NAME

migration_guide − OpenSSL migration guide

SYNOPSIS

See the individual manual pages for details.

DESCRIPTION

This guide details the changes required to migrate to new versions of OpenSSL. Currently this covers OpenSSL 3.0. For earlier versions refer to <https://github.com/openssl/openssl/blob/master/CHANGES.md>. For an overview of some of the key concepts introduced in OpenSSL 3.0 see **crypto**(7).

OPENSSL 3.0

Main Changes from OpenSSL 1.1.1

Major Release

OpenSSL 3.0 is a major release and consequently any application that currently uses an older version of OpenSSL will at the very least need to be recompiled in order to work with the new version. It is the intention that the large majority of applications will work unchanged with OpenSSL 3.0 if those applications previously worked with OpenSSL 1.1.1. However this is not guaranteed and some changes may be required in some cases. Changes may also be required if applications need to take advantage of some of the new features available in OpenSSL 3.0 such as the availability of the FIPS module.

License Change

In previous versions, OpenSSL was licensed under the dual OpenSSL and SSLeay licenses <https://www.openssl.org/source/license-openssl-ssleay.txt> (both licenses apply). From OpenSSL 3.0 this is replaced by the Apache License v2 <https://www.openssl.org/source/apache-license-2.0.txt>.

Providers and FIPS support

One of the key changes from OpenSSL 1.1.1 is the introduction of the Provider concept. Providers collect together and make available algorithm implementations. With OpenSSL 3.0 it is possible to specify, either programmatically or via a config file, which providers you want to use for any given application. OpenSSL 3.0 comes with 5 different providers as standard. Over time third parties may distribute additional providers that can be plugged into OpenSSL. All algorithm implementations available via providers are accessed through the ''high level'' APIs (for example those functions prefixed with `EVP`). They cannot be accessed using the ''Low Level APIs''.

One of the standard providers available is the FIPS provider. This makes available FIPS validated cryptographic algorithms. The FIPS provider is disabled by default and needs to be enabled explicitly at configuration time using the `enable-fips` option. If it is enabled, the FIPS provider gets built and installed in addition to the other standard providers. No separate installation procedure is necessary. There is however a dedicated `install_fips` make target, which serves the special purpose of installing only the FIPS provider into an existing OpenSSL installation.

Not all algorithms may be available for the application at a particular moment. If the application code uses any digest or cipher algorithm via the EVP interface, the application should verify the result of the **EVP_EncryptInit**(3), **EVP_EncryptInit_ex**(3), and **EVP_DigestInit**(3) functions. In case when the requested algorithm is not available, these functions will fail.

See also ''Legacy Algorithms'' for information on the legacy provider.

See also ''Completing the installation of the FIPS Module'' and ''Using the FIPS Module in applications''.

Low Level APIs

OpenSSL has historically provided two sets of APIs for invoking cryptographic algorithms: the ''high level'' APIs (such as the `EVP` APIs) and the ''low level'' APIs. The high level APIs are typically designed to work across all algorithm types. The ''low level'' APIs are targeted at a specific algorithm implementation. For example, the EVP APIs provide the functions **EVP_EncryptInit_ex**(3), **EVP_EncryptUpdate**(3) and **EVP_EncryptFinal**(3) to perform symmetric encryption. Those functions can be used with the algorithms AES, CHACHA, 3DES etc. On the other hand, to do AES encryption using

the low level APIs you would have to call AES specific functions such as **AES_set_encrypt_key**(3), **AES_encrypt**(3), and so on. The functions for 3DES are different. Use of the low level APIs has been informally discouraged by the OpenSSL development team for a long time. However in OpenSSL 3.0 this is made more formal. All such low level APIs have been deprecated. You may still use them in your applications, but you may start to see deprecation warnings during compilation (dependent on compiler support for this). Deprecated APIs may be removed from future versions of OpenSSL so you are strongly encouraged to update your code to use the high level APIs instead.

This is described in more detail in ''Deprecation of Low Level Functions''

Legacy Algorithms

Some cryptographic algorithms such as **MD2** and **DES** that were available via the EVP APIs are now considered legacy and their use is strongly discouraged. These legacy EVP algorithms are still available in OpenSSL 3.0 but not by default. If you want to use them then you must load the legacy provider. This can be as simple as a config file change, or can be done programmatically. See**OSSL_PR OVIDER−legacy**(7) for a complete list of algorithms. Applications using the EVP APIs to access these algorithms should instead use more modern algorithms. If that is not possible then these applications should ensure that the legacy provider has been loaded. This can be achieved either programmatically or via configuration. See **crypto**(7) man page for more information about providers.

Engines and "METHOD" APIs

The refactoring to support Providers conflicts internally with the APIs used to support engines, including the ENGINE API and any function that creates or modifies custom ''METHODS'' (for example **EVP_MD_meth_new**(3), **EVP_CIPHER_meth_new**(3), **EVP_PKEY_meth_new**(3), **RSA_meth_new**(3), **EC_KEY_METHOD_new**(3), etc.). These functions are being deprecated in OpenSSL 3.0, and users of these APIs should know that their use can likely bypass provider selection and configuration, with unintended consequences. This is particularly relevant for applications written to use the OpenSSL 3.0 FIPS module, as detailed below. Authors and maintainers of external engines are strongly encouraged to refactor their code transforming engines into providers using the new Provider API and avoiding deprecated methods.

Support of legacy engines

If openssl is not built without engine support or deprecated API support, engines will still work. However, their applicability will be limited.

New algorithms provided via engines will still work.

Engine-backed keys can be loaded via custom **OSSL_STORE** implementation. In this case the **EVP_PKEY** objects created via **ENGINE_load_private_key**(3) will be concidered legacy and will continue to work.

To ensure the future compatibility, the engines should be turned to providers. To prefer the provider-based hardware offload, you can specify the default properties to prefer your provider.

Versioning Scheme

The OpenSSL versioning scheme has changed with the OpenSSL 3.0 release. The new versioning scheme has this format:

MAJOR.MINOR.PATCH

For OpenSSL 1.1.1 and below, different patch levels were indicated by a letter at the end of the release version number. This will no longer be used and instead the patch level is indicated by the final number in the version. A change in the second (MINOR) number indicates that new features may have been added. OpenSSL versions with the same major number are API and ABI compatible. If the major number changes then API and ABI compatibility is not guaranteed.

For more information, see **OpenSSL_version**(3).

Other major new features

Certificate Management Protocol (CMP, RFC 4210)

This also covers CRMF (RFC 4211) and HTTP transfer (RFC 6712) See **openssl-cmp** (1) and **OSSL_CMP_exec_certreq** (3) as starting points.

HTTP(S) client

A proper HTTP(S) client that supports GET and POST, redirection, plain and ASN.1-encoded contents, proxies, and timeouts.

Key Derivation Function API (EVP_KDF)

This simplifies the process of adding new KDF and PRF implementations.

Previously KDF algorithms had been shoe-horned into using the EVP_PKEY object which was not a logical mapping. Existing applications that use KDF algorithms using EVP_PKEY (scrypt, TLS1 PRF and HKDF) may be slower as they use an EVP_KDF bridge internally. All new applications should use the new **EVP_KDF** (3) interface. See also ''Key Derivation Function (KDF)'' in **OSSL_PROVIDER-default** (7) and ''Key Derivation Function (KDF)'' in **OSSL_PROVIDER-FIPS** (7).

Message Authentication Code API (EVP_MAC)

This simplifies the process of adding MAC implementations.

This includes a generic EVP_PKEY to EVP_MAC bridge, to facilitate the continued use of MACs through raw private keys in functionality such as **EVP_DigestSign** (3) and **EVP_DigestVerify** (3).

All new applications should use the new **EVP_MAC** (3) interface. See also ''Message Authentication Code (MAC)'' in **OSSL_PROVIDER-default** (7) and ''Message Authentication Code (MAC)'' in **OSSL_PROVIDER-FIPS** (7).

Support for Linux Kernel TLS

In order to use KTLS, support for it must be compiled in using the `enable-ktls` configuration option. It must also be enabled at run time using the **SSL_OP_ENABLE_KTLS** option.

