Progress®



Manage OpenEdge Keys and Certificates

OpenEdge®

1

Manage OpenEdge Keys and Certificates

All OpenEdge server and client components that implement secure HTTP (HTTPS) or Secure Socket Layer (SSL) connections require access to private keys and digital certificates to negotiate these connections and to enable them to function securely.

For all OpenEdge components, OpenEdge provides utilities that allow you to install and manage keys and digital certificates (in keystores) so that the components can access them. For Open Clients, clients of OpenEdge Web services, OpenEdge provides utilities for some clients or it relies on utilities provided by the client platform to manage the required certificate stores.

This section describes how to use the OpenEdge utilities.

An SSL server requires access to a private key and a digital (public-key) certificate to authorize the identity of the server. Clients require access to public-key certificates that allow them to authenticate the servers that they access. Both servers and clients must obtain their keys and certificates from a trusted source, typically a Certificate Authority (CA). For more information on keys, certificates, and CAs, see *Introduction to Security and Auditing*.

For details, see the following topics:

- Manage keystores for OpenEdge servers
- Manage certificate stores for OpenEdge clients and servers

Manage keystores for OpenEdge servers

You can manage the private keys and the corresponding digital certificates for OpenEdge servers that support SSL connections using a **keystore**, which is located in the *OpenEdge-Install-Dir*\keys directory. Each SSL server requires at least one **keystore entry** that contains a single private key and corresponding digital (public-key) certificate, also known as a public/private key pair. With this keystore entry, you can configure any supported OpenEdge server to enable and manage SSL connection from clients. For more information on the OpenEdge servers that support SSL server configuration, see the sections on the OpenEdge-supported SSL server components described in *Introduction to Security and Auditing*.

If you require only data encryption and do not need to verify the identity of SSL servers (typically, for intranet configurations only), OpenEdge comes installed with a default keystore entry. This default entry contains a common private key and digital certificate pair that you can use without any further management beyond enabling SSL connections on OpenEdge clients and servers. For more information on the default SSL server identity, see the sections on SSL in *Introduction to Security and Auditing*.

However, to establish a trusted OpenEdge SSL server identity suitable for use on the Internet or a more secure intranet, you must complete several steps using the functions of the pkiutil and certutil command-line utilities installed with OpenEdge.

Note: Before you run an OpenEdge command-line utility, set the DLC environment variable to the *OpenEdge-Install-dir* pathname and set the WRKDIR environment variable to your working directory. For an example, see the *OpenEdge-install-dir*/bin/pkiutil shell script on UNIX or the *OpenEdge-install-dir*\bin\pkiutil.bat file in Windows.

Running the command-line utility in a Proenvcommand window properly sets DLC and WRKDIR for you.

Establish a trusted SSL server identity

There are several steps required to establish a trusted identity for any OpenEdge SSL server using the pkiutil command-line utility.

Caution:

While the default_server keystore entry provided by the Progress Server Certificate Authority also uses a default password ("password"), you must password-protect any private key store entries that you create from a public-key certificate issued by a trusted external CA. The secrecy of your password is critical to using this key store entry for authenticating a server.

To establish and maintain a trusted SSL server identity using the **pkiutil** utility:

1. Use the -newreq operation to generate a proposed public/private key pair together with a digital certificate request that is suitable for sending to any CA for authorization. You must provide a password to secure this certificate request. You must later provide this password to any OpenEdge server which you want to access this key store entry for securing SSL connections to it. See Supply a keystore entry password to an OpenEdge server on page 5.
2. Use e-mail (or any method required by the CA) to submit the certificate request to the trusted CA, who returns a digital certificate that is signed by the CA. This process authenticates any server providing access to the private key.
3. Use the -import operation to import the CA-signed digital certificate and store it together with the associated private key as an entry in the keystore.

4. Use the `-display` or `-list` operations to review an individual digital certificate file or any keystore entries for important digital certificate information, such as expiration dates.
5. Use the `-remove` operation to remove any unused or expired keystore entries that you specify and retain them in a backup area of the keystore.

