SSL Plus

Version 4.0

Upgrade Guide

PUB- 0300-0212
January 16, 2003

Certicom Corp.
5520 Explorer Drive,
4th Floor,
Mississauga, Ontario,
Canada, L4W 5L1
905.507.4200

certicom

This document is intended for those customers of SSL Plus who are upgrading from previous versions of the product (Desktop and Embedded) to version 4.0.

You can reach Certicom 's technical support department by telephone at
1-800-511-8011, by fax at 1-800-474-3877, or by email at support@certicom.com.

1 Introduction

Content of this document

This document is intended for customers of SSL Plus who are upgrading from version 3.X (Desktop or Embedded series) to version 4.0. It describes the changes to the architecture of the product, and how these changes have affected the API.

The main change is the integration of both SSL Plus Desktop and Embedded into a single product. SSL Plus v4.0 is suitable for both environments, so customers no longer have to choose which product to use. To allow SSL Plus v4.0 to be used to develop applications in a constrained environment, the API has been re-designed to allow you to include only the library code that you need. This reduces the code size of your application (see “Installable Objects” on page 3 for more on this).

In the API, the context structure has been split into a global context and a connection context. This marks a change from previous versions of SSL Plus, where all the data needed to maintain an SSL connection was stored in a single structure. In the new model, the global context contains all the global data for one or more connections, while the connection context contains only what is needed to maintain a single connection. The functions to support both types of structure are described in “Global and Connection Contexts” on page 4.

Minor changes have been made to handshaking and data exchange; these are described in “Handshake and Data Exchange” on page 4.

To help you to use the new API, two tables of function mappings have been included in this document. These tables list the SSL Plus Desktop and Embedded functions along with their version 4.0 equivalents. As a result of combining both the Desktop and Embedded products, the SSL Plus 4.0 library contains fewer functions, which is another benefit of switching to the new product.

Two sample applications (sslsampleclient.c and sslsampleserver.c) have been included with the product.

2 Upgrading to SSL Plus V4.0

Architecture Changes

Cryptographic Libraries

SSL Plus V4.0 now uses Security Builder as its cryptographic engine. Security Builder replaces the EZCrypto component which was used in previous releases of SSL Plus to abstract away the differences between all the supported cryptographic engines. This change to SSL Plus is transparent - there is no impact on the API.

Global and Connection Contexts

The SSL and TLS contexts have each been replaced with two structures: global and connection contexts. A connection context keeps track of a single SSL connection. You create a new connection context each time you make an SSL connection with a peer. A global context contains all the global data for one or more SSL connections. This includes information such as certificate chains, trusted roots and exportable RSA key pairs. You create a global context at the beginning of your application, and then create a connection context from it each time you make a new SSL connection.

See “Changes to the SSL Plus API” on page 4 for more information on the SSL Plus functions that create global and connection contexts.

Note that the context structures have resulted in certain restrictions being placed on the use of the API. See “Restrictions on the use of the SSL Plus API” on page 7.

Installable Objects

The SSL Plus API has been redesigned to allow you to include only the library code you need in your application. By calling an installable object function, you can determine the level of support for cipher suites and protocols in your application. For example, by calling the function **`SSL_PROTOCOL_SSLV3_SERVERSIDE()`**, you can include support for version 3 of the SSL protocol in your server application. Support for other protocols is not included in your application. This results in a smaller code size for your application.

There are several other categories of installable objects; see “Changes to the SSL Plus API” on page 4. All the installable object functions are documented in the **SSL Plus Programmer’s Reference.**

Changes to the SSL Plus API

Note that the naming convention has changed in this version of SSL Plus: `SSLSetCipherSuites()` is now `ssl_SetCipherSuites()`; `TLSSetAlertFunc()` is now `ssl_SetAlertFunc()`etc.

See "Function Mappings" on page 12 for a list of the API changes between SSL Plus 3.x for Desktop Systems (1.x for Embedded Systems) and version 4.0.

Global and Connection Contexts

Four functions have been added to the SSL Plus API to support global and connection contexts. They are `ssl_CreateGlobalContext()`, `ssl_DestroyGlobalContext()`, `ssl_CreateConnectionContext()` and `ssl_DestroyConnectionContext()`. These functions replace those which supported the SSL and TLS contexts in previous versions of SSL Plus.