New Algorithms

- KDF algorithms ''SINGLE STEP'' and ''SSH''

  See **EVP_KDF-SS** (7) and **EVP_KDF-SSHKDF** (7)

- MAC Algorithms ''GMAC'' and ''KMAC''

  See **EVP_MAC-GMAC** (7) and **EVP_MAC-KMAC** (7).

- KEM Algorithm ''RSASVE''

  See **EVP_KEM-RSA** (7).

- Cipher Algorithm ''AES-SIV''

  See ''SIV Mode'' in **EVP_EncryptInit** (3).

- AES Key Wrap inverse ciphers supported by EVP layer.

  The inverse ciphers use AES decryption for wrapping, and AES encryption for unwrapping. The algorithms are: ''AES-128-WRAP-INV'', ''AES-192-WRAP-INV'', ''AES-256-WRAP-INV'', ''AES-128-WRAP-PAD-INV'', ''AES-192-WRAP-PAD-INV'' and ''AES-256-WRAP-PAD-INV''.

- CTS ciphers added to EVP layer.

  The algorithms are ''AES-128-CBC-CTS'', ''AES-192-CBC-CTS'', ''AES-256-CBC-CTS'', ''CAMELLIA-128-CBC-CTS'', ''CAMELLIA-192-CBC-CTS'' and ''CAMELLIA-256-CBC-CTS''. CS1, CS2 and CS3 variants are supported.

CMS and PKCS#7 updates

- Added CAdES-BES signature verification support.
- Added CAdES-BES signature scheme and attributes support (RFC 5126) to CMS API.

- Added AuthEnvelopedData content type structure (RFC 5083) using AES_GCM

  This uses the AES-GCM parameter (RFC 5084) for the Cryptographic Message Syntax. Its purpose is to support encryption and decryption of a digital envelope that is both authenticated and encrypted using AES GCM mode.

- **PKCS7_get_octet_string** (3) and **PKCS7_type_is_other** (3) were made public.

PKCS#12 API updates

The default algorithms for pkcs12 creation with the **PKCS12_create()** function were changed to more modern PBKDF2 and AES based algorithms. The default MAC iteration count was changed to PKCS12_DEFAULT_ITER to make it equal with the password-based encryption iteration count. The default digest algorithm for the MAC computation was changed to SHA−256. The pkcs12 application now supports −legacy option that restores the previous default algorithms to support interoperability with legacy systems.

Added enhanced PKCS#12 APIs which accept a library context **OSSL_LIB_CTX** and (where relevant) a property query. Other APIs which handle PKCS#7 and PKCS#8 objects have also been enhanced where required. This includes:

PKCS12_add_key_ex (3), **PKCS12_add_safe_ex** (3), **PKCS12_add_safes_ex** (3), **PKCS12_create_ex** (3), **PKCS12_decrypt_skey_ex** (3), **PKCS12_init_ex** (3), **PKCS12_item_decrypt_d2i_ex** (3), **PKCS12_item_i2d_encrypt_ex** (3), **PKCS12_key_gen_asc_ex** (3), **PKCS12_key_gen_uni_ex** (3), **PKCS12_key_gen_utf8_ex** (3), **PKCS12_pack_p7encdata_ex** (3), **PKCS12_pbe_crypt_ex** (3), **PKCS12_PBE_keyivgen_ex** (3), **PKCS12_SAFEBAG_create_pkcs8_encrypt_ex** (3), **PKCS5_pbe2_set_iv_ex** (3), **PKCS5_pbe_set0_algor_ex** (3), **PKCS5_pbe_set_ex** (3), **PKCS5_pbkdf2_set_ex** (3), **PKCS5_v2_PBE_keyivgen_ex** (3), **PKCS5_v2_scrypt_keyivgen_ex** (3), **PKCS8_decrypt_ex** (3), **PKCS8_encrypt_ex** (3), **PKCS8_set0_pbe_ex** (3).

As part of this change the EVP_PBE_xxx APIs can also accept a library context and property query and will call an extended version of the key/IV derivation function which supports these parameters. This includes **EVP_PBE_CipherInit_ex** (3), **EVP_PBE_find_ex** (3) and **EVP_PBE_scrypt_ex** (3).

Windows thread synchronization changes

Windows thread synchronization uses read/write primitives (SRWLock) when supported by the OS, otherwise CriticalSection continues to be used.

Trace API

A new generic trace API has been added which provides support for enabling instrumentation through trace output. This feature is mainly intended as an aid for developers and is disabled by default. To utilize it, OpenSSL needs to be configured with the `enable-trace` option.

If the tracing API is enabled, the application can activate trace output by registering BIOs as trace channels for a number of tracing and debugging categories. See **OSSL_trace_enabled** (3).

Key validation updates

EVP_PKEY_public_check (3) and **EVP_PKEY_param_check** (3) now work for more key types. This includes RSA, DSA, ED25519, X25519, ED448 and X448. Previously (in 1.1.1) they would return −2. For key types that do not have parameters then **EVP_PKEY_param_check** (3) will always return 1.

Other notable deprecations and changes

The function code part of an OpenSSL error code is no longer relevant

This code is now always set to zero. Related functions are deprecated.

STACK and HASH macros have been cleaned up

The type-safe wrappers are declared everywhere and implemented once. See **DEFINE_STACK_OF** (3) and **DECLARE_LHASH_OF** (3).

The RAND_DRBG subsystem has been removed

The new **EVP_RAND**(3) is a partial replacement: the DRBG callback framework is absent. The RAND_DRBG API did not fit well into the new provider concept as implemented by EVP_RAND and EVP_RAND_CTX.

Removed **FIPS_mode()** and **FIPS_mode_set()**

These functions are legacy APIs that are not applicable to the new provider model. Applications should instead use **EVP_default_properties_is_fips_enabled**(3) and **EVP_default_properties_enable_fips**(3).

Key generation is slower

The Miller-Rabin test now uses 64 rounds, which is used for all prime generation, including RSA key generation. This affects the time for larger keys sizes.

The default key generation method for the regular 2−prime RSA keys was changed to the FIPS 186−4 B.3.6 method (Generation of Probable Primes with Conditions Based on Auxiliary Probable Primes). This method is slower than the original method.

Change PBKDF2 to conform to SP800−132 instead of the older PKCS5 RFC2898

This checks that the salt length is at least 128 bits, the derived key length is at least 112 bits, and that the iteration count is at least 1000. For backwards compatibility these checks are disabled by default in the default provider, but are enabled by default in the fips provider.

To enable or disable the checks see **OSSL_KDF_PARAM_PKCS5** in **EVP_KDF−PBKDF2**(7). The parameter can be set using **EVP_KDF_derive**(3).

Enforce a minimum DH modulus size of 512 bits

Smaller sizes now result in an error.

SM2 key changes

EC EVP_PKEYs with the SM2 curve have been reworked to automatically become EVP_PKEY_SM2 rather than EVP_PKEY_EC.

Unlike in previous OpenSSL versions, this means that applications cannot call `EVP_PKEY_set_alias_type(pkey, EVP_PKEY_SM2)`to get SM2 computations.

Parameter and key generation is also reworked to make it possible to generate EVP_PKEY_SM2 parameters and keys. Applications must now generate SM2 keys directly and must not create an EVP_PKEY_EC key first. It is no longer possible to import an SM2 key with domain parameters other than the SM2 elliptic curve ones.

Validation of SM2 keys has been separated from the validation of regular EC keys, allowing to improve the SM2 validation process to reject loaded private keys that are not conforming to the SM2 ISO standard. In particular, a private scalar *k* outside the range *1 <= k < n−1* is now correctly rejected.

EVP_PKEY_set_alias_type() method has been removed

This function made a **EVP_PKEY** object mutable after it had been set up. In OpenSSL 3.0 it was decided that a provided key should not be able to change its type, so this function has been removed.

Functions that return an internal key should be treated as read only

Functions such as **EVP_PKEY_get0_RSA**(3) behave slightly differently in OpenSSL 3.0. Previously they returned a pointer to the low-level key used internally by libcrypto. From OpenSSL 3.0 this key may now be held in a provider. Calling these functions will only return a handle on the internal key where the EVP_PKEY was constructed using this key in the first place, for example using a function or macro such as **EVP_PKEY_assign_RSA**(3), **EVP_PKEY_set1_RSA**(3), etc. Where the EVP_PKEY holds a provider managed key, then these functions now return a cached copy of the key. Changes to the internal provider key that take place after the first time the cached key is accessed will not be reflected back in the cached copy. Similarly any changes made to the cached copy by application code will not be reflected back in the internal provider key.