Note: For an overview of the `pkiutil` command-line utility, see Use pkiutil to manage an OpenEdge keystore on page 5.

Supply a keystore entry password to an OpenEdge server

When you configure an OpenEdge server to access a keystore entry, you must provide it with the same password that you used to create the keystore entry. If you configure the server using OpenEdge Management, you can enter this password directly in the fields provided. However, if you configure the server by manually editing the `ubroker.properties` file for that server or specifying the password on a command line or in a startup script (as required when starting a database server for the OpenEdge RDBMS), you must provide an encrypted value for the password. This will protect the password itself from being easily discovered. OpenEdge provides the `genpassword` command-line utility for obtaining a password's encrypted value. For more information, see Use genpassword to obtain a keystore password-encrypted value on page 7.

Use pkiutil to manage an OpenEdge keystore

The `pkiutil` command-line utility provides all the operations necessary to create and manage keystore entries for OpenEdge SSL servers (see Manage keystores for OpenEdge servers on page 4). This utility manages all input and output for the keystore in the *OpenEdge-Install-Dir*`\keys` directory. For more information on the structure of this directory, see About the keystore content on page 6.The `pkiutil` utility has the following general command-line syntax:

```
pkiutil [options] function arguments
```

options

Change the type of information and defaults for different functions (*function*) of the utility.

function arguments

One of the following functions (*function*) and the objects they affect (*arguments*):

- ***-newreq alias*** — Generates a new public/private key pair and a corresponding certificate signing request (suitable for submission to a CA), stored under the alias name specified by *alias*
- ***-import alias cert-file*** — Imports a CA-signed SSL server digital certificate from the disk file *cert-file*, pairs it with the private key generated for a public key request identified by the alias name *alias*, and places the pair in the key store as a new entry identified by *alias*
- ***-print alias*** — Displays the public-key certificate request identified by *alias*.
- ***-list*** **[** ***alias*** **...** **]** — Displays a list of specified (*alias*) or all current keystore entries
- ***-display cert-file*** — Displays the digital certificate file information contained in the operating system disk file *cert-file*

- ***-remove alias ...*** — Removes one or more specified (*alias*) keystore entries

For complete information on the options and functions of the pkiutil command-line utility, see pkiutil on page 15

About the keystore content

The OpenEdge keystore maintains private keys and digital certificates for OpenEdge SSL servers in several locations. These include private keys and digital certificates that you have authorized by a CA and imported for use by an SSL server, and private keys and public-key certificate requests that you generate and have pending for authorization by a CA. You must manage this keystore entirely with the pkiutil command-line utility. See Use pkiutil to manage an OpenEdge keystore on page 5 for additional information.

The keystore resides in the *OpenEdge-Install-Dir*\keys directory. This directory contains the following files and subdirectories:

- **alias.pem**—Files containing a single keystore entry that you have created from an imported CA-authorized digital certificate that contains the public key joined with the private key that you generated along with the original public-key certificate request. Each file is named with the *alias* that you chose for the original private key and certificate request using the -newreq operation of pkiutil. The initial key store entry is the default OpenEdge entry default_server.pem, as authorized by the Progress Software Corporation CA. For more information on this default keystore entry, see the sections on SSL in *Introduction to Security and Auditing*.
- **policy**—A subdirectory containing a pscpki.cnf configuration file. The pkiutil utility uses this file to control the process of generating new SSL server private/public keys and generating digital certificate requests that can be sent to a CA in order to obtain a public-key certificate for the OpenEdge SSL server. Initially, this is the only subdirectory.
- requests —A subdirectory containing all newly generated private keys and public-key certificate requests in the form of the following two files:
  - **alias.pk1**—This file holds the PKCS #1-formatted, password-encrypted, private key for the given keystore *alias* entry.
  - **alias.pk10**—This file holds the PKCS #10-formatted public-key certificate request that you send to a CA to obtain the SSL server's public-key certificate for the given keystore *alias* entry.
- **backup**—A subdirectory containing any removed keystore entries. The pkiutil utility removes an existing keystore entry when you:
  - Explicitly remove it using the -remove operation of pkiutil.
  - Update an existing keystore entry with a new digital certificate. You will perform this operation when the previous public-key certificate has expired and you have applied to the CA for a renewed public-key certificate.

