NAME

SSL_CONF_cmd_value_type, SSL_CONF_finish, SSL_CONF_cmd - send configuration command

SYNOPSIS

```
 #include <openssl/ssl.h>

 int SSL_CONF_cmd(SSL_CONF_CTX *cctx, const char *cmd, const char *value);
 int SSL_CONF_cmd_value_type(SSL_CONF_CTX *cctx, const char *cmd);
 int SSL_CONF_finish(SSL_CONF_CTX *cctx);
```

DESCRIPTION

The function *SSL_CONF_cmd()* performs configuration operation **cmd** with optional parameter **value** on **ctx**. Its purpose is to simplify application configuration of **SSL_CTX** or **SSL** structures by providing a common framework for command line options or configuration files.

SSL_CONF_cmd_value_type() returns the type of value that **cmd** refers to.

The function *SSL_CONF_finish()* must be called after all configuration operations have been completed. It is used to finalise any operations or to process defaults.

SUPPORTED COMMAND LINE COMMANDS

Currently supported **cmd** names for command lines (i.e. when the flag **SSL_CONF_CMDLINE** is set) are listed below. Note: all **cmd** names are case sensitive. Unless otherwise stated commands can be used by both clients and servers and the **value** parameter is not used. The default prefix for command line commands is **-** and that is reflected below.

-sigalgs

This sets the supported signature algorithms for TLS v1.2. For clients this value is used directly for the supported signature algorithms extension. For servers it is used to determine which signature algorithms to support.

The **value** argument should be a colon separated list of signature algorithms in order of decreasing preference of the form **algorithm+hash**. **algorithm** is one of **RSA**, **DSA** or **ECDSA** and **hash** is a supported algorithm OID short name such as **SHA1**, **SHA224**, **SHA256**, **SHA384** of **SHA512**. Note: algorithm and hash names are case sensitive.

If this option is not set then all signature algorithms supported by the OpenSSL library are permissible.

-client_sigalgs

This sets the supported signature algorithms associated with client authentication for TLS v1.2. For servers the value is used in the supported signature algorithms field of a certificate request. For clients it is used to determine which signature algorithm to with the client certificate. If a server does not request a certificate this option has no effect.

The syntax of **value** is identical to **-sigalgs**. If not set then the value set for **-sigalgs** will be used instead.

-curves

This sets the supported elliptic curves. For clients the curves are sent using the supported curves extension. For servers it is used to determine which curve to use. This setting affects curves used for both signatures and key exchange, if applicable.

The **value** argument is a colon separated list of curves. The curve can be either the **NIST** name (e.g. **P-256**) or an OpenSSL OID name (e.g **prime256v1**). Curve names are case sensitive.

-named_curve

This sets the temporary curve used for ephemeral ECDH modes. Only used by servers

The **value** argument is a curve name or the special value **auto** which picks an appropriate curve based on client and server preferences. The curve can be either the **NIST** name (e.g. **P-256**) or an OpenSSL OID name (e.g **prime256v1**). Curve names are case sensitive.

-cipher
Sets the cipher suite list to **value**. Note: syntax checking of **value** is currently not performed unless a **SSL** or **SSL_CTX** structure is associated with **cctx**.

-cert
Attempts to use the file **value** as the certificate for the appropriate context. It currently uses *SSL_CTX_use_certificate_chain_file()* if an **SSL_CTX** structure is set or *SSL_use_certificate_file()* with filetype PEM if an **SSL** structure is set. This option is only supported if certificate operations are permitted.

-key
Attempts to use the file **value** as the private key for the appropriate context. This option is only supported if certificate operations are permitted. Note: if no **-key** option is set then a private key is not loaded unless the flag **SSL_CONF_FLAG_REQUIRE_PRIVATE** is set.

-dhparam
Attempts to use the file **value** as the set of temporary DH parameters for the appropriate context. This option is only supported if certificate operations are permitted.

-no_renegotiation
Disables all attempts at renegotiation in TLSv1.2 and earlier, same as setting **SSL_OP_NO_RENEGOTIATION**.

-min_protocol, **-max_protocol**
Sets the minimum and maximum supported protocol. Currently supported protocol values are **SSLv3**, **TLSv1**, **TLSv1.1**, **TLSv1.2** for TLS and **DTLSv1**, **DTLSv1.2** for DTLS, and **None** for no limit. If the either bound is not specified then only the other bound applies, if specified. To restrict the supported protocol versions use these commands rather than the deprecated alternative commands below.

