15 78 D3 AD A2 8F 1A E7 C2 B6 26 1C
            :       C1
  27 30   13:     SEQUENCE {
  29 06    9:       OBJECT IDENTIFIER
            :         md5withRSAEncryption (1 2 840 113549 1 1 4)
  40 05    0:       NULL
            :       }
  42 30   38:     SEQUENCE {
  44 31   16:       SET {
  46 30   14:         SEQUENCE {
  48 06    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
  53 13    7:           PrintableString 'testosp'
            :           }
            :         }
  62 31   18:       SET {
  64 30   16:         SEQUENCE {
  66 06    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
  71 13    9:           PrintableString 'OSPServer'
            :           }
            :         }
            :       }
  82 30   30:     SEQUENCE {
  84 17   13:       UTCTime '010213213213Z'
  99 17   13:       UTCTime '020214213213Z'
            :       }
 114 30  100:     SEQUENCE {
 116 31   11:       SET {
 118 30    9:         SEQUENCE {
 120 06    3:           OBJECT IDENTIFIER countryName (2 5 4 6)
 125 13    2:           PrintableString 'US'
            :           }
            :         }
 129 31   16:       SET {
 131 30   14:         SEQUENCE {
 133 06    3:           OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
 138 13    7:           PrintableString 'Georgia'
            :           }
            :         }
 147 31   16:       SET {
 149 30   14:         SEQUENCE {
 151 06    3:           OBJECT IDENTIFIER localityName (2 5 4 7)
 156 13    7:           PrintableString 'Atlanta'
            :           }
            :         }
 165 31   33:       SET {
 167 30   31:         SEQUENCE {
 169 06    3:           OBJECT IDENTIFIER organizationName (2 5 4 10)
 174 13   24:           PrintableString 'Internet Widgits Pty Ltd'
            :           }
            :         }
```

```
200 31   14:       SET {
202 30   12:         SEQUENCE {
204 06    3:           OBJECT IDENTIFIER commonName (2 5 4 3)
209 13    5:           PrintableString 'osp01'
           :           }
           :         }
           :       }
216 30  159:     SEQUENCE {
219 30   13:       SEQUENCE {
221 06    9:        OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
232 05    0:         NULL
           :         }
234 03  141:       BIT STRING 0 unused bits
           :         30 81 56 02 81 71 00 B8 D3 0A AD 14 D3 F8 30 34
           :         2B AC 36 CD 6A 7D B1 21 83 55 D2 DE 3D C3 9F 26
           :         E0 4B 92 32 6D 42 DC 63 E7 77 50 62 96 35 E7 25
           :         87 ED C4 33 C5 67 DE 38 40 A2 DC 03 5E 4E B4 86
           :         23 C0 A8 58 AE 3C 22 62 C9 A4 F3 96 17 1D 9F FC
           :         56 5E 91 8C EC 7D 6F FA 32 A4 20 2E 1B C7 8D AC
           :         F9 15 4A 90 67 4D 14 2F 71 14 BD C8 67 52 EF F1
           :         B5 BF E2 61 8C 18 F5 0D 57 C0 5E 5E 03 C1 B2 71
           :         9A 4D 02 C8 82 AD 83 02 03 01 00 01
           :       }
           :     }
378 30   13:   SEQUENCE {
380 06    9:    OBJECT IDENTIFIER md5withRSAEncryption (1 2 840 113549 1 1 4)
391 05    0:     NULL
           :     }
393 03   65:   BIT STRING 0 unused bits
           :     DD 28 DB EE E3 03 82 EC 43 F6 B2 CD 4C B5 7D E6
           :     BD 04 77 92 9A 26 01 8B 86 CA C9 FA A9 23 A4 D7
           :     9F 75 3C D1 2C 0A 6D 9A A5 3A 34 50 FA 76 15 C6
           :     68 34 92 A5 D9 9B 84 87 3E 4A 4F 8F 2A AD 29 AE
           :   }
```

```
0 warnings, 0 errors.
```

Private Key:

Private keys are not included in the TEP protocol, but if you are using test_app (a client simulator included with the OSP Toolkit), the private key must be in the following format. This private key was generated using OpenSSL.