  In all cases, pkiutil places removed keystore entries in this directory in case you find it necessary to recover and use them again.

Note: Performing successive -remove or -import operations on the same keystore entry repeatedly overwrites that entry in the backup subdirectory.

Caution: If you upgrade or uninstall OpenEdge, Progress recommends that you back up your current version of the OpenEdge keystore directory tree (*OpenEdge-Install-Dir*\keys) to prevent losing valuable keys and certificates.

Use genpassword to obtain a keystore password-encrypted value

When you must configure an OpenEdge SSL server by manually editing the ubroker.properties file, or for the OpenEdge RDMS when you start up the database server to enable SSL connections, you must specify the password to allow access to the required private-key alias. The value you specify is available to anyone who can read the file or command line where you enter it. To prevent access to this password by unauthorized users, you must specify an encrypted form of the password that is equivalent to the password itself.

Note: You must also provide the encrypted form of the password ("password") for the default_server alias. In OpenEdge Management, when you configure an SSL server with the default_server alias, OpenEdge automatically provides the encrypted form of this password.

OpenEdge provides the genpassword command-line utility that you can use to obtain the encoded and encrypted form for the real password.

For example, when the following code is executed in the OpenEdge Proenv command window, you can generate an encrypted value for a password, represented in this example as *password*:

```
proenv>genpassword -password topsecret
243d3f343726213624
proenv>
```

Later, to verify that an existing encrypted value matches the real password value, you can run genpassword, as follows:

```
proenv>genpassword -password password -verify 243d3f343726213624
The passwords match.

proenv>
```

For more information on the options of the genpassword command-line utility, see genpassword on page 14.

Manage certificate stores for OpenEdge clients and servers

You can manage trusted CA/root digital (public-key) certificates for OpenEdge clients and servers that support SSL connections using a **root certificate store** located in the *OpenEdge-Install-Dir*\certs directory. Each OpenEdge SSL client and server requires the **root certificate store entry** that contains the public-key certificate from the CA who signed and issued the public-key certificate for the SSL server that the client and server needs to access. Without access to this CA's root digital certificate, the OpenEdge clients and servers will be unable to validate the identity of the SSL server and will abort the SSL connection process. For more information on the OpenEdge client and server components that support SSL client and server configuration, see the sections on the supported SSL client components in *Introduction to Security and Auditing*.

If you require only data encryption and do not need to verify the identity of SSL servers (typically, for intranet configurations only), OpenEdge comes with the root digital certificate from the Progress Software Corporation CA (who also signed and issued the default_server key store digital certificate for OpenEdge SSL servers already installed). The Progress Software Corporation CA root digital certificate is distributed in PEM format as d9855a82.0 and in DER format as pscca.cer (suitable for importing into a Windows workstation for use by an OpenEdge .NET Open Client). This default entry contains a common root public-key certificate that you can use to access any supported OpenEdge SSL server. For more information on the default root public-key certificate, see the sections on the OpenEdge default server identity in *Introduction to Security and Auditing*.

Install trusted CA/root certificates

To allow OpenEdge client access to an SSL server whose identity you need to verify, you must install the appropriate root digital certificate to authenticate that server. An SSL server can have its identity established from one of two basic sources:

- One of the trusted public CA root digital certificates distributed by Progress Software Corporation that includes RSA, Thawte, and Verisign
- A root digital certificate from an internal CA that you have set up on your own certificate server or from another external or public CA other than RSA, Thawte, or Verisign

OpenEdge automatically installs root certificates in the OpenEdge root certificate store from RSA, Thawte, and Verisign. However, if you use your own internal-use CA or a public CA other than these three, you must install the required root certificates yourself.