For the above reasons the keys returned from these functions should typically be treated as read-only. To

emphasise this the value returned from **EVP_PKEY_get0_RSA**(3), **EVP_PKEY_get0_DSA**(3), **EVP_PKEY_get0_EC_KEY**(3) and **EVP_PKEY_get0_DH**(3) have been made const. This may break some existing code. Applications broken by this change should be modified. The preferred solution is to refactor the code to avoid the use of these deprecated functions. Failing this the code should be modified to use a const pointer instead. The **EVP_PKEY_get1_RSA**(3), **EVP_PKEY_get1_DSA**(3), **EVP_PKEY_get1_EC_KEY**(3) and **EVP_PKEY_get1_DH**(3) functions continue to return a non-const pointer to enable them to be ''freed''. However they should also be treated as read-only.

The public key check has moved from **EVP_PKEY_derive()** to **EVP_PKEY_derive_set_peer()**

This may mean result in an error in **EVP_PKEY_derive_set_peer**(3) rather than during **EVP_PKEY_derive**(3). To disable this check use EVP_PKEY_derive_set_peer_ex(dh, peer, 0).

The print format has cosmetic changes for some functions

The output from numerous ''printing'' functions such as **X509_signature_print**(3), **X509_print_ex**(3), **X509_CRL_print_ex**(3), and other similar functions has been amended such that there may be cosmetic differences between the output observed in 1.1.1 and 3.0. This also applies to the **−text** output from the **openssl x509** and **openssl crl** applications.

Interactive mode from the **openssl** program has been removed

From now on, running it without arguments is equivalent to **openssl help**.

The error return values from some control calls (ctrl) have changed

One significant change is that controls which used to return −2 for invalid inputs, now return −1 indicating a generic error condition instead.

DH and DHX key types have different settable parameters

Previously (in 1.1.1) these conflicting parameters were allowed, but will now result in errors. See **EVP_PKEY−DH**(7) for further details. This affects the behaviour of **openssl−genpkey**(1) for DH parameter generation.

EVP_CIPHER_CTX_set_flags() ordering change

If using a cipher from a provider the **EVP_CIPH_FLAG_LENGTH_BITS** flag can only be set **after** the cipher has been assigned to the cipher context. See ''FLAGS'' in **EVP_EncryptInit**(3) for more information.

Validation of operation context parameters

Due to move of the implementation of cryptographic operations to the providers, validation of various operation parameters can be postponed until the actual operation is executed where previously it happened immediately when an operation parameter was set.

For example when setting an unsupported curve with **EVP_PKEY_CTX_set_ec_paramgen_curve_nid()** this function call will not fail but later keygen operations with the EVP_PKEY_CTX will fail.

Removal of function code from the error codes

The function code part of the error code is now always set to 0. For that reason the **ERR_GET_FUNC()** macro was removed. Applications must resolve the error codes only using the library number and the reason code.

Installation and Compilation

Please refer to the INSTALL.md file in the top of the distribution for instructions on how to build and install OpenSSL 3.0. Please also refer to the various platform specific NOTES files for your specific platform.

Upgrading from OpenSSL 1.1.1

Upgrading to OpenSSL 3.0 from OpenSSL 1.1.1 should be relatively straight forward in most cases. The most likely area where you will encounter problems is if you have used low level APIs in your code (as discussed above). In that case you are likely to start seeing deprecation warnings when compiling your application. If this happens you have 3 options:

1. Ignore the warnings. They are just warnings. The deprecated functions are still present and you may still use them. However be aware that they may be removed from a future version of OpenSSL.
2. Suppress the warnings. Refer to your compiler documentation on how to do this.
3. Remove your usage of the low level APIs. In this case you will need to rewrite your code to use the high level APIs instead

Error code changes

As OpenSSL 3.0 provides a brand new Encoder/Decoder mechanism for working with widely used file formats, application code that checks for particular error reason codes on key loading failures might need an update.

Password-protected keys may deserve special attention. If only some errors are treated as an indicator that the user should be asked about the password again, it's worth testing these scenarios and processing the newly relevant codes.

There may be more cases to treat specially, depending on the calling application code.

Upgrading from OpenSSL 1.0.2

Upgrading to OpenSSL 3.0 from OpenSSL 1.0.2 is likely to be significantly more difficult. In addition to the issues discussed above in the section about ''Upgrading from OpenSSL 1.1.1'', the main things to be aware of are:

1. The build and installation procedure has changed significantly.

   Check the file INSTALL.md in the top of the installation for instructions on how to build and install OpenSSL for your platform. Also read the various NOTES files in the same directory, as applicable for your platform.

2. Many structures have been made opaque in OpenSSL 3.0.

   The structure definitions have been removed from the public header files and moved to internal header files. In practice this means that you can no longer stack allocate some structures. Instead they must be heap allocated through some function call (typically those function names have a `_new` suffix to them). Additionally you must use ''setter'' or ''getter'' functions to access the fields within those structures.

   For example code that previously looked like this:

   ```
   EVP_MD_CTX md_ctx;

   /* This line will now generate compiler errors */
   EVP_MD_CTX_init(&md_ctx);
   ```

   The code needs to be amended to look like this:

   ```
   EVP_MD_CTX *md_ctx;

   md_ctx = EVP_MD_CTX_new();
   ...
   ...
   EVP_MD_CTX_free(md_ctx);
   ```

3. Support for TLSv1.3 has been added.

   This has a number of implications for SSL/TLS applications. See the TLS1.3 page <https://wiki.openssl.org/index.php/TLS1.3> for further details.

More details about the breaking changes between OpenSSL versions 1.0.2 and 1.1.0 can be found on the OpenSSL 1.1.0 Changes page <https://wiki.openssl.org/index.php/OpenSSL_1.1.0_Changes>.

Upgrading from the OpenSSL 2.0 FIPS Object Module

The OpenSSL 2.0 FIPS Object Module was a separate download that had to be built separately and then

integrated into your main OpenSSL 1.0.2 build. In OpenSSL 3.0 the FIPS support is fully integrated into the mainline version of OpenSSL and is no longer a separate download. For further information see ''Completing the installation of the FIPS Module''.

The function calls **FIPS_mode()** and **FIPS_mode_set()** have been removed from OpenSSL 3.0. You should rewrite your application to not use them. See **fips_module** (7) and **OSSL_PROVIDER-FIPS** (7) for details.

Completing the installation of the FIPS Module

The FIPS Module will be built and installed automatically if FIPS support has been configured. The current documentation can be found in the README-FIPS <https://github.com/openssl/openssl/blob/master/README-FIPS.md> file.

Programming

Applications written to work with OpenSSL 1.1.1 will mostly just work with OpenSSL 3.0. However changes will be required if you want to take advantage of some of the new features that OpenSSL 3.0 makes available. In order to do that you need to understand some new concepts introduced in OpenSSL 3.0. Read ''Library contexts'' in **crypto** (7) for further information.

Library Context

A library context allows different components of a complex application to each use a different library context and have different providers loaded with different configuration settings. See ''Library contexts'' in **crypto** (7) for further info.

If the user creates an **OSSL_LIB_CTX** via **OSSL_LIB_CTX_new** (3) then many functions may need to be changed to pass additional parameters to handle the library context.

Using a Library Context − Old functions that should be changed

If a library context is needed then all EVP_* digest functions that return a **const EVP_MD *** such as **EVP_sha256()** should be replaced with a call to **EVP_MD_fetch** (3). See ''ALGORITHM FETCHING'' in **crypto** (7).

If a library context is needed then all EVP_* cipher functions that return a **const EVP_CIPHER *** such as **EVP_aes_128_cbc()** should be replaced with a call to **EVP_CIPHER_fetch** (3). See ''ALGORITHM FETCHING'' in **crypto** (7).