For more information on the global and connection contexts, see the **SSL Plus User's Guide** and the **SSL Plus Programmer's Reference**.

Installable Objects

There are seven categories of installable objects:

- Cipher suite objects
- Protocol objects (see the table below)
- Client authentication mode objects
- Private key decryption objects
- Certificate format objects
- Key and certificate decoding objects
- Mobile Trust root objects

The SSL Plus API provides functions to install each of these objects. The `SSLInstallModules()` function has been removed. See the **SSL Plus Programmer's Reference** for more information.

Table 1 shows the SSL protocol enumerated types in SSL Plus v3.x and the

Table 1: Mapping of SSL Plus v3.x protocol enumerated types and SSL Plus v4.x protocol objects

SSL Plus v3.x	SSL Plus v4.x
SSL_Version_Undetermined	**SSL_PROTOCOL_TLSV1_SSLV3_SSLV2_(CLIENTSIDE/SERVERSIDE)**
TLS_Version_1_0_With_2_0_Hello	**SSL_PROTOCOL_TLSV1_SSLV3_SSLV2_(CLIENTSIDE/SERVERSIDE)**
SSL_Version_3_0_With_2_0_Hello	**SSL_PROTOCOL_SSLV3_SSLV2_(CLIENTSIDE/SERVERSIDE)**
SSL_Version_3_0	**SSL_PROTOCOL_SSLV3_SSLV2_(CLIENTSIDE/SERVERSIDE)**
SSL_Version_3_0_Only	**SSL_PROTOCOL_SSLV3_(CLIENTSIDE/SERVERSIDE)**

Table 1: Mapping of SSL Plus v3.x protocol enumerated types and SSL Plus v4.x protocol objects

SSL Plus v3.x	SSL Plus v4.x
TLS_Version_1_0_Only	**SSL_PROTOCOL_TLSV1_(CLIENTSIDE /SERVERSIDE)**
SSL_Version_2_0	**SSL_PROTOCOL_SSLV2_(CLIENTSIDE /SERVERSIDE)**
TLS_Version_1_0	**SSL_PROTOCOL_TLSV1_SSLV3_(CLIENTSIDE/SERVERSIDE)**

corresponding SSL protocol objects in SSL Plus v4.x. The name of each protocol object indicates which SSL protocol(s) it supports. For example, **`SSL_PROTOCOL_TLSV1_SSLV3_CLIENTSIDE`** supports both the TLSv1.0 and SSLv3 protocols (on the client side).

Handshake and Data Exchange

The **`ssl_RequestRenegotiation()`** function only requests a renegotiation of the handshake. In previous versions of SSL Plus (Desktop only) if you called this function from a client application it would also initiate the handshake. In this version you must call **`ssl_Handshake()`** following this function to complete the handshake.

The **`ssl_Read()`** and **`ssl_Write()`** functions no longer initiate a handshake if called before **`ssl_Handshake()`**. This marks a change from previous releases of SSL Plus (Desktop only).

A new function has been added that allows you to free memory on idle connections. When a connection context is created, the library allocates memory for the read and write buffers. When the context is destroyed, the library frees the memory. This scheme reduces heap fragmentation. **`ssl_FreeRecordBuffers()`** allows you free the read and write buffers on idle connections. If the buffers contain data, they are not freed. When data transfer resumes, the library re-allocates the buffers.

Connection Status

Three new functions have been added. They are **`ssl_GetContextProtocolVersion()`**, **`ssl_GetMasterSecret()`** and **`ssl_WasSessionResumed()`**.

`ssl_GetContextProtocolVersion()` allows you to determine which version of the SSL protocol is being negotiated during a handshake. This is useful in the certificate callback function where your application may need this information in order to carry out the appropriate validation. Note that the equivalent function to **`SSLGetProtocolVersion()`** and **`TLSGetProtocolVersion()`** is now **`ssl_GetNegotiatedProtocolVersion()`**. As in previous releases of SSL Plus, this function must be called after the handshake has been completed.

Session Resumption

No changes have been made to session resumption. As before, you must set all the session callbacks for the library to implement session resumption.

SSL Plus RAD API

The SSL Plus RAD (Rapid Application Development) API provides a shortcut for many of the most common operations in an SSL Plus application. With the RAD API you can

- Initialize a global context structure.
- Create a session database.
- Initialize a socket layer.
- Create a socket.
- Make a connection using a socket.
- Accept a connection using a socket.
- Close a socket.