OpenEdge provides the following command-line utilities to install and manage root certificates in the OpenEdge certificate store:

- **certutil**—Installs, lists, and manages CA/root certificates from any CA as entries in the OpenEdge root certificate store, and manages the certificate store for the client and server. You can also remove certificate store entries using this utility. The utility moves all removed entries to a backup subdirectory of the root certificate store for future recovery and use.

  Note: For .NET and Java Open Clients and Web service clients of OpenEdge application servers, you must use other utilities to manage the root certificate stores for those clients and servers. For more information, see *Use the Open Client Toolkit*.

- **mkhashfile**—Provides simple installation of PEM-encoded root certificates into the OpenEdge root certificate store from any CA, but provides no other management functions for the OpenEdge certificate store. You can use certutil for the additional root certificate management.

Note: Before you run an OpenEdge command-line utility, set the DLC environment variable to the *OpenEdge-install-dir* pathname and set the WRKDIR environment variable to your working directory. For an example, see the *OpenEdge-install-dir*/bin/certutil shell script on UNIX or the *OpenEdge-install-dir*\bin\certutil.bat file in Windows. Running the command-line utility in a Proenv command window properly sets DLC and WRKDIR for you.

Use certutil to manage an OpenEdge root certificate store

The certutil command-line utility provides functions to install root certificates from any CA and to manage all of the entries in the OpenEdge root certificate store.

The certutil utility has the following general command-line syntax:

```
certutil [options] functionarguments
```

options

Changes the type of information and defaults for different functions (*function*) of the utility.

function arguments

Specify one of the following functions (*function*) and the objects they affect (*arguments*):

- **-import cert-file**—Imports a trusted CA root certificate from the disk file *cert-file*, and creates a root certificate store entry identified by a generated alias name (*alias*, as specified for other functions of this utility)
- **-list [alias . . .]** —Displays a list of specified (*alias*) or all current certificate store entries
- **-display cert-file**—Displays the digital certificate file information contained in the operating system disk file *cert-file*
- **-remove alias . . .**—Removes one or more specified (*alias*) certificate store entries

For more information on the options and functions of the certutil command-line utility, see certutil on page 12.

Use mkhashfile to install root certificates in the OpenEdge root certificate store

The mkhashfile command-line utility provides a simple way to install a root certificate that is authorized by your own internal-use CA, or any CA that can provide you with a PEM-encoded certificate (typically in a file named with the .pem extension). If you are using your own certificate server to provide the certificate, refer to the documentation for the certificate server administration software for information on how to obtain PEM-encoded certificates. After you have the certificate accessible to your OpenEdge SSL client machine, you can use the mkhashfile command-line utility to install it in the OpenEdge root certificate store.

Note: If the root certificate is not a PEM-encoded certificate, Progress recommends that you use the `certutil` command-line utility, specifying the format option. For details about the `certutil` command-line utility and all its options and functions, see the detailed syntax information for the certutil command listed in certutil on page 12

To use `mkhashfile` to create an entry in the OpenEdge root certificate store for a local PEM-encoded certificate file, `vsigntca.pem`, specify the file with the `mkhashfile` command that you enter in the OpenEdge `Proenv` command window. For example:

```
proenv>mkhashfile vsigntca.pem

OpenEdge Release 11.4 as of Fri Oct 14 00:15:12 EST 2011

Running SSLC command ...
Copying vsigntca.pem and 18d46017.0 to C:\Progress\OpenEdge\certs
proenv>
```

The utility generates the entry as a file with an encrypted filename, `18d46017.0`, which is the alias used to identify the certificate store entry. You can then manage this entry along with all other entries in the OpenEdge certificate store using the `certutil` utility. For more information see Using certutil to manage an OpenEdge root certificate store.