-no_ssl3, **-no_tls1**, **-no_tls1_1**, **-no_tls1_2**
Disables protocol support for SSLv3, TLSv1.0, TLSv1.1 or TLSv1.2 by setting the corresponding options **SSL_OP_NO_SSLv3**, **SSL_OP_NO_TLSv1**, **SSL_OP_NO_TLSv1_1** and **SSL_OP_NO_TLSv1_2** respectively. These options are deprecated, instead use**-min_pr otocol** and **-max_protocol**.

-bugs
Various bug workarounds are set, same as setting **SSL_OP_ALL**.

-comp
Enables support for SSL/TLS compression, same as clearing **SSL_OP_NO_COMPRESSION**. This command was introduced in OpenSSL 1.1.0. As of OpenSSL 1.1.0, compression is off by default.

-no_comp
Disables support for SSL/TLS compression, same as setting **SSL_OP_NO_COMPRESSION**. As of OpenSSL 1.1.0, compression is off by default.

-no_ticket
Disables support for session tickets, same as setting **SSL_OP_NO_TICKET**.

-serverpref
Use server and not client preference order when determining which cipher suite, signature algorithm or elliptic curve to use for an incoming connection. Equivalent to **SSL_OP_CIPHER_SERVER_PREFERENCE**. Only used by servers.

-no_resumption_on_reneg
set SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION flag. Only used by servers.

-legacyrenegotiation
permits the use of unsafe legacy renegotiation. Equivalent to setting **SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION**.

-legacy_server_connect, **-no_legacy_server_connect**

permits or prohibits the use of unsafe legacy renegotiation for OpenSSL clients only. Equivalent to setting or clearing **SSL_OP_LEGACY_SERVER_CONNECT**. Set by default.

-strict

enables strict mode protocol handling. Equivalent to setting **SSL_CERT_FLAG_TLS_STRICT**.

SUPPORTED CONFIGURATION FILE COMMANDS

Currently supported **cmd** names for configuration files (i.e. when the flag **SSL_CONF_FLAG_FILE** is set) are listed below. All configuration file **cmd** names are case insensitive so **signaturealgorithms** is recognised as well as **SignatureAlgorithms**. Unless otherwise stated the **value** names are also case insensitive.

Note: the command prefix (if set) alters the recognised **cmd** values.

CipherString

Sets the cipher suite list to **value**. Note: syntax checking of **value** is currently not performed unless an **SSL** or **SSL_CTX** structure is associated with **cctx**.

Certificate

Attempts to use the file **value** as the certificate for the appropriate context. It currently uses *SSL_CTX_use_certificate_chain_file()* if an **SSL_CTX** structure is set or *SSL_use_certificate_file()* with filetype PEM if an **SSL** structure is set. This option is only supported if certificate operations are permitted.

PrivateKey

Attempts to use the file **value** as the private key for the appropriate context. This option is only supported if certificate operations are permitted. Note: if no **PrivateKey** option is set then a private key is not loaded unless the **SSL_CONF_FLAG_REQUIRE_PRIVATE** is set.

ChainCAFile, **ChainCAPath**, **VerifyCAFile**, **VerifyCAPath**

These options indicate a file or directory used for building certificate chains or verifying certificate chains. These options are only supported if certificate operations are permitted.

ServerInfoFile

Attempts to use the file **value** in the ‘‘serverinfo’’ extension using the function SSL_CTX_use_serverinfo_file.

DHParameters

Attempts to use the file **value** as the set of temporary DH parameters for the appropriate context. This option is only supported if certificate operations are permitted.

NoRenegotiation

Disables all attempts at renegotiation in TLSv1.2 and earlier, same as setting **SSL_OP_NO_RENEGOTIATION**.

SignatureAlgorithms

This sets the supported signature algorithms for TLS v1.2. For clients this value is used directly for the supported signature algorithms extension. For servers it is used to determine which signature algorithms to support.

The **value** argument should be a colon separated list of signature algorithms in order of decreasing preference of the form **algorithm+hash**. **algorithm** is one of **RSA**, **DSA** or **ECDSA** and **hash** is a supported algorithm OID short name such as **SHA1**, **SHA224**, **SHA256**, **SHA384** of **SHA512**. Note: algorithm and hash names are case sensitive.

If this option is not set then all signature algorithms supported by the OpenSSL library are permissible.

ClientSignatureAlgorithms

This sets the supported signature algorithms associated with client authentication for TLS v1.2. For servers the value is used in the supported signature algorithms field of a certificate request. For clients it is used to determine which signature algorithm to with the client certificate.

The syntax of **value** is identical to **SignatureAlgorithms**. If not set then the value set for **SignatureAlgorithms** will be used instead.

Curves

This sets the supported elliptic curves. For clients the curves are sent using the supported curves extension. For servers it is used to determine which curve to use. This setting affects curves used for both signatures and key exchange, if applicable.