Some functions can be passed an object that has already been set up with a library context such as **d2i_X509** (3), **d2i_X509_CRL** (3), **d2i_X509_REQ** (3) and **d2i_X509_PUBKEY** (3). If NULL is passed instead then the created object will be set up with the default library context. Use **X509_new_ex** (3), **X509_CRL_new_ex** (3), **X509_REQ_new_ex** (3) and **X509_PUBKEY_new_ex** (3) if a library context is required.

All functions listed below with a *NAME* have a replacment function *NAME_ex* that takes **OSSL_LIB_CTX** as an additional argument. Functions that have other mappings are listed along with the respective name.

- **ASN1_item_new** (3), **ASN1_item_d2i** (3), **ASN1_item_d2i_fp** (3), **ASN1_item_d2i_bio** (3), **ASN1_item_sign** (3) and **ASN1_item_verify** (3)
- **BIO_new** (3)
- **b2i_RSA_PVK_bio()** and **i2b_PVK_bio()**
- **BN_CTX_new** (3) and **BN_CTX_secure_new** (3)
- **CMS_AuthEnvelopedData_create** (3), **CMS_ContentInfo_new** (3), **CMS_data_create** (3), **CMS_digest_create** (3), **CMS_EncryptedData_encrypt** (3), **CMS_encrypt** (3), **CMS_EnvelopedData_create** (3), **CMS_ReceiptRequest_create0** (3) and **CMS_sign** (3)
- **CONF_modules_load_file** (3)
- **CTLOG_new** (3), **CTLOG_new_from_base64** (3) and **CTLOG_STORE_new** (3)
- **CT_POLICY_EVAL_CTX_new** (3)

- **d2i_AutoPrivateKey** (3), **d2i_PrivateKey** (3) and **d2i_PUBKEY** (3)
- **d2i_PrivateKey_bio** (3) and **d2i_PrivateKey_fp** (3)

  Use **d2i_PrivateKey_ex_bio** (3) and **d2i_PrivateKey_ex_fp** (3)
- **EC_GROUP_new** (3)

  Use **EC_GROUP_new_by_curve_name_ex** (3) or **EC_GROUP_new_from_params** (3).
- **EVP_DigestSignInit** (3) and **EVP_DigestVerifyInit** (3)
- **EVP_PBE_CipherInit** (3), **EVP_PBE_find** (3) and **EVP_PBE_scrypt** (3)
- **PKCS5_PBE_keyivgen** (3)
- **EVP_PKCS82PKEY** (3)
- **EVP_PKEY_CTX_new_id** (3)

  Use **EVP_PKEY_CTX_new_from_name** (3)
- **EVP_PKEY_derive_set_peer** (3), **EVP_PKEY_new_raw_private_key** (3) and **EVP_PKEY_new_raw_public_key** (3)
- **EVP_SignFinal** (3) and **EVP_VerifyFinal** (3)
- **NCONF_new** (3)
- **OCSP_RESPID_match** (3) and **OCSP_RESPID_set_by_key** (3)
- **OPENSSL_thread_stop** (3)
- **OSSL_STORE_open** (3)
- **PEM_read_bio_Parameters** (3), **PEM_read_bio_PrivateKey** (3), **PEM_read_bio_PUBKEY** (3), **PEM_read_PrivateKey** (3) and **PEM_read_PUBKEY** (3)
- **PEM_write_bio_PrivateKey** (3), **PEM_write_bio_PUBKEY** (3), **PEM_write_PrivateKey** (3) and **PEM_write_PUBKEY** (3)
- **PEM_X509_INFO_read_bio** (3) and **PEM_X509_INFO_read** (3)
- **PKCS12_add_key** (3), **PKCS12_add_safe** (3), **PKCS12_add_safes** (3), **PKCS12_create** (3), **PKCS12_decrypt_skey** (3), **PKCS12_init** (3), **PKCS12_item_decrypt_d2i** (3), **PKCS12_item_i2d_encrypt** (3), **PKCS12_key_gen_asc** (3), **PKCS12_key_gen_uni** (3), **PKCS12_key_gen_utf8** (3), **PKCS12_pack_p7encdata** (3), **PKCS12_pbe_crypt** (3), **PKCS12_PBE_keyivgen** (3), **PKCS12_SAFEBAG_create_pkcs8_encrypt** (3)
- **PKCS5_pbe_set0_algor** (3), **PKCS5_pbe_set** (3), **PKCS5_pbe2_set_iv** (3), **PKCS5_pbkdf2_set** (3) and **PKCS5_v2_scrypt_keyivgen** (3)
- **PKCS7_encrypt** (3), **PKCS7_new** (3) and **PKCS7_sign** (3)
- **PKCS8_decrypt** (3), **PKCS8_encrypt** (3) and **PKCS8_set0_pbe** (3)
- **RAND_bytes** (3) and **RAND_priv_bytes** (3)
- **SMIME_write_ASN1** (3)
- **SSL_load_client_CA_file** (3)
- **SSL_CTX_new** (3)
- **TS_RESP_CTX_new** (3)
- **X509_CRL_new** (3)
- **X509_load_cert_crl_file** (3) and **X509_load_cert_file** (3)
- **X509_LOOKUP_by_subject** (3) and **X509_LOOKUP_ctrl** (3)

- **X509_NAME_hash** (3)
- **X509_new** (3)
- **X509_REQ_new** (3) and **X509_REQ_verify** (3)
- **X509_STORE_CTX_new** (3), **X509_STORE_set_default_paths** (3), **X509_STORE_load_file** (3), **X509_STORE_load_locations** (3) and **X509_STORE_load_store** (3)

New functions that use a Library context

The following functions can be passed a library context if required. Passing NULL will use the default library context.

- **BIO_new_from_core_bio** (3)
- **EVP_ASYM_CIPHER_fetch** (3) and **EVP_ASYM_CIPHER_do_all_provided** (3)
- **EVP_CIPHER_fetch** (3) and **EVP_CIPHER_do_all_provided** (3)
- **EVP_default_properties_enable_fips** (3) and **EVP_default_properties_is_fips_enabled** (3)
- **EVP_KDF_fetch** (3) and **EVP_KDF_do_all_provided** (3)
- **EVP_KEM_fetch** (3) and **EVP_KEM_do_all_provided** (3)
- **EVP_KEYEXCH_fetch** (3) and **EVP_KEYEXCH_do_all_provided** (3)
- **EVP_KEYMGMT_fetch** (3) and **EVP_KEYMGMT_do_all_provided** (3)
- **EVP_MAC_fetch** (3) and **EVP_MAC_do_all_provided** (3)
- **EVP_MD_fetch** (3) and **EVP_MD_do_all_provided** (3)
- **EVP_PKEY_CTX_new_from_pkey** (3)
- **EVP_PKEY_Q_keygen** (3)
- **EVP_Q_mac** (3) and **EVP_Q_digest** (3)
- **EVP_RAND** (3) and **EVP_RAND_do_all_provided** (3)
- **EVP_set_default_properties** (3)
- **EVP_SIGNATURE_fetch** (3) and **EVP_SIGNATURE_do_all_provided** (3)
- **OSSL_CMP_CTX_new** (3) and **OSSL_CMP_SRV_CTX_new** (3)
- **OSSL_CRMF_ENCRYPTEDVALUE_get1_encCert** (3)
- **OSSL_CRMF_MSG_create_popo** (3) and **OSSL_CRMF_MSGS_verify_popo** (3)
- **OSSL_CRMF_pbm_new** (3) and **OSSL_CRMF_pbmp_new** (3)
- **OSSL_DECODER_CTX_add_extra** (3) and **OSSL_DECODER_CTX_new_for_pkey** (3)
- **OSSL_DECODER_fetch** (3) and **OSSL_DECODER_do_all_provided** (3)
- **OSSL_ENCODER_CTX_add_extra** (3)
- **OSSL_ENCODER_fetch** (3) and **OSSL_ENCODER_do_all_provided** (3)
- **OSSL_LIB_CTX_free** (3), **OSSL_LIB_CTX_load_config** (3) and **OSSL_LIB_CTX_set0_default** (3)
- **OSSL_PROVIDER_add_builtin** (3), **OSSL_PROVIDER_available** (3), **OSSL_PROVIDER_do_all** (3), **OSSL_PROVIDER_load** (3), **OSSL_PROVIDER_set_default_search_path** (3) and **OSSL_PROVIDER_try_load** (3)
- **OSSL_SELF_TEST_get_callback** (3) and **OSSL_SELF_TEST_set_callback** (3)
- **OSSL_STORE_attach** (3)
- **OSSL_STORE_LOADER_fetch** (3) and **OSSL_STORE_LOADER_do_all_provided** (3)

- **RAND_get0_primary** (3), **RAND_get0_private** (3), **RAND_get0_public** (3), **RAND_set_DRBG_type** (3) and **RAND_set_seed_source_type** (3)

Providers

Providers are described in detail here ‘‘Providers’’ in **crypto** (7). See also ‘‘OPENSSL PROVIDERS’’ in **crypto** (7).