The RAD API also provides default implementations of all the mandatory callback functions, as well as functions to check the validity of a certificate and manipulate entries in the session database.

The RAD API is optional; you can still develop your application using the standard SSL Plus API alone.

See the **SSL Plus Programmer's Reference** for a complete list of the RAD API functions. See the sample applications (sslsampleclient.c and sslsampleserver.c) for an illustration of how to use the functions.

Restrictions on the use of the SSL Plus API

A number of restrictions have been placed on the API to prevent the context structures from being modified during an SSL connection. These restrictions are in place because SSL Plus does not carry out any form of locking (using mutexes or semaphores) in multi-threaded applications.

The SSL Plus library provides an application framework which, if used in your multi-threaded application, should allow you to keep the number of mutexes required by SSL Plus to a minimum. This is described in this section.

Multi-threaded applications can avoid the use of mutexes with respect to managing the SSL Plus global and connection context structures if they allocate the job of creating and configuring the global context to a single thread. If this same thread is also responsible for creating connection contexts (using **`ssl_CreateConnectionContext()`**) then no mutexes are needed.

Multi-threaded applications need to protect the functions listed in the **Needs Mutex** column of Tables 2 and 3 with mutexes whenever more than one thread might call any of these functions.

Multi-threaded applications which need multiple global contexts can follow the above paradigm for a single global context, but manage the creation of contexts in separate threads (taking care to protect the **`ssl_CreateConnectionContext()`** function with a mutex, as described below).

The sections below describe the more complicated cases where you may wish to access the **same** global or connection context from **multiple** threads.

Modifying the global context in a multi-threaded application

Most SSL Plus applications only need to create a single global context. Once you configure the global context, we recommend you do not change it (with the exception of calling **`ssl_GenerateRSAExportKeyPair()`**). This context can be used to create all future SSL connections. Each connection created from a global context has read-only access to that context. During the lifetime of the connection the library reads the global objects (protocol engine, cipher suites, local identities etc) configured in the global context. It is possible to change the global context configuration after the SSL connections have been created, however there are restrictions on when you can safely do so. Multi-threaded applications may also have to implement a mutex locking and unlocking mechanism to serialize access to the global context. Table 2 describes the restrictions that have been placed on the functions which modify the global context.

Creating multiple connection contexts in a multi-threaded application

Most SSL Plus applications should only create new SSL connections from a single thread. If two or more threads try to create a new SSL connection, you must ensure that calls to **`ssl_CreateConnectionContext()`** are serialized. This is usually done by enclosing the call to the function with a locking/unlocking mechanism. If only a single thread is used to create all connections, then no locking is needed.

Accessing a single connection context in a multi-threaded application

Most SSL Plus applications should only use a single thread per connection context. If you have multiple threads which access the same context, you must serialize access to the context. If only a single thread accesses the connection context, no locking is needed. Table 3 describes the restrictions that have been placed on the functions which modify the connection context.

Note that in multi-threaded applications your callback functions must also be thread-safe and re-entrant. For example, in the session database functions (which are used for session resumption) you must serialize access to the session database.

The column headings for Table 2 are described below.

- **Needs mutex -** If your application is multi-threaded, you must prevent the global context from being modified by two threads at the same time. All the functions which are marked with an "X" must be enclosed with a lock/unlock mechanism to prevent them from being called simultaneously. Functions that are not marked with an "X" do not need to be protected in this way.

- **Anytime -** Functions that are marked with an "X" can be called at any time.

- **No existing connections -** Functions that are marked with an "X" cannot be called while there exist connection contexts that have been created from this global context.

- **No connections handshaking -** Functions that are marked with an "X" cannot be called while any connection contexts created from this global context are being used to negotiate (or renegotiate) a handshake.

- **No connections reading data -** Functions that are marked with an "X" cannot be called while any connection contexts created from this global context are being used to read data with `ssl_Read()`.