The **value** argument is a colon separated list of curves. The curve can be either the **NIST** name (e.g. **P-256**) or an OpenSSL OID name (e.g **prime256v1**). Curve names are case sensitive.

MinProtocol

This sets the minimum supported SSL, TLS or DTLS version.

Currently supported protocol values are **SSLv3**, **TLSv1**, **TLSv1.1**, **TLSv1.2**, **DTLSv1** and **DTLSv1.2**. The value **None** will disable the limit.

MaxProtocol

This sets the maximum supported SSL, TLS or DTLS version.

Currently supported protocol values are **SSLv3**, **TLSv1**, **TLSv1.1**, **TLSv1.2**, **DTLSv1** and **DTLSv1.2**. The value **None** will disable the limit.

Protocol

This can be used to enable or disable certain versions of the SSL, TLS or DTLS protocol.

The **value** argument is a comma separated list of supported protocols to enable or disable. If a protocol is preceded by **-** that version is disabled.

All protocol versions are enabled by default. You need to disable at least one protocol version for this setting have any effect. Only enabling some protocol versions does not disable the other protocol versions.

Currently supported protocol values are **SSLv3**, **TLSv1**, **TLSv1.1**, **TLSv1.2**, **DTLSv1** and **DTLSv1.2**. The special value **ALL** refers to all supported versions.

This can't enable protocols that are disabled using **MinProtocol** or **MaxProtocol**, but can disable protocols that are still allowed by them.

The **Protocol** command is fragile and deprecated; do not use it. Use **MinProtocol** and **MaxProtocol** instead. If you do use**Pr otocol**, make sure that the resulting range of enabled protocols has no ‘‘holes’’, e.g. if TLS 1.0 and TLS 1.2 are both enabled, make sure to also leave TLS 1.1 enabled.

Options

The **value** argument is a comma separated list of various flags to set. If a flag string is preceded **-** it is disabled. See the *SSL_CTX_set_options(3)* function for more details of individual options.

Each option is listed below. Where an operation is enabled by default the **-flag** syntax is needed to disable it.

SessionTicket: session ticket support, enabled by default. Inverse of **SSL_OP_NO_TICKET**: that is **-SessionTicket** is the same as setting **SSL_OP_NO_TICKET**.

Compression: SSL/TLS compression support, enabled by default. Inverse of **SSL_OP_NO_COMPRESSION**.

EmptyFragments: use empty fragments as a countermeasure against a SSL 3.0/TLS 1.0 protocol vulnerability affecting CBC ciphers. It is set by default. Inverse of **SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS**.

Bugs: enable various bug workarounds. Same as **SSL_OP_ALL**.

DHSingle: enable single use DH keys, set by default. Inverse of **SSL_OP_DH_SINGLE**. Only used by servers.

ECDHSingle enable single use ECDH keys, set by default. Inverse of **SSL_OP_ECDH_SINGLE**. Only used by servers.

ServerPreference use server and not client preference order when determining which cipher suite, signature algorithm or elliptic curve to use for an incoming connection. Equivalent to **SSL_OP_CIPHER_SERVER_PREFERENCE**. Only used by servers.

NoResumptionOnRenegotiation set **SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION** flag. Only used by servers.

UnsafeLegacyRenegotiation permits the use of unsafe legacy renegotiation. Equivalent to **SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION**.

UnsafeLegacyServerConnect permits the use of unsafe legacy renegotiation for OpenSSL clients only. Equivalent to **SSL_OP_LEGACY_SERVER_CONNECT**. Set by default.

VerifyMode

The **value** argument is a comma separated list of flags to set.

Peer enables peer verification: for clients only.

Request requests but does not require a certificate from the client. Servers only.

Require requests and requires a certificate from the client: an error occurs if the client does not present a certificate. Servers only.

Once requests a certificate from a client only on the initial connection: not when renegotiating. Servers only.

ClientCAFile, **ClientCAPath**

A file or directory of certificates in PEM format whose names are used as the set of acceptable names for client CAs. Servers only. This option is only supported if certificate operations are permitted.

SUPPORTED COMMAND TYPES

The function *SSL_CONF_cmd_value_type()* currently returns one of the following types:

SSL_CONF_TYPE_UNKNOWN

The **cmd** string is unrecognised, this return value can be use to flag syntax errors.

SSL_CONF_TYPE_STRING

The value is a string without any specific structure.

SSL_CONF_TYPE_FILE

The value is a file name.

SSL_CONF_TYPE_DIR

The value is a directory name.

SSL_CONF_TYPE_NONE

The value string is not used e.g. a command line option which doesn't take an argument.