```
  0 30  606: SEQUENCE {
  4 02    1:   INTEGER 0
  7 02  129:   INTEGER
           :     00 CE 77 EE F4 70 84 5B DE FA E7 04 36 9B C7 6E
           :     EE 70 91 96 19 54 51 C9 F0 41 D5 9C 08 6E A4 6A
           :     F7 E6 27 D2 AA B8 96 E8 49 DD 39 FD 19 35 8A 90
           :     DA 0D C3 FC FC 68 B8 AA C8 5C BC 8E A9 94 3E 31
           :     27 92 77 E1 F2 37 84 D7 FA DD 6F 7A 8E 9F 79 4B
           :     89 E1 11 D3 6F D9 CE CD DF 9F E1 43 97 55 F9 84
           :     11 A5 F9 D0 D8 5B 1A FB FD F1 F0 55 8E 70 64 09
           :     B1 C1 DD C8 15 40 A1 B5 9B 82 7B 6F 45 CF 51 C7
           :     03
139 02    3:   INTEGER 65537
144 02  129:   INTEGER
           :     00 9A A1 C6 65 CA 12 B4 95 8E 2D 81 C3 F7 DE 7D
           :     36 FD 62 A0 01 9A AA 18 5F 1A 89 F5 63 28 A0 E7
           :     CB 69 AB 8A 54 11 17 D9 11 E2 F7 D4 8E 18 BD 01
           :     2F 9C 7A 8A 5B D3 71 BD B8 5F 64 C1 C4 E5 6D E4
           :     E6 ED 3A 11 60 4F 95 E9 D7 F4 C4 D0 39 1D 58 3A
           :     E4 A8 1E B2 5C 87 6D 63 F0 9D D3 FC 7B AA 79 9B
           :     23 FF 01 1B 4E 68 B2 21 7D F6 92 F9 FB 38 1F D0
           :     0E 96 33 12 56 9E 30 7D DA C4 A8 43 CA DD 47 B2
           :     47
```

```
 276 02   65:   INTEGER
            :     00 FB 87 D9 94 58 F9 D5 83 91 20 5B 9E DA B2 44
            :     4A 2A 70 CC AB 2A F8 C9 6B BB 6C A0 4F BD 91 FC
            :     79 BD 06 D1 80 F7 78 D0 E2 9A EA B3 60 53 D2 7E
            :     88 AA 12 EF 7E C3 E3 1C 0C 26 A3 8C D8 E2 9E FB
            :     F3
 343 02   65:   INTEGER
            :     00 D1 64 4B 82 D1 33 1B 7A 5E EC 27 45 1A BC 31
            :     DA 90 80 84 38 00 6A A5 A9 9E 02 C8 AC 95 0E EE
            :     94 89 D8 C5 53 D4 03 66 CB 8E DF 34 1F 95 38 3E
            :     34 18 CA A4 FD A8 4C 33 20 91 9D 4E 2D DE 8D 1C
            :     B1
 410 02   65:   INTEGER
            :     00 8B 35 69 33 A4 C4 BC 15 08 A9 5C B8 39 F8 71
            :     3A A4 4E B0 14 5F 50 E4 91 C5 AB EE D5 BE DE 9D
            :     DE 01 FC 3A 36 E7 DB A3 17 48 AC BA 8B 08 4A 0D
            :     E4 8A 54 7B DF 56 5E 78 FE 75 02 4D 5F 35 B5 CC
            :     4F
 477 02   64:   INTEGER
            :     6B F8 1C 5B A3 05 12 58 AC 13 77 18 51 F5 D2 7B
            :     11 E1 C2 CB E6 37 9C 9E 7E 84 8E F7 36 91 11 82
            :     74 25 DB 72 07 33 71 D0 EA A5 BB 19 C2 1D 7F 78
            :     9B 83 FA 49 79 5C 79 4C 84 00 29 03 D3 A6 DC B1
 543 02   65:   INTEGER
            :     00 86 C9 4D A9 37 5A C9 61 5C 5A BB F4 FB 0B F8
            :     FE 9D 56 E2 73 DE 80 A1 71 7F 02 A6 57 BA 70 ED
            :     B7 B5 9F 30 EE B5 18 FC 64 8A E5 2E AD 81 B5 C2
            :     96 79 82 6E 5E 0B 76 42 0D 68 B7 38 04 62 84 F6
            :     60
            :   }

0 warnings, 0 errors.
```