Table 2: Restrictions on functions that modify the global context

	Needs mutex[a]	Anytime	No existing connections	No connections h/dshaking	No connections reading data
ssl_CreateGlobalContext()		**X**			
ssl_DestroyGlobalContext()	**X**		**X**		
ssl_SetProtocol()	**X**		**X**		
ssl_SetCryptographicStrength()	**X**			**X**	
ssl_SetClientAuthModes()	**X**			**X**	

Table 2: Restrictions on functions that modify the global context

	Needs mutex[a]	Anytime	No existing connections	No connections h/dshaking	No connections reading data
ssl_SetCipherSuites()	**X**			**X**	
ssl_SetSessionExpiryTime()	**X**			**X**	
ssl_SetIOSemantics()	**X**				**X**
ssl_CreateCertList()		**X**			
ssl_AddCertificate()		**X**			
ssl_GenerateRSAExportKeyPair()	**X**				
ssl_AddIdentity()	**X**			**X**	
ssl_AddTrustedCerts()	**X**			**X**	
ssl_AddTrustedRoots()		**X**			
ssl_DestroyCertList()		**X**			
ssl_CreateConnectionContext()	**X**				
ssl_SetIOFuncs()	**X**		**X**		
ssl_SetAlertFunc()	**X**		**X**		
ssl_SetSessionFuncs()	**X**		**X**		
ssl_SetCheckCertificateChainFunc ()	**X**		**X**		
ssl_SetPRNG()	**X**		**X**		
ssl_SetSurrenderFunc()	**X**		**X**		
ssl_SetLogFunc()	**X**		**X**		
ssl_GenerateRandomSeed()		**X**			
ssl_ExtractCertificateNameItem()		**X**			
ssl_ExtractRawCertData()		**X**			
ssl_GetVersion()		**X**			
ssl_DecodeRecord()		**X**			

a. See the description under “Restrictions on the use of the SSL Plus API”.

The column headings for Table 3 are described below.

- **Anytime -** The functions marked with an "X" can be called at any time.

- **Needs mutex -** If your application is multi-threaded, you must prevent the connection context from being modified by two threads at the same time. All the functions which are marked with an "X" must be enclosed with a lock/unlock mechanism to prevent them from being called simultaneously. Functions that are not marked with an "X" do not need to be protected in this way.

Table 3: Restrictions on functions that modify the connection context

	Anytime	Needs mutex[a]
ssl_DestroyConnectionContext()		**X**
ssl_Handshake()		**X**
ssl_RequestRenegotiation()		**X**
ssl_Read()		**X**
ssl_Write()		**X**
ssl_GetReadPendingSize()		**X**
ssl_GetWritePendingSize()		**X**
ssl_ServiceWriteQueue()		**X**
ssl_Close()		**X**
ssl_GetNegotiatedProtocolVersion()	**X**	
ssl_GetContextProtocolVersion()	**X**	
ssl_GetNegotiatedCipher()	**X**	
ssl_GetMasterSecret()	**X**	
ssl_WasSessionResumed()	**X**	
ssl_FreeRecordBuffers()		**X**

a. See the description under “Restrictions on the use of the SSL Plus API”.

Backward Compatibility Issues

- SSL V4.0 client applications will not handshake correctly with SSL Plus Desktop server applications that use ECC cipher suites. SSL V4.0 server applications are unaffected; they will be able to handshake with older client applications (Desktop and Embedded).

- Client applications built with SSL Plus Desktop V3.1.2 (or earlier) that use the protocol **`SSL_Version_Undetermined`**, or **`SSL_Version_3_0_With_2_0_Hello`**, or **`TLS_Version_1_0_With_2_0_Hello`**, or **`SSL_Version_3_0`** and that use either the **`TLS_RSA_WITH_RC4_128_MD5`** or **`TLS_RSA_EXPORT_WITH_RC40_40_MD5`** cipher suite will not be able to handshake with an SSL V4.0 server that uses the protocol **`SSL_PROTOCOL_SSLV2_SERVERSIDE`**.

You may wish to contact Certicom technical support at the phone numbers below.

1-800-511-8011 (Voice)