NOTES

The order of operations is significant. This can be used to set either defaults or values which cannot be overridden. For example if an application calls:

```
SSL_CONF_cmd(ctx, "Protocol", "-SSLv3");
SSL_CONF_cmd(ctx, userparam, uservalue);
```

it will disable SSLv3 support by default but the user can override it. If however the call sequence is:

```
SSL_CONF_cmd(ctx, userparam, uservalue);
SSL_CONF_cmd(ctx, "Protocol", "-SSLv3");
```

SSLv3 is **always** disabled and attempt to override this by the user are ignored.

By checking the return code of *SSL_CTX_cmd()* it is possible to query if a given **cmd** is recognised, this is useful if *SSL_CTX_cmd()* values are mixed with additional application specific operations.

For example an application might call *SSL_CTX_cmd()* and if it returns -2 (unrecognised command) continue with processing of application specific commands.

Applications can also use *SSL_CTX_cmd()* to process command lines though the utility function *SSL_CTX_cmd_argv()* is normally used instead. One way to do this is to set the prefix to an appropriate value using *SSL_CONF_CTX_set1_prefix()*, pass the current argument to **cmd** and the following argument to **value** (which may be NULL).

In this case if the return value is positive then it is used to skip that number of arguments as they have been processed by *SSL_CTX_cmd()*. If -2 is returned then **cmd** is not recognised and application specific arguments can be checked instead. If -3 is returned a required argument is missing and an error is indicated. If 0 is returned some other error occurred and this can be reported back to the user.

The function *SSL_CONF_cmd_value_type()* can be used by applications to check for the existence of a command or to perform additional syntax checking or translation of the command value. For example if the return value is **SSL_CONF_TYPE_FILE** an application could translate a relative pathname to an absolute pathname.

EXAMPLES

Set supported signature algorithms:

```
SSL_CONF_cmd(ctx, "SignatureAlgorithms", "ECDSA+SHA256:RSA+SHA256:DSA+SHA256");
```

There are various ways to select the supported protocols.

This set the minimum protocol version to TLSv1, and so disables SSLv3. This is the recommended way to disable protocols.

```
SSL_CONF_cmd(ctx, "MinProtocol", "TLSv1");
```

The following also disables SSLv3:

```
SSL_CONF_cmd(ctx, "Protocol", "-SSLv3");
```

The following will first enable all protocols, and then disable SSLv3. If no protocol versions were disabled before this has the same effect as “-SSLv3”, but if some versions were disables this will re-enable them before disabling SSLv3.

```
SSL_CONF_cmd(ctx, "Protocol", "ALL,-SSLv3");
```

Only enable TLSv1.2:

```
SSL_CONF_cmd(ctx, "MinProtocol", "TLSv1.2");
SSL_CONF_cmd(ctx, "MaxProtocol", "TLSv1.2");
```

This also only enables TLSv1.2:

```
SSL_CONF_cmd(ctx, "Protocol", "-ALL,TLSv1.2");
```

Disable TLS session tickets:

```
SSL_CONF_cmd(ctx, "Options", "-SessionTicket");
```

Enable compression:

```
SSL_CONF_cmd(ctx, "Options", "Compression");
```

Set supported curves to P-256, P-384:

```
SSL_CONF_cmd(ctx, "Curves", "P-256:P-384");
```

RETURN VALUES

SSL_CONF_cmd() returns 1 if the value of **cmd** is recognised and **value** is **NOT** used and 2 if both **cmd** and **value** are used. In other words it returns the number of arguments processed. This is useful when processing command lines.

A return value of -2 means **cmd** is not recognised.

A return value of -3 means **cmd** is recognised and the command requires a value but **value** is NULL.

A return code of 0 indicates that both **cmd** and **value** are valid but an error occurred attempting to perform the operation: for example due to an error in the syntax of **value** in this case the error queue may provide additional information.

SSL_CONF_finish() returns 1 for success and 0 for failure.

SEE ALSO

SSL_CONF_CTX_new(3), *SSL_CONF_CTX_set_flags(3)*, *SSL_CONF_CTX_set1_prefix(3)*, *SSL_CONF_CTX_set_ssl_ctx(3)*, *SSL_CONF_cmd_argv(3)*, *SSL_CTX_set_options(3)*

HISTORY

SSL_CONF_cmd() was first added to OpenSSL 1.0.2

SSL_OP_NO_SSL2 doesn't have effect since 1.1.0, but the macro is retained for backwards compatibility.

SSL_CONF_TYPE_NONE was first added to OpenSSL 1.1.0. In earlier versions of OpenSSL passing a command which didn't take an argument would return **SSL_CONF_TYPE_UNKNOWN**.

MinProtocol and **MaxProtocol** where added in OpenSSL 1.1.0.

COPYRIGHT