For more information on the `mkhashfile` command-line utility, see mkhashfile on page 15

2

Installing and managing keys and digital certificates

This section identifies each OpenEdge utility that allows you to install and manage keys and digital certificates (in key stores and certificate stores) so the components can use them. For Open Clients and clients of Progress Web services, OpenEdge provides utilities for some clients or relies on utilities provided by the client platform to manage the required certificate stores.

The utilities presented in this section are:

- certutil on page 12
- gendomreg on page 13
- genpassword on page 14
- mkhashfile on page 15
- pkiutil on page 15
- sslj on page 18

For details, see the following topics:

- certutil
- gendomreg
- genpassword
- mkhashfile
- pkiutil

- sslj

certutil

Provides all the functions necessary to install and manage root certificates from any Certification Authority (CA) as entries in the root certificate store of an OpenEdge client machine (located in *OpenEdge-Install-Dir*\certs).

Operating system	Syntax

UNIX
Windows
	`certutil [ -brief | -verbose ]`
	`{ [ -format { DER | PEM } ] -display cert-file`
	`| [ -format { DER | PEM } ] -import cert-file`
	`| -list [alias . . . ]`
	`| -remove alias . . .`
	`}`

`-brief`

Provides less information or as specified for the function.

`-verbose`

Provides more information or as specified for the function.

`-format { DER | PEM }`

Specifies the certificate format for the `-import` and `-display` functions. The default input format for a certificate is Privacy Enhanced Mail (PEM). Because some CAs issue root digital certificates in a binary format (DER), you must specify `-format DER` to import these certificates.

`-display` *cert-file*

Displays the digital certificate file information contained in the operating system disk file, *cert-file*. You must specify *cert-file* as a fully qualified operating system file path name. The `-verbose` option displays complete certificate information, and the `-brief` option displays less certificate information for each certificate store entry.

`-import` *cert-file*

Imports a trusted CA digital certificate from the disk file *cert-file*. The *cert-file* argument must specify a fully qualified operating system file pathname. The function creates a root certificate store entry with a generated alias name and displays that alias name for view (specified by *alias* for other functions of this command).

Note:

If you installed a new version (11.6 or later) of OpenEdge, the root certificates must be reimported from the previous version (11.5 or earlier).

All root digital certificate store entry alias names are exactly eight hexadecimal characters in length and have a `.0` (dot-zero) file extension. All other files in the root certificate store are ignored.

`-list [`*`alias`*`...]`

Displays a list of certificate store entries identified by each alias name (*`alias`*). You can specify multiple aliases, but you cannot use wild cards. If you specify no alias, `certutil` displays all entries in the certificate store. The `-verbose` option displays complete certificate information and the `-brief` option displays less certificate information per key store entry.

`-remove` *`alias`*`...`

Removes one or more certificate store entries that you specify by their *`alias`*. You cannot use wild cards. Moves each specified certificate store entry into the `backup` subdirectory and overwrites any previous `backup` subdirectory entry with the same alias name.

For more information on managing root certificates in the OpenEdge root certificate store, see Manage OpenEdge Keys and Certificates on page 3.

gendomreg

Creates an encrypted registry file that can be used to seal a client-principal token. It is most often used when the token must be validated against multiple domains. Multiple *`domain_name`*,*`access_code`* pairs are specified in a formatted CSV (Comma-Separated Value) text file.

Operating system	Syntax
UNIX **Windows**	`gendomreg` *`domain`*`.csv` *`registyfile`*

`domain``.csv`

The filename of the CSV-formatted text file that contains a list of domain name and access code pairs. The domain name must be separated from the access code by a comma. Each pair must appear on a separate line.

For example:

```
 ,
domain1,domain-access-code1
domain2,domain-access-code2
domain3,domain-access-code3
```

Notice that the first line (,) references the OpenEdge default blank domain, which supports user credentials where domains are not specified



Warning: Since the CSV is a clear text file and might be read by almost anyone, be sure to remove it from your server and save it in a secure location.

registyfile

The filename of the encrypted registry file that GENDOMREG generates, which can be used to seal a client-principal object.