1-800-474-3877 (Fax)

email: support@certicom.com

Function Mappings

Table 4: Mapping of SSL Plus Desktop API to SSL Plus v4.0 API

SSL Plus v3.x (Desktop)	SSL Plus v4.0
SSLInstallModules()	N/A
SSLSetGlobalOption()	N/A
SSLInitialize()	ssl_CreateGlobalContext(), ssl_CreateConnectionContext()
SSLContextSize()	Not required.
SSLInitContext()	ssl_CreateGlobalContext(), ssl_CreateConnectionContext()
SSLDeleteContext()	ssl_DestroyGlobalContext(), ssl_DestroyConnectionContext()
SSLDuplicateContext()	ssl_CreateConnectionContext()
SSLDuplicateChildContext()	ssl_CreateConnectionContext()
SSLSetProtocol()	ssl_SetProtocol()
SSLSetProtocolVersion()	ssl_SetProtocol()
SSLSetPrivateKey()	ssl_CreateCertList()
SSLSetExportPrivateKey()	ssl_GenerateRSAExportKeyPair()
SSLSetDHParams()	N/A
SSLSetRequestClientCert()	ssl_SetClientAuthModes()
SSLAddCertificate()	ssl_AddCertificate(), ssl_AddIdentity()
SSLAddDistinguishedName()	N/A
SSLAddTrustedCertificate()	ssl_AddCertificate(), ssl_AddTrustedCerts()
SSLSetPeerID()	ssl_CreateConnectionContext()
SSLSetRandomFunc()	ssl_SetPRNG()
SSLSetRandomRef()	ssl_CreateConnectionContext()
SSLSetSurrenderFunc()	ssl_SetSurrenderFunc()
SSLSetSurrenderRef()	ssl_CreateConnectionContext()
SSLSetReadFunc()	ssl_SetIOFuncs()
SSLSetWriteFunc()	ssl_SetIOFuncs()
SSLSetIORef()	ssl_CreateConnectionContext()

Table 4: Mapping of SSL Plus Desktop API to SSL Plus v4.0 API

SSL Plus v3.x (Desktop)	SSL Plus v4.0
SSLSetAddSessionFunc()	ssl_SetSessionFuncs()
SSLSetGetSessionFunc()	ssl_SetSessionFuncs()
SSLSetDeleteSessionFunc()	ssl_SetSessionFuncs()
SSLSetSessionRef()	ssl_CreateConnectionContext()
SSLSetCheckCertificateFunc()	ssl_SetCheckCertificateChainFunc()
SSLSetCheckCertificateRef()	ssl_CreateConnectionContext()
SSLSetCheckCertificateChainFunc()	ssl_SetCheckCertificateChainFunc()
SSLSetCipherSuites()	ssl_SetCipherSuites()
SSLSetCryptographicStrength()	ssl_SetCryptographicStrength()
SSLSetCipherNotifyFunc()	N/A
SSLSetCipherNotifyRef()	N/A
SSLSetMaximumRecordLength()	ssl_CreateConnectionContext()
SSLGetProtocolVersion()	ssl_GetNegotiatedProtocolVersion(), ssl_GetContextProtocolVersion()
SSLGetPeerCertificateChainLength()	Get from certificate chain callback function.
SSLGetPeerCertificate()	Call ssl_ExtractRawCertData() from the certificate chain callback function to get the raw certificate data.
SSLGetPeerCertificateRef()	N/A
SSLGetTrustedCertificateRef()	N/A
SSLGetNegotiatedCipher()	ssl_GetNegotiatedCipher()
SSLGetWritePendingSize()	ssl_GetWritePendingSize()
SSLGetReadPendingSize()	ssl_GetReadPendingSize()
SSLHandshake()	ssl_Handshake()
SSLServiceWriteQueue()	ssl_ServiceWriteQueue()
SSLWrite()	ssl_Write()
SSLRead()	ssl_Read()
SSLClose()	ssl_Close()
SSLSetIOSemantics()	ssl_SetIOSemantics()
SSLRequestRenegotiation()	ssl_RequestRenegotiation()
SSLCallSurrenderFunc()	N/A