Fetching algorithms and property queries

Implicit and Explicit Fetching is described in detail here ‘‘ALGORITHM FETCHING’’ in **crypto** (7).

*Mapping EVP controls and flags to provider **OSSL_PARAM** parameters*

The existing functions for controls (such as **EVP_CIPHER_CTX_ctrl** (3)) and manipulating flags (such as **EVP_MD_CTX_set_flags** (3))internally use **OSSL_PARAMS** to pass information to/from provider objects. See **OSSL_PARAM** (3) for additional information related to parameters.

For ciphers see ‘‘CONTROLS’’ in **EVP_EncryptInit** (3), ‘‘FLAGS’’ in **EVP_EncryptInit** (3) and ‘‘PARAMETERS’’ in **EVP_EncryptInit** (3).

For digests see ‘‘CONTROLS’’ in **EVP_DigestInit** (3), ‘‘FLAGS’’ in **EVP_DigestInit** (3) and ‘‘PARAMETERS’’ in **EVP_DigestInit** (3).

Deprecation of Low Level Functions

A significant number of APIs have been deprecated in OpenSSL 3.0. This section describes some common categories of deprecations. See ‘‘Deprecated function mappings’’ for the list of deprecated functions that refer to these categories.

Providers are a replacement for engines and low-level method overrides

Any accessor that uses an ENGINE is deprecated (such as **EVP_PKEY_set1_engine()**). Applications using engines should instead use providers.

Before providers were added algorithms were overriden by changing the methods used by algorithms. All these methods such as **RSA_new_method()** and **RSA_meth_new()** are now deprecated and can be replaced by using providers instead.

Deprecated i2d and d2i functions for low-level key types

Any i2d and d2i functions such as **d2i_DHparams()** that take a low-level key type have been deprecated. Applications should instead use the **OSSL_DECODER** (3) and **OSSL_ENCODER** (3) APIs to read and write files. See ‘‘Migration’’ in **d2i_RSAPrivateKey** (3) for further details.

Deprecated low-level key object getters and setters

Applications that set or get low-level key objects (such as **EVP_PKEY_set1_DH()** or **EVP_PKEY_get0()**) should instead use the OSSL_ENCODER (See **OSSL_ENCODER_to_bio** (3)) or OSSL_DECODER (See **OSSL_DECODER_from_bio** (3)) APIs, or alternatively use **EVP_PKEY_fromdata** (3) or **EVP_PKEY_todata** (3).

Deprecated low-level key parameter getters

Functions that access low-level objects directly such as **RSA_get0_n** (3) are now deprecated. Applications should use one of **EVP_PKEY_get_bn_param** (3), **EVP_PKEY_get_int_param** (3), l<**EVP_PKEY_get_size_t_param** (3)>, **EVP_PKEY_get_utf8_string_param** (3), **EVP_PKEY_get_octet_string_param** (3) or **EVP_PKEY_get_params** (3) to access fields from an EVP_PKEY. Gettable parameters are listed in ‘‘Common RSA parameters’’ in **EVP_PKEY-RSA** (7), ‘‘DH parameters’’ in **EVP_PKEY-DH** (7), ‘‘DSA parameters’’ in **EVP_PKEY-DSA** (7), ‘‘FFC parameters’’ in **EVP_PKEY-FFC** (7), ‘‘Common EC parameters’’ in **EVP_PKEY-EC** (7) and ‘‘Common X25519, X448, ED25519 and ED448 parameters’’ in **EVP_PKEY-X25519** (7). Applications may also use **EVP_PKEY_todata** (3) to return all fields.

Deprecated low-level key parameter setters

Functions that access low-level objects directly such as **RSA_set0_crt_params** (3) are now deprecated.

Applications should use **EVP_PKEY_fromdata**(3) to create new keys from user provided key data. Keys should be immutable once they are created, so if required the user may use **EVP_PKEY_todata**(3), **OSSL_PARAM_merge**(3), and **EVP_PKEY_fromdata**(3) to create a modified key. See ''Examples'' in **EVP_PKEY-DH**(7) for more information. See ''Deprecated low-level key generation functions'' for information on generating a key using parameters.

Deprecated low-level object creation

Low-level objects were created using methods such as **RSA_new**(3), **RSA_up_ref**(3) and **RSA_free**(3). Applications should instead use the high-level EVP_PKEY APIs, e.g. **EVP_PKEY_new**(3), **EVP_PKEY_up_ref**(3) and **EVP_PKEY_free**(3). See also**EVP_PKEY_CTX_new_fr om_name**(3) and **EVP_PKEY_CTX_new_from_pkey**(3).

EVP_PKEYs may be created in a variety of ways: See also ''Deprecated low-level key generation functions'', ''Deprecated low-level key reading and writing functions'' and ''Deprecated low-level key parameter setters''.

Deprecated low-level encryption functions

Low-level encryption functions such as **AES_encrypt**(3) and **AES_decrypt**(3) have been informally discouraged from use for a long time. Applications should instead use the high level EVP APIs **EVP_EncryptInit_ex**(3), **EVP_EncryptUpdate**(3), and **EVP_EncryptFinal_ex**(3) or **EVP_DecryptInit_ex**(3), **EVP_DecryptUpdate**(3) and **EVP_DecryptFinal_ex**(3).

Deprecated low-level digest functions

Use of low-level digest functions such as **SHA1_Init**(3) have been informally discouraged from use for a long time. Applications should instead use the the high level EVP APIs **EVP_DigestInit_ex**(3), **EVP_DigestUpdate**(3) and **EVP_DigestFinal_ex**(3), or the quick one-shot **EVP_Q_digest**(3).

Note that the functions **SHA1**(3), **SHA224**(3), **SHA256**(3), **SHA384**(3) and **SHA512**(3) have changed to macros that use **EVP_Q_digest**(3).

Deprecated low-level signing functions

Use of low-level signing functions such as **DSA_sign**(3) have been informally discouraged for a long time. Instead applications should use **EVP_DigestSign**(3) and **EVP_DigestVerify**(3). See also **EVP_SIGNATURE-RSA**(7), **EVP_SIGNATURE-DSA**(7), **EVP_SIGNATURE-ECDSA**(7) and **EVP_SIGNATURE-ED25519**(7).

Deprecated low-level MAC functions

Low-level mac functions such as **CMAC_Init**(3) are deprecated. Applications should instead use the new **EVP_MAC**(3) interface, using **EVP_MAC_CTX_new**(3), **EVP_MAC_CTX_free**(3), **EVP_MAC_init**(3), **EVP_MAC_update**(3) and **EVP_MAC_final**(3) or the single-shot MAC function **EVP_Q_mac**(3). See **EVP_MAC**(3), **EVP_MAC-HMAC**(7), **EVP_MAC-CMAC**(7), **EVP_MAC-GMAC**(7), **EVP_MAC-KMAC**(7), **EVP_MAC-BLAKE2**(7), **EVP_MAC-Poly1305**(7) and **EVP_MAC-Siphash**(7) for additional information.

Note that the one-shot method **HMAC()** is still available for compatability purposes.

Deprecated low-level validation functions

Low-level validation functions such as **DH_check**(3) have been informally discouraged from use for a long time. Applications should instead use the high-level EVP_PKEY APIs such as **EVP_PKEY_check**(3), **EVP_PKEY_param_check**(3), **EVP_PKEY_param_check_quick**(3), **EVP_PKEY_public_check**(3), **EVP_PKEY_public_check_quick**(3), **EVP_PKEY_private_check**(3), and **EVP_PKEY_pairwise_check**(3).