Note: You can copy *registyfile* to the /webapps/ROOT/WEB-INF/config directory of a PAS for OpenEdge instance, such as the OpenEdge Authentication Gateway.

genpassword

Accepts the clear-text value of a password and generates the encoded and encrypted form for the specified password.

Operating system	Syntax
UNIX **Windows**	genpassword -password *text* **[** -verify *encrypted-password* **]**

-password *text*

Where *text* is the clear-text value of the real password. When you specify the -password option alone, the tool displays a string of characters that represent the encrypted password value. You can then use this value directly to manually specify a required password as a property or command-line parameter value when manually configuring an OpenEdge client or server.

-verify *encrypted-password*

Where *encrypted-password* is the value of the encrypted password. When you specify the -verify option, the tool displays a message indicating if the real password and the encrypted password value match one another.

mkhashfile

Provides a simple way to install a root certificate in the OpenEdge root certificate store of a client machine. Such a certificate can be authorized by your own internal-use Certification Authority (CA) or by any CA that can provide you with a PEM-encoded certificate.

Operating system	Syntax
UNIX **Windows**	`mkhashfile` *`PEM-certificate-pathname`*

`PEM-certificate-pathname`

Pathname of a PEM-encoded certificate file (typically with a `.pem` extension) containing a root certificate that you want to store in an OpenEdge root certificate store. The command creates a copy of this file with a hashed filename and places it in the *`OpenEdge-install-dir`*`/certs` directory of the client machine. The generated filename becomes the alias for the root certificate store entry.

You can use `certutil` to manage the root certificates that you install with this utility. For more information on managing root certificates in the OpenEdge root certificate store, see certutil on page 12 and Manage OpenEdge Keys and Certificates on page 3.

pkiutil

Provides the functions to create and manage key store entries for OpenEdge SSL servers. It creates these entries from pairs of private keys and digital certificates that it stores in the OpenEdge server key store (located in *`OpenEdge-Install-Dir`*`\keys`).

Note: You must submit a public-key certificate request that is generated for each new key store entry that you want to create a Certification Authority (CA) with this utility. The CA then returns the necessary server (public-key) certificate for you to import and completes creation of the new key store entry.

Operating system	Syntax
UNIX **Windows**	pkiutil [-brief | -verbose] { [-format { DER | PEM }] -display *cert-file* | [-format { DER | PEM }] -import *alias cert-file* | -list [*alias* . . .] | [-keysize *size*] -newreq *alias* | -print *alias* | -remove *alias* . . . | -exportp12 -alias *alias-name* -p12file *p12file-path*

-brief

Provides less information or as specified for the function.

-verbose

Provides more information or as specified for the function.

-format { DER | PEM }

Specifies the certificate format for the -import and -display functions. The default input format for a certificate is Privacy Enhanced Mail (PEM). Because some CAs issue public-key certificates in a binary format (DER) you must specify -format DER to import these certificates.

-display *cert-file*

Displays the digital certificate file information contained in the operating system disk file, *cert-file*. You must specify *cert-file* as a fully qualified operating system file pathname. The -verbose option displays complete certificate information, and the -brief option displays less certificate information for each key store entry.

-import *alias cert-file*

Imports a CA-issued SSL server digital (public-key) certificate from the disk file, *cert-file*, pairs it with the -newreq-generated private key identified by the specified alias name (*alias*), and places the pair in the key store as a new entry identified by *alias*. The function prompts for the same password used to generate the public-key certificate request for this entry.

-list [*alias* . . .]

Displays a list of key store entries identified by each alias name (*alias*). You can specify multiple aliases, but you cannot use wild cards. If you specify no alias, pkiutil displays all entries in the key store. The -verbose option displays complete certificate information, and the -brief option displays less certificate information per key store entry.