Table 4: Mapping of SSL Plus Desktop API to SSL Plus v4.0 API

SSL Plus v3.x (Desktop)	SSL Plus v4.0
SSLExtractSubjectDNField()	ssl_ExtractCertificateNameItem()
SSLCountSubjectDNFields()	ssl_ExtractCertificateNameItem()
SSLExtractSubjectDNFieldIndex()	ssl_ExtractCertificateNameItem()
SSLExtractExtension()	N/A
SSLCountExtensions()	N/A
SSLExtractExtensionIndex()	N/A
SSLExtractValidityDates()	N/A
SSLExtractSerialNumber()	N/A
SSLFreeSerialNumber()	N/A
SSLFreeExtension()	N/A
SSLFreeAVA()	N/A
SSLAddOIDValue()	N/A
SSLSetChainNum()	N/A
SSLLoadLocalIdentity()	ssl_AddCertificate(), ssl_AddIdentity()[a]
SSLLoadTrustedCertificateFile()	ssl_AddCertificate(), ssl_AddTrustedCerts()[a]
SSLEncodeCertificateRequest()	N/A
SSLFormatCertificateRequest()	N/A
SSLGeneratePrivateKey()	N/A
SSLGenerateExportPrivateKey()	ssl_GenerateRSAExportPrivateKey()
SSLFormatPrivateKey()	N/A
SSLConstructPrivateKey()	N/A
SSLGetCipherSuiteInfo()	N/A
SSLGetIndexedCiphersuiteInfo()	N/A
SSLCiphersuiteNameToNumber()	N/A
SSLGetCiphersuiteCount()	N/A
SSLGetDefaultCipherSuites()	N/A
SSLAddTrustedServerCertificate()	ssl_AddTrustedCerts()
SSLGetTrustedServerCertificate()	N/A

Table 4: Mapping of SSL Plus Desktop API to SSL Plus v4.0 API

SSL Plus v3.x (Desktop)	SSL Plus v4.0
SSLPregenerateECDHClientParameters()	**N/A**
SSLGetLibVersion()	**ssl_GetVersion()**
SSLGetErrorStr	**N/A**

a. SSL Plus version 4.0 provides functions to add certificates and keys as the local identity or trusted CAs, but not functions which perform file I/O.

The following table describes the mapping of the SSL Plus v3.x Desktop API to the SSL Plus v4.0 RAD API. Calls to the functions in the left-hand column can be replaced by a **single** call to **one** of the RAD functions in the right-hand column.

Table 5: Mapping of SSL Plus Desktop API to SSL Plus v4.0 RAD API

SSL Plus v3.x (Desktop)	SSL Plus v4.0 RAD API

SSLInitialize()
SSLInitContext()
SSLDuplicateContext()
SSLDuplicateChildContext()
SSLSetProtocol()
SSLSetProtocolVersion()
SSLSetRandomFunc()
SSLSetSurrenderFunc()
SSLSetReadFunc()
SSLSetWriteFunc()
SSLSetAddSessionFunc()
SSLSetGetSessionFunc()
SSLSetDeleteSessionFunc()
SSLSetCheckCertificateFunc()
SSLSetCipherSuites()
	sslrad_InitializeCompleteServerGlobalContext()
	sslrad_InitializeECCOnlyServerGlobalContext()
	sslrad_InitializeRSAOnlyServerGlobalContext()
	sslrad_InitializeCompleteNoClientAuthServerGlobalContext()
	sslrad_InitializeECCOnlyNoClientAuthServerGlobalContext()
	sslrad_InitializeRSAOnlyNoClientAuthServerGlobalContext()
	sslrad_InitializeCompleteClientGlobalContext()
	sslrad_InitializeECCOnlyClientGlobalContext()
	sslrad_InitializeRSAOnlyClientGlobalContext()
	sslrad_InitializeCompleteNoClientAuthClientGlobalContext()
	sslrad_InitializeECCOnlyNoClientAuthClientGlobalContext()
	sslrad_InitializeRSAOnlyNoClientAuthClientGlobalContext()

SSLSetRequestClientCert()	**sslrad_InitializeCompleteServerGlobalContext()**
	sslrad_InitializeECCOnlyServerGlobalContext()
	sslrad_InitializeRSAOnlyServerGlobalContext()

