Implementing Secure Transport I/O Functions

342

One of the trickiest parts of using the Secure Transport API is the I/O functions. I posted about this on the old DevForums but I figured I should bring that content into the new world. So, here we go…

The correct way of implementing a Secure Transport read I/O function (

```
SSLReadFunc
```

) depends on your blocking strategy. In blocking mode the function must do one of two things:

- If there's an error, return that error as the function result.

  ```
  errSSLInternal
  ```

  is fine here because, internally, Secure Transport translates any error to

  ```
  errSSLClosedAbort
  ```

  .
  In this case the value stored in

  ```
  *data_length
  ```

  is ignored.
- If there's no error, the function must block waiting for

  ```
  *data_length
  ```

  bytes. Once it's read that many bytes, it should return

  ```
  errSecSuccess
  ```

  .

Note It's obvious that, in the no error case, you don't need to set an output value in

```
*data_length
```

because it necessarily matches the input value.
In non-blocking mode the behaviour is somewhat different:

- In the error case, behave the same as you did in blocking mode.
- In the no error case, return as much data as is available up to the

  ```
  *data_length
  ```

  limit. Set

  ```
  *data_length
  ```

  to the number of bytes read. If that's all of the requested bytes, return

  ```
  errSecSuccess
  ```

  , otherwise return

  ```
  errSSLWouldBlock
  ```

  .

You implement the write function (

```
SSLWriteFunc
```

) is an exactly analogous way.
IMPORTANT If you implement non-blocking mode in your I/O functions, you must be prepared for

```
errSSLWouldBlock
```

to come back from any Secure Transport routine that might call an I/O function (including

```
SSLHandshake
```

,

```
SSLRead
```

and

```
SSLWrite
```

). At that point you have to monitor the channel for more data arriving (or, in the write case, space becoming available) and, when that happens, retry the Secure Transport call.
I've filed a bug (r. 31371376) to add this information to the standard documentation.

Share and Enjoy — Quinn "The Eskimo!" Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Security

Asked 3 years ago by eskimo

Reply to this question