[-keysize *size***]** -newreq *alias*

Generates a new private/public-key pair and a corresponding public-key certificate request (suitable for submission to a CA), stored under the alias name specified by *alias*, and placed in the *OpenEdge-Install-Dir*\keys\requests directory.

You must specify an *alias* name between 5 and 39 characters long and use only the following characters:

- "0" to "9"
- "a" to "z"
- "A" to "Z"
- "_" and "-"

Note: The character "-" cannot be used as the first character.

The function prompts for a password with a minimum of four characters using any printable ASCII character. You must use this same password later to create and allow access to the key store entry generated from this certificate request.

When pkiutil generates the keys and certificate request for this function, by default it generates keys using the RSA asymmetric encryption algorithm with a 1024-bit key size. If you require a different key size, you can specify the number of bits to generate using the -keysize option (valid key sizes must be 512, 1024, or 2048 bits).

-print *alias*

Displays the public-key certificate request identified by *alias*.

-remove *alias* **. . .**

Removes the specified entries from the key store by their *alias* and moves them to the backup subdirectory of the key store. Any entry that has the same *alias* is overwritten. You cannot use wild cards.

-exportp12

Generates the PKCS12 keystore file for PAS for OpenEdge instances using the .pem server or public keys, both of which are also used by other OpenEdge server products.

-alias *alias-name*

Specifies the name of the PKCS12 keystore file.

-p12file *p12file-path*

Specifies the path of the location in which the PKCS12 keystore file is saved after it is generated.

sslj

The `sslj` utility helps you to test and troubleshoot SSL or TLS communication and list the ciphers and protocols using the same Java JSSE library that is used by all Java based OpenEdge products such as classic Appserver, PAS for OE, classic WebSpeed, Java OpenClient, and OEM. The utility is located in `$DLC/bin`.

Here is the complete syntax of this tool. Each operation is discussed in detail in the sections that follow:

```
sslj operation [option]
     operation:
                s_server [option]
                s_client [option]
                list-protocols
                list-ciphers
                version
                help [s_server | s_client]
```

s_server

Starts the HTTPS server with the parameters provided.

Operating system	Syntax

UNIX
Windows
	`sslj s_server -accept` *port value* `-cert` *certificate path* `-certPass` *certificate password*
	`[-CAfile` *CA file path*`]` `[-CAPass` *CA password*`]`
	`[-protocol]` `[-debug` *debug value*`]`
	`[-log` *log directory path*`]` `[-verify]`

`-accept` *port value*

Provides the port number that the server uses for the SSL handshake.

`-cert` *certificate path*

Provides the path where the server certificate resides.

Note: The supported file extensions for the certificate are `.jks, .jceks, .p12, .pfx, .bks`.

`-certPass` *certificate password*

Provides the password for the server certificate.

`-CAfile` *CA file path*

Provides the path for the CA certificate for client authentication.

Note: The supported file extensions for the certificate are `.jks` and `.p12`. Make sure that you provide the absolute path, for example, `C:\xyz\abc.p12`.

`-CAPass` *`CA password`*

Provides the password for the CA certificate.

`-protocol`

Specifies the protocol that the server uses.

`-debug` *`debug value`*

Turns on the debugging, the valid values are `all` and `ssl`. Here are the values that you can choose for `ssl`:

- `record`: Enables tracing for each record. The valid values are `plaintext` for hex dump of record and `packet` for raw SSL or TLS packets.
- `handshake`: Displays each handshake message. The valid values are `data` for hex dump of each handshake message and `verbose` for handshake message printing.
- `keygen`: Displays the key generation data.
- `session`: Displays the details of the session activity.
- `defaultctx`: Displays the default SSL initialization.
- `sslctx`: Displays the SSL context tracing.
- `sessioncache`: Displays the session cache tracing.
- `keymanager`: Displays the key manager tracing.
- `trustmanager`: Displays the trust manager tracing.

Note: If you have provided a filename using the `-log` option, the above details are updated in the file or else, they are displayed on the `proenv`.

`-log` *`log directory path`*

Specifies the path where the `s_server.log` file is generated.