Deprecated low-level key exchange functions

Many low-level functions have been informally discouraged from use for a long time. Applications should instead use **EVP_PKEY_derive**(3). See **EVP_KEYEXCH-DH**(7), **EVP_KEYEXCH-ECDH**(7) and **EVP_KEYEXCH-X25519**(7).

Deprecated low-level key generation functions

Many low-level functions have been informally discouraged from use for a long time. Applications should instead use **EVP_PKEY_keygen_init** (3) and **EVP_PKEY_generate** (3) as described in **EVP_PKEY-DSA** (7), **EVP_PKEY-DH** (7), **EVP_PKEY-RSA** (7), **EVP_PKEY-EC** (7) and **EVP_PKEY-X25519** (7). The 'quick' one-shot function**EVP_PKEY_Q_k eygen** (3) and macros for the most common cases: <**EVP_RSA_gen** (3)> and **EVP_EC_gen** (3) may also be used.

Deprecated low-level key reading and writing functions

Use of low-level objects (such as DSA) has been informally discouraged from use for a long time. Functions to read and write these low-level objects (such as **PEM_read_DSA_PUBKEY()**) should be replaced. Applications should instead use**OSSL_ENCODER_to_bio** (3) and **OSSL_DECODER_from_bio** (3).

Deprecated low-level key printing functions

Use of low-level objects (such as DSA) has been informally discouraged from use for a long time. Functions to print these low-level objects such as **DSA_print()** should be replaced with the equivalent EVP_PKEY functions. Application should use one of **EVP_PKEY_print_public** (3), **EVP_PKEY_print_private** (3), **EVP_PKEY_print_params** (3), **EVP_PKEY_print_public_fp** (3), **EVP_PKEY_print_private_fp** (3) or **EVP_PKEY_print_params_fp** (3). Note that internally these use **OSSL_ENCODER_to_bio** (3) and **OSSL_DECODER_from_bio** (3).

Deprecated function mappings

The following functions have been deprecated in 3.0.

- **AES_bi_ige_encrypt()** and **AES_ige_encrypt()**

  There is no replacement for the IGE functions. New code should not use these modes. These undocumented functions were never integrated into the EVP layer. They implemented the AES Infinite Garble Extension (IGE) mode and AES Bi-directional IGE mode. These modes were never formally standardised and usage of these functions is believed to be very small. In particular **AES_bi_ige_encrypt()** has a known bug. It accepts 2 AES keys, but only one is ever used. The security implications are believed to be minimal, but this issue was never fixed for backwards compatibility reasons.

- **AES_encrypt()**, **AES_decrypt()**, **AES_set_encrypt_key()**, **AES_set_decrypt_key()**, **AES_cbc_encrypt()**, **AES_cfb128_encrypt()**, **AES_cfb1_encrypt()**, **AES_cfb8_encrypt()**, **AES_ecb_encrypt()**, **AES_ofb128_encrypt()**

- **AES_unwrap_key()**, **AES_wrap_key()**

  See ''Deprecated low-level encryption functions''

- **AES_options()**

  There is no replacement. It returned a string indicating if the AES code was unrolled.

- **ASN1_digest()**, **ASN1_sign()**, **ASN1_verify()**

  There are no replacements. These old functions are not used, and could be disabled with the macro NO_ASN1_OLD since OpenSSL 0.9.7.

- **ASN1_STRING_length_set()**

  Use **ASN1_STRING_set** (3) or **ASN1_STRING_set0** (3) instead. This was a potentially unsafe function that could change the bounds of a previously passed in pointer.

- **BF_encrypt()**, **BF_decrypt()**, **BF_set_key()**, **BF_cbc_encrypt()**, **BF_cfb64_encrypt()**, **BF_ecb_encrypt()**, **BF_ofb64_encrypt()**

  See ''Deprecated low-level encryption functions''. The Blowfish algorithm has been moved to the Legacy Provider.

- **BF_options()**

  There is no replacement. This option returned a constant string.

- **BIO_get_callback()**, **BIO_set_callback()**, **BIO_debug_callback()**

  Use the respective non-deprecated **_ex()** functions.

- **BN_is_prime_ex()**, **BN_is_prime_fasttest_ex()**

  Use **BN_check_prime** (3) which that avoids possible misuse and always uses at least 64 rounds of the Miller-Rabin primality test.

- **BN_pseudo_rand()**, **BN_pseudo_rand_range()**

  Use **BN_rand** (3) and **BN_rand_range** (3).

- **BN_X931_derive_prime_ex()**, **BN_X931_generate_prime_ex()**, **BN_X931_generate_Xpq()**

  There are no replacements for these low-level functions. They were used internally by **RSA_X931_derive_ex()** and **RSA_X931_generate_key_ex()** which are also deprecated. Use **EVP_PKEY_keygen** (3) instead.

- **Camellia_encrypt()**, **Camellia_decrypt()**, **Camellia_set_key()**, **Camellia_cbc_encrypt()**, **Camellia_cfb128_encrypt()**, **Camellia_cfb1_encrypt()**, **Camellia_cfb8_encrypt()**, **Camellia_ctr128_encrypt()**, **Camellia_ecb_encrypt()**, **Camellia_ofb128_encrypt()**

  See ‘‘Deprecated low-level encryption functions’’.

- **CAST_encrypt()**, **CAST_decrypt()**, **CAST_set_key()**, **CAST_cbc_encrypt()**, **CAST_cfb64_encrypt()**, **CAST_ecb_encrypt()**, **CAST_ofb64_encrypt()**

  See ‘‘Deprecated low-level encryption functions’’. The CAST algorithm has been moved to the Legacy Provider.

- **CMAC_CTX_new()**, **CMAC_CTX_cleanup()**, **CMAC_CTX_copy()**, **CMAC_CTX_free()**, **CMAC_CTX_get0_cipher_ctx()**

  See ‘‘Deprecated low-level MAC functions’’.

- **CMAC_Init()**, **CMAC_Update()**, **CMAC_Final()**, **CMAC_resume()**

  See ‘‘Deprecated low-level MAC functions’’.

- **CRYPTO_mem_ctrl()**, **CRYPTO_mem_debug_free()**, **CRYPTO_mem_debug_malloc()**, **CRYPTO_mem_debug_pop()**, **CRYPTO_mem_debug_push()**, **CRYPTO_mem_debug_realloc()**, **CRYPTO_mem_leaks()**, **CRYPTO_mem_leaks_cb()**, **CRYPTO_mem_leaks_fp()**, **CRYPTO_set_mem_debug()**

  Memory-leak checking has been deprecated in favor of more modern development tools, such as compiler memory and leak sanitizers or Valgrind.

- **CRYPTO_cts128_encrypt_block()**, **CRYPTO_cts128_encrypt()**, **CRYPTO_cts128_decrypt_block()**, **CRYPTO_cts128_decrypt()**, **CRYPTO_nistcts128_encrypt_block()**, **CRYPTO_nistcts128_encrypt()**, **CRYPTO_nistcts128_decrypt_block()**, **CRYPTO_nistcts128_decrypt()**

  Use the higher level functions **EVP_CipherInit_ex2()**, **EVP_CipherUpdate()** and **EVP_CipherFinal_ex()** instead. See the ‘‘cts_mode’’ parameter in ‘‘Gettable and Settable EVP_CIPHER_CTX parameters’’ in **EVP_EncryptInit** (3). See ‘‘EXAMPLES’’ in **EVP_EncryptInit** (3) for a AES-256-CBC-CTS example.

