Configuring IceSSL for OpenSSL

Previous Next

After installing IceSSL, an application typically needs to define a handful of additional properties to configure settings such as the location of certificate and key files. This page provides an introduction to configuring the plug-in for applications using the OpenSSL version of the plug-in.

On this page:

Configuring IceSSL for OpenSSL on Linux

Our first example shows the properties that are sufficient in many situations:

```
Ice.Plugin.IceSSL=IceSSL:createIceSSL
IceSSL.DefaultDir=/opt/certs
IceSSL.CertFile=cert.pfx
IceSSL.CAs=ca.pem
IceSSL.Password=password
```

The `IceSSL.DefaultDir` property is a convenient way to specify the default location of your certificate and key files. The two properties that follow it define the files containing the program's certificate with private key and trusted CA certificate, respectively. This example assumes the files contain RSA keys, and IceSSL requires the files to use the Privacy Enhanced Mail (PEM) encoding. Finally, the `IceSSL.Password` property specifies the password of the private key.

It is a security risk to define a password in a plain text file, such as an Ice configuration file, because anyone who can gain read access to your configuration file can obtain your password. IceSSL also supports alternate ways to supply a password.

DSA Example for OpenSSL on Linux

If you used DSA to generate your keys, one additional property is necessary:

```
Ice.Plugin.IceSSL=IceSSL:createIceSSL
IceSSL.DefaultDir=/opt/certs
IceSSL.CertFile=cert_dsa.pfx
IceSSL.CAs=ca.pem
IceSSL.Password=password
IceSSL.Ciphers=DEFAULT:DSS
```

The `IceSSL.Ciphers` property adds support for DSS authentication to the plug-in's default set of ciphersuites.

Back to Top ^

RSA and DSA Example for OpenSSL on Linux

It is also possible to specify certificates and keys for both RSA and DSA by including two filenames in the `IceSSL.CertFile` property. The file names must be separated using the platform's path separator. The example below demonstrates the configuration:

```
Ice.Plugin.IceSSL=IceSSL:createIceSSL
IceSSL.DefaultDir=/opt/certs
IceSSL.CertFile=cert_rsa.pfx:cert_dsa.pfx
IceSSL.CAs=ca.pem
IceSSL.Password=password
IceSSL.Ciphers=DEFAULT:DSS
```

Back to Top ^

ADH Example for OpenSSL on Linux

The following example uses ADH (the Anonymous Diffie-Hellman cipher). ADH is not a good choice in most cases because, as its name implies, there is no authentication of the communicating parties, and it is vulnerable to man-in-the-middle attacks. However, it still provides encryption of the session traffic and requires very little administration and therefore may be useful in certain situations. The configuration properties shown below demonstrate how to use ADH:

```
Ice.Plugin.IceSSL=IceSSL:createIceSSL
IceSSL.Ciphers=ADH:!LOW:!MD5:!EXP:!3DES:@STRENGTH
IceSSL.VerifyPeer=0
```

The `IceSSL.Ciphers` property enables support for ADH (which is disabled by default) and eliminates low-strength ciphersuites.

The `IceSSL.VerifyPeer` property changes the plug-in's default behavior with respect to certificate verification. Without this setting, IceSSL rejects a connection if the peer does not supply a certificate (as is the case with ADH).

Configuring IceSSL for OpenSSL on Windows

C++ Windows applications can use an alternative implementation of the IceSSL plug-in that uses OpenSSL instead of the default implementation that uses SChannel:

```
Ice.Plugin.IceSSL=IceSSLOpenSSL:createIceSSLOpenSSL

IceSSL.DefaultDir=C:\certs
IceSSL.CAs=cacert.pem
IceSSL.CertFile=cert.pfx
IceSSL.Password=password
```

Applications using the OpenSSL plug-in cannot use Windows certificate stores.

Back to Top ^

ADH Example for OpenSSL on Windows

The following example uses ADH (the Anonymous Diffie-Hellman cipher). ADH is not a good choice in most cases because, as its name implies, there is no authentication of the communicating parties, and it is vulnerable to man-in-the-middle attacks. However, it still provides encryption of the session traffic and requires very little administration and therefore may be useful in certain situations. The configuration properties shown below demonstrate how to use ADH:

```
Ice.Plugin.IceSSL=IceSSLOpenSSL:createIceSSLOpenSSL
IceSSL.Ciphers=ADH:!LOW:!MD5:!EXP:!3DES:@STRENGTH
IceSSL.VerifyPeer=0
```

The `IceSSL.Ciphers` property enables support for ADH (which is disabled by default) and eliminates low-strength ciphersuites.

The `IceSSL.VerifyPeer` property changes the plug-in's default behavior with respect to certificate verification. Without this setting, IceSSL rejects a connection if the peer does not supply a certificate (as is the case with ADH).

Back to Top ^

See Also

Previous Next