`-verify`

(Optional) Validates the client certificate when the server starts.

Examples:

```
proenv> sslj s_server -accept 8888 -cert server.p12 -certPass password
-CAfile ca.jks -CAPass password -debug ssl:handshake -log C:\Progress\WRK\-verify
 -TLSv1.1


proenv> sslj s_server -accept 8888 -cert server.jks -certPass password
-CAfile ca.jks -CAPass password -debug ssl -log C:\Progress\WRK\-verify -TLSv1.1
```

Note: For more information on an example on creating and managing a .jks keystore, see the *OpenEdge Management and OpenEdge Explorer: Getting Started* guide.

s_client

Starts the HTTPS client with the parameters provided.

Operating system	Syntax

UNIX
Windows
	`sslj s_client -connect host:port`
	`-CAfile CA file path-CAPass CA password`
	`[-cert certificate path] [-certPass certificate password]`
	`[-protocol] [-debug debug value]`
	`[-log log directory path]`

`-connect host:port`

Provides the host and port name that the client uses to connect to the server.

`-CAfile CA file path`

Provides the path for the CA certificate for server authentication.

Note: The supported file extensions for the certificate are `.jks` and `.p12`. Make sure that you provide the absolute path, for example, `C:\xyz\abc.p12`.

`-CAPass CA password`

Provides the password for the CA certificate.

`-cert certificate path`

Provides the path where the server certificate for client authentication resides.

Note: The supported file extensions for the certificate are `.jks, .jceks, .p12, .pfx, .bks`.

`-certPass certificate password`

Provides the password for the server certificate.

`-protocol`

Specifies the protocol that the client uses.

`-debug debug value`

Turns on the debugging, the valid values are `all` and `ssl`. Here are the values that you can choose for `ssl`:

- `record`: Enables tracing for each record. The valid values are `plaintext` for hex dump of record and `packet` for raw SSL or TLS packets.
- `handshake`: Displays each handshake message. The valid values are `data` for hex dump of each handshake message and `verbose` for handshake message printing.
- `keygen`: Displays the key generation data.
- `session`: Displays the details of the session activity.
- `defaultctx`: Displays the default SSL initialization.
- `sslctx`: Displays the SSL context tracing.
- `sessioncache`: Displays the session cache tracing.
- `keymanager`: Displays the key manager tracing.
- `trustmanager`: Displays the trust manager tracing.

Note: If you have provided a filename using the `-log` option, the above details are updated in the file or else, they are displayed on the `proenv`.

`-log` *`log directory path`*

Specifies the path where the `s_client.log` file is generated.

Examples:

```
proenv>sslj s_client -connect localhost:8888 -CAfile ca.p12 -CAPass password
-cert server.p12 -certPass password -debug ssl:session -log C:\Progress\WRK -TLSv1.1
```

```
proenv>sslj s_client -connect localhost:8888 -CAfile ca.jks -CAPass password
-cert server.jks -certPass password -debug ssl -log C:\Progress\WRK -TLSv1.1
```

Note: For more information on an example on creating and managing a .jks keystore, see the *OpenEdge Management and OpenEdge Explorer: Getting Started* guide.

list-protocols

Lists all the protocols that are enabled by your platform and that are supported by OpenEdge.

Operating system	Syntax
UNIX **Windows**	`sslj list-protocols`

Example:

```
proenv>sslj list-protocols
```

list-ciphers

Lists all the ciphers that are enabled by your platform and that are supported by OpenEdge.

Operating system	Syntax
UNIX **Windows**	`sslj list-ciphers`

Example:

```
proenv>sslj list-ciphers
```

version

Displays the version of the JSSE library.

Operating system	Syntax
UNIX **Windows**	`sslj version`

Example:

```
proenv>sslj version
```

sslj help

Lists a detailed usage of the `sslj` utility.

Operating system	Syntax
UNIX **Windows**	`sslj help [s_server | s_client]`