- **d2i_DHparams()**, **d2i_DHxparams()**, **d2i_DSAparams()**, **d2i_DSAPrivateKey()**, **d2i_DSAPrivateKey_bio()**, **d2i_DSAPrivateKey_fp()**, **d2i_DSA_PUBKEY()**, **d2i_DSA_PUBKEY_bio()**, **d2i_DSA_PUBKEY_fp()**, **d2i_DSAPublicKey()**, **d2i_ECParameters()**, **d2i_ECPrivateKey()**, **d2i_ECPrivateKey_bio()**, **d2i_ECPrivateKey_fp()**, **d2i_EC_PUBKEY()**, **d2i_EC_PUBKEY_bio()**, **d2i_EC_PUBKEY_fp()**, **o2i_ECPublicKey()**, **d2i_RSAPrivateKey()**,

d2i_RSAPrivateKey_bio(), **d2i_RSAPrivateKey_fp()**, **d2i_RSA_PUBKEY()**, **d2i_RSA_PUBKEY_bio()**, **d2i_RSA_PUBKEY_fp()**, **d2i_RSAPublicKey()**, **d2i_RSAPublicKey_bio()**, **d2i_RSAPublicKey_fp()**

See ''Deprecated i2d and d2i functions for low-level key types''

- **DES_crypt()**, **DES_fcrypt()**, **DES_encrypt1()**, **DES_encrypt2()**, **DES_encrypt3()**, **DES_decrypt3()**, **DES_ede3_cbc_encrypt()**, **DES_ede3_cfb64_encrypt()**, **DES_ede3_cfb_encrypt()**,**DES_ede3_ofb64_encrypt()**, **DES_ecb_encrypt()**, **DES_ecb3_encrypt()**, **DES_ofb64_encrypt()**, **DES_ofb_encrypt()**, DES_cfb64_encrypt **DES_cfb_encrypt()**, **DES_cbc_encrypt()**, **DES_ncbc_encrypt()**, **DES_pcbc_encrypt()**, **DES_xcbc_encrypt()**, **DES_cbc_cksum()**, **DES_quad_cksum()**, **DES_check_key_parity()**, **DES_is_weak_key()**, **DES_key_sched()**, **DES_options()**, **DES_random_key()**, **DES_set_key()**, **DES_set_key_checked()**, **DES_set_key_unchecked()**, **DES_set_odd_parity()**, **DES_string_to_2keys()**, **DES_string_to_key()**

  See ''Deprecated low-level encryption functions''. Algorithms for ''DESX-CBC'', ''DES-ECB'', ''DES-CBC'', ''DES-OFB'', ''DES-CFB'', ''DES-CFB1'' and ''DES-CFB8'' have been moved to the Legacy Provider.

- **DH_bits()**, **DH_security_bits()**, **DH_size()**

  Use **EVP_PKEY_get_bits** (3), **EVP_PKEY_get_security_bits** (3) and **EVP_PKEY_get_size** (3).

- **DH_check()**, **DH_check_ex()**, **DH_check_params()**, **DH_check_params_ex()**, **DH_check_pub_key()**, **DH_check_pub_key_ex()**

  See ''Deprecated low-level validation functions''

- **DH_clear_flags()**, **DH_test_flags()**, **DH_set_flags()**

  The **DH_FLAG_CACHE_MONT_P** flag has been deprecated without replacement. The **DH_FLAG_TYPE_DH** and **DH_FLAG_TYPE_DHX** have been deprecated. Use **EVP_PKEY_is_a()** to determine the type of a key. There is no replacement for setting these flags.

- **DH_compute_key() DH_compute_key_padded()**

  See ''Deprecated low-level key exchange functions''.

- **DH_new()**, **DH_new_by_nid()**, **DH_free()**, **DH_up_ref()**

  See ''Deprecated low-level object creation''

- **DH_generate_key()**, **DH_generate_parameters_ex()**

  See ''Deprecated low-level key generation functions''.

- **DH_get0_pqg()**, **DH_get0_p()**, **DH_get0_q()**, **DH_get0_g()**, **DH_get0_key()**, **DH_get0_priv_key()**, **DH_get0_pub_key()**, **DH_get_length()**, **DH_get_nid()**

  See ''Deprecated low-level key parameter getters''

- **DH_get_1024_160()**, **DH_get_2048_224()**, **DH_get_2048_256()**

  Applications should instead set the **OSSL_PKEY_PARAM_GROUP_NAME** as specified in ''DH parameters'' in **EVP_PKEY-DH** (7)) to one of ''dh_1024_160'', ''dh_2048_224'' or ''dh_2048_256'' when generating a DH key.

- **DH_KDF_X9_42()**

  Applications should use **EVP_PKEY_CTX_set_dh_kdf_type** (3) instead.

- **DH_get_default_method()**, **DH_get0_engine()**, DH_meth_*(), **DH_new_method()**, **DH_OpenSSL()**, **DH_get_ex_data()**, **DH_set_default_method()**, **DH_set_method()**, **DH_set_ex_data()**

  See ''Providers are a replacement for engines and low-level method overrides''

- **DHparams_print()**, **DHparams_print_fp()**

  See ‘‘Deprecated low-level key printing functions’’

- **DH_set0_key()**, **DH_set0_pqg()**, **DH_set_length()**

  See ‘‘Deprecated low-level key parameter setters’’

- **DSA_bits()**, **DSA_security_bits()**, **DSA_size()**

  Use **EVP_PKEY_get_bits** (3), **EVP_PKEY_get_security_bits** (3) and **EVP_PKEY_get_size** (3).

- **DHparams_dup()**, **DSA_dup_DH()**

  There is no direct replacement. Applications may use **EVP_PKEY_copy_parameters** (3) and **EVP_PKEY_dup** (3) instead.

- **DSA_generate_key()**, **DSA_generate_parameters_ex()**

  See ‘‘Deprecated low-level key generation functions’’.

- **DSA_get0_engine()**, **DSA_get_default_method()**, **DSA_get_ex_data()**, **DSA_get_method()**, DSA_meth_*(), **DSA_new_method()**, **DSA_OpenSSL()**, **DSA_set_default_method()**, **DSA_set_ex_data()**, **DSA_set_method()**

  See ‘‘Providers are a replacement for engines and low-level method overrides’’.

- **DSA_get0_p()**, **DSA_get0_q()**, **DSA_get0_g()**, **DSA_get0_pqg()**, **DSA_get0_key()**, **DSA_get0_priv_key()**, **DSA_get0_pub_key()**

  See ‘‘Deprecated low-level key parameter getters’’.

- **DSA_new()**, **DSA_free()**, **DSA_up_ref()**

  See ‘‘Deprecated low-level object creation’’

- **DSAparams_dup()**

  There is no direct replacement. Applications may use **EVP_PKEY_copy_parameters** (3) and **EVP_PKEY_dup** (3) instead.

- **DSAparams_print()**, **DSAparams_print_fp()**, **DSA_print()**, **DSA_print_fp()**

  See ‘‘Deprecated low-level key printing functions’’

- **DSA_set0_key()**, **DSA_set0_pqg()**

  See ‘‘Deprecated low-level key parameter setters’’

- **DSA_set_flags()**, **DSA_clear_flags()**, **DSA_test_flags()**

  The **DSA_FLAG_CACHE_MONT_P** flag has been deprecated without replacement.

- **DSA_sign()**, **DSA_do_sign()**, **DSA_sign_setup()**, **DSA_verify()**, **DSA_do_verify()**

  See ‘‘Deprecated low-level signing functions’’.

- **ECDH_compute_key()**

  See ‘‘Deprecated low-level key exchange functions’’.

- **ECDH_KDF_X9_62()**

  Applications may either set this using the helper function **EVP_PKEY_CTX_set_ecdh_kdf_type** (3) or by setting an **OSSL_PARAM** using the ‘‘kdf-type’’ as shown in ‘‘EXAMPLES’’ in **EVP_KEYEXCH-ECDH** (7)

- **ECDSA_sign()**, **ECDSA_sign_ex()**, **ECDSA_sign_setup()**, **ECDSA_do_sign()**, **ECDSA_do_sign_ex()**, **ECDSA_verify()**, **ECDSA_do_verify()**

  See ‘‘Deprecated low-level signing functions’’.

- **ECDSA_size()**

  Applications should use **EVP_PKEY_get_size** (3).

- **EC_GF2m_simple_method()**, **EC_GFp_mont_method()**, **EC_GFp_nist_method()**, **EC_GFp_nistp224_method()**, **EC_GFp_nistp256_method()**, **EC_GFp_nistp521_method()**, **EC_GFp_simple_method()**

  There are no replacements for these functions. Applications should rely on the library automatically assigning a suitable method internally when an EC_GROUP is constructed.

