NAME

SSL_CTX_set_read_ahead, SSL_CTX_get_read_ahead, SSL_set_read_ahead, SSL_get_read_ahead, SSL_CTX_get_default_read_ahead - manage whether to read as many input bytes as possible

SYNOPSIS

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

void SSL_set_read_ahead(SSL *s, int yes);
int SSL_get_read_ahead(const SSL *s);

SSL_CTX_set_read_ahead(SSL_CTX *ctx, int yes);
long SSL_CTX_get_read_ahead(SSL_CTX *ctx);
long SSL_CTX_get_default_read_ahead(SSL_CTX *ctx);
```

DESCRIPTION

SSL_CTX_set_read_ahead() and **SSL_set_read_ahead()** set whether we should read as many input bytes as possible (for nonblocking reads) or not. For example if **x** bytes are currently required by OpenSSL, but **y** bytes are available from the underlying BIO (where **y** > **x**), then OpenSSL will read all **y** bytes into its buffer (providing that the buffer is large enough) if reading ahead is on, or **x** bytes otherwise. Setting the parameter **yes** to 0 turns reading ahead is off, other values turn it on. **SSL_CTX_set_default_read_ahead()** is identical to **SSL_CTX_set_read_ahead()**.

SSL_CTX_get_read_ahead() and **SSL_get_read_ahead()** indicate whether reading ahead has been set or not. **SSL_CTX_get_default_read_ahead()** is identical to **SSL_CTX_get_read_ahead()**.

NOTES

These functions have no impact when used with DTLS. The return values for **SSL_CTX_get_read_head()** and **SSL_get_read_ahead()** are undefined for DTLS. Setting **read_ahead** can impact the behaviour of the **SSL_pending()** function (see **SSL_pending**(3)).

Since **SSL_read()** can return **SSL_ERROR_WANT_READ** for non-application data records, and **SSL_has_pending()** can't tell the difference between processed and unprocessed data, it's recommended that if read ahead is turned on that **SSL_MODE_AUTO_RETRY** is not turned off using **SSL_CTX_clear_mode()**. That will prevent getting **SSL_ERROR_WANT_READ** when there is still a complete record available that hasn't been processed.

If the application wants to continue to use the underlying transport (e.g. TCP connection) after the SSL connection is finished using **SSL_shutdown()** reading ahead should be turned off. Otherwise the SSL structure might read data that it shouldn't.

RETURN VALUES

SSL_get_read_ahead() and **SSL_CTX_get_read_ahead()** return 0 if reading ahead is off, and non zero otherwise.

SEE ALSO

ssl(7), **SSL_pending**(3)

COPYRIGHT