Table 6: Mapping of SSL Plus Embedded API to SSL Plus v4.0 API

SSL Plus v1.x (Embedded)	SSL Plus v4.0
TLSContextSize()	**Not required.**
TLSInitContext()	**ssl_CreateGlobalContext(), ssl_CreateConnectionContext()**
TLSEnableNonBlocking()	**ssl_SetIOSemantics()**
TLSDeleteContext()	**ssl_DestroyGlobalContext(), ssl_DestroyConnectionContext()**
TLSSetCipherSuites()	**ssl_SetCipherSuites()**
TLSSetProtocolVersion()	**ssl_SetProtocol()**
TLSSetCryptographicStrength()	**ssl_SetCryptographicStrength()**
TLSGenerateKeyAgreementParams()	**N/A**
TLSAddECDSACertificate()	**ssl_AddCertificate(), ssl_AddIdentity()**
TLSAddRSACertificate()	**ssl_AddCertificate(), ssl_AddIdentity()**
TLSAddTrustedCertificate()	**ssl_AddCertificate(), ssl_AddTrustedCerts()**
TLSAddTrustedCertificateNoCopy()	**ssl_AddCertificate(), ssl_AddTrustedCerts()**
TLSSetECDSAPrivateKey()	**ssl_CreateCertList(), ssl_AddIdentity()**
TLSSetRSAPrivateKey()	**ssl_CreateCertList(), ssl_AddIdentity()**
TLSSetSessionExpiryTime()	**ssl_SetSessionExpiryTime()**
TLSSetPeerID()	**ssl_CreateConnectionContext()**
TLSDeriveARC4Key()	**N/A**
TLSSetPwdECDSAPrivateKey()	**ssl_CreateCertList(), ssl_AddIdentity()**
TLSSetPwdRSAPrivateKey()	**ssl_CreateCertList(), ssl_AddIdentity()**
TLSExtractCertificateNameItem()	**ssl_ExtractCertificateNameItem()**
TLSHandshake()	**ssl_Handshake()**
TLSGetNegotiatedCipher()	**ssl_GetNegotiatedCipher()**
TLSGetProtocolVersion()	**ssl_GetNegotiatedProtocolVersion(), ssl_GetContextProtocolVersion()**
TLSClose()	**ssl_Close()**
TLSWrite()	**ssl_Write()**
TLSWriteNoCopy()	**N/A**
TLSRead()	**ssl_Read()**

Table 6: Mapping of SSL Plus Embedded API to SSL Plus v4.0 API

SSL Plus v1.x (Embedded)	SSL Plus v4.0
TLSReadNoCopy()	**N/A**
TLSNoCopyBufferSize()	**N/A**
TLSPeekNextReadLength()	**ssl_GetReadPendingSize()**
TLSLibClose()	**N/A**
TLSLibVersion()	**ssl_GetVersion()**
TLSSetOption()	**N/A**
TLSSetRandomFunc()	**ssl_SetPRNG()**
TLSSetRandomRef()	**ssl_CreateConnectionContext()**
TLSRandomCallback()	**ssl_Default_RandomCallback()**
TLSGenerateRandom()	**ssl_GenerateRandomSeed()**
TLSSetIORef()	**ssl_CreateConnectionContext()**
TLSSetReadFunc()	**ssl_SetIOFuncs()**
TLSSetWriteFunc()	**ssl_SetIOFuncs()**
TLSSetAlertFunc()	**ssl_SetAlertFunc()**
TLSSetAlertRef()	**ssl_CreateConnectionContext()**
TLSSetAddSessionFunc()	**ssl_SetSessionFuncs()**
TLSSetSessionRef()	**ssl_CreateConnectionContext()**
TLSSetGetSessionFunc()	**ssl_SetSessionFuncs()**
TLSSetDeleteSessionFunc()	**ssl_SetSessionFuncs()**
TLSSetCheckCertificateChainFunc()	**ssl_SetCheckCertificateChainFunc()**
TLSSetCheckCertificateChainRef()	**ssl_CreateConnectionContext()**

The following table describes the mapping of the SSL Plus v1.x Embedded API to the SSL Plus v4.0 RAD API. Calls to the functions in the left-hand column can be replaced by a **single** call to **one** of the RAD functions in the right-hand column

Table 7: Mapping of SSL Plus Embedded API to SSL Plus v4.0 RAD API

SSL Plus v1.x (Embedded)	SSL Plus v4.0 RAD API
TLSInitContext()	**sslrad_InitializeCompleteClientGlobalContext()**
TLSSetProtocolVersion()	**sslrad_InitializeECCOnlyClientGlobalContext()**
TLSSetRandomFunc()	**sslrad_InitializeRSAOnlyClientGlobalContext()**
TLSSetReadFunc()	**sslrad_InitializeCompleteNoClientAuthClientGlobalContext()**
TLSSetWriteFunc()	**sslrad_InitializeECCOnlyNoClientAuthClientGlobalContext()**
TLSSetAddSessionFunc()	**sslrad_InitializeRSAOnlyNoClientAuthClientGlobalContext()**
TLSSetGetSessionFunc()	
TLSSetDeleteSessionFunc()	
TLSSetCheckCertificateFunc()	
TLSSetCipherSuites()	