- **EC_GROUP_clear_free()**

  Use **EC_GROUP_free** (3) instead.

- **EC_GROUP_get_curve_GF2m()**, **EC_GROUP_get_curve_GFp()**, **EC_GROUP_set_curve_GF2m()**, **EC_GROUP_set_curve_GFp()**

  Applications should use **EC_GROUP_get_curve** (3) and **EC_GROUP_set_curve** (3).

- **EC_GROUP_have_precompute_mult()**, **EC_GROUP_precompute_mult()**, **EC_KEY_precompute_mult()**

  These functions are not widely used. Applications should instead switch to named curves which OpenSSL has hardcoded lookup tables for.

- **EC_GROUP_new()**, **EC_GROUP_method_of()**, **EC_POINT_method_of()**

  EC_METHOD is now an internal-only concept and a suitable EC_METHOD is assigned internally without application intervention. Users of**EC_GR OUP_new()** should switch to a different suitable constructor.

- **EC_KEY_can_sign()**

  Applications should use **EVP_PKEY_can_sign** (3) instead.

- **EC_KEY_check_key()**

  See ‘‘Deprecated low-level validation functions’’

- **EC_KEY_set_flags()**, **EC_KEY_get_flags()**, **EC_KEY_clear_flags()**

  See ‘‘Common EC parameters’’ in **EVP_PKEY−EC** (7) which handles flags as seperate parameters for **OSSL_PKEY_PARAM_EC_POINT_CONVERSION_FORMAT**, **OSSL_PKEY_PARAM_EC_GROUP_CHECK_TYPE**, **OSSL_PKEY_PARAM_EC_ENCODING**, **OSSL_PKEY_PARAM_USE_COFACTOR_ECDH** and **OSSL_PKEY_PARAM_EC_INCLUDE_PUBLIC**. See also ‘‘EXAMPLES’’ in **EVP_PKEY−EC** (7)

- **EC_KEY_dup()**, **EC_KEY_copy()**

  There is no direct replacement. Applications may use **EVP_PKEY_copy_parameters** (3) and **EVP_PKEY_dup** (3) instead.

- **EC_KEY_decoded_from_explicit_params()**

  There is no replacement.

- **EC_KEY_generate_key()**

  See ‘‘Deprecated low-level key generation functions’’.

- **EC_KEY_get0_group()**, **EC_KEY_get0_private_key()**, **EC_KEY_get0_public_key()**, **EC_KEY_get_conv_form()**, **EC_KEY_get_enc_flags()**

  See ‘‘Deprecated low-level key parameter getters’’.

- **EC_KEY_get0_engine()**, **EC_KEY_get_default_method()**, **EC_KEY_get_method()**, **EC_KEY_new_method()**, **EC_KEY_get_ex_data()**, **EC_KEY_OpenSSL()**, **EC_KEY_set_ex_data()**, **EC_KEY_set_default_method()**, EC_KEY_METHOD_*(),

EC_KEY_set_method()

See ‘‘Providers are a replacement for engines and low-level method overrides’’

- **EC_METHOD_get_field_type()**

  Use **EC_GROUP_get_field_type** (3) instead. See ‘‘Providers are a replacement for engines and low-level method overrides’’

- **EC_KEY_key2buf()**, **EC_KEY_oct2key()**, **EC_KEY_oct2priv()**, **EC_KEY_priv2buf()**, **EC_KEY_priv2oct()**

  There are no replacements for these.

- **EC_KEY_new()**, **EC_KEY_new_by_curve_name()**, **EC_KEY_free()**, **EC_KEY_up_ref()**

  See ‘‘Deprecated low-level object creation’’

- **EC_KEY_print()**, **EC_KEY_print_fp()**

  See ‘‘Deprecated low-level key printing functions’’

- **EC_KEY_set_asn1_flag()**, **EC_KEY_set_conv_form()**, **EC_KEY_set_enc_flags()**

  See ‘‘Deprecated low-level key parameter setters’’.

- **EC_KEY_set_group()**, **EC_KEY_set_private_key()**, **EC_KEY_set_public_key()**, **EC_KEY_set_public_key_affine_coordinates()**

  See ‘‘Deprecated low-level key parameter setters’’.

- **ECParameters_print()**, **ECParameters_print_fp()**, **ECPKParameters_print()**, **ECPKParameters_print_fp()**

  See ‘‘Deprecated low-level key printing functions’’

- **EC_POINT_bn2point()**, **EC_POINT_point2bn()**

  These functions were not particularly useful, since EC point serialization formats are not individual big-endian integers.

- **EC_POINT_get_affine_coordinates_GF2m()**, **EC_POINT_get_affine_coordinates_GFp()**, **EC_POINT_set_affine_coordinates_GF2m()**, **EC_POINT_set_affine_coordinates_GFp()**

  Applications should use **EC_POINT_get_affine_coordinates** (3) and **EC_POINT_set_affine_coordinates** (3) instead.

- **EC_POINT_get_Jprojective_coordinates_GFp()**, **EC_POINT_set_Jprojective_coordinates_GFp()**

  These functions are not widely used. Applications should instead use the **EC_POINT_set_affine_coordinates** (3) and **EC_POINT_get_affine_coordinates** (3) functions.

- **EC_POINT_make_affine()**, **EC_POINTs_make_affine()**

  There is no replacement. These functions were not widely used, and OpenSSL automatically performs this conversion when needed.

- **EC_POINT_set_compressed_coordinates_GF2m()**, **EC_POINT_set_compressed_coordinates_GFp()**

  Applications should use **EC_POINT_set_compressed_coordinates** (3) instead.

- **EC_POINTs_mul()**

  This function is not widely used. Applications should instead use the **EC_POINT_mul** (3) function.

- **ENGINE_*()**

  All engine functions are deprecated. An engine should be rewritten as a provider. See ‘‘Providers are a

replacement for engines and low-level method overrides''.

- **ERR_load_*()**, **ERR_func_error_string()**, **ERR_get_error_line()**, **ERR_get_error_line_data()**, **ERR_get_state()**

  OpenSSL now loads error strings automatically so these functions are not needed.

- **ERR_peek_error_line_data()**, **ERR_peek_last_error_line_data()**

  The new functions are **ERR_peek_error_func**(3), **ERR_peek_last_error_func**(3), **ERR_peek_error_data**(3), **ERR_peek_last_error_data**(3), **ERR_get_error_all**(3), **ERR_peek_error_all**(3) and **ERR_peek_last_error_all**(3). Applications should use **ERR_get_error_all**(3), or pick information with ERR_peek functions and finish off with getting the error code by using **ERR_get_error**(3).

- **EVP_CIPHER_CTX_iv()**, **EVP_CIPHER_CTX_iv_noconst()**, **EVP_CIPHER_CTX_original_iv()**

  Applications should instead use **EVP_CIPHER_CTX_get_updated_iv**(3), **EVP_CIPHER_CTX_get_updated_iv**(3) and **EVP_CIPHER_CTX_get_original_iv**(3) respectively. See**EVP_CIPHER_CTX_get_original_i v**(3) for further information.

- **EVP_CIPHER_meth_*()**, **EVP_MD_CTX_set_update_fn()**, **EVP_MD_CTX_update_fn()**, **EVP_MD_meth_*()**

  See ''Providers are a replacement for engines and low-level method overrides''.

- **EVP_PKEY_CTRL_PKCS7_ENCRYPT()**, **EVP_PKEY_CTRL_PKCS7_DECRYPT()**, **EVP_PKEY_CTRL_PKCS7_SIGN()**, **EVP_PKEY_CTRL_CMS_ENCRYPT()**, **EVP_PKEY_CTRL_CMS_DECRYPT()**, and **EVP_PKEY_CTRL_CMS_SIGN()**

  These control operations are not invoked by the OpenSSL library anymore and are replaced by direct checks of the key operation against the key type when the operation is initialized.

- **EVP_PKEY_CTX_get0_dh_kdf_ukm()**, **EVP_PKEY_CTX_get0_ecdh_kdf_ukm()**

  See the ''kdf-ukm'' item in ''DH key exchange parameters'' in **EVP_KEYEXCH-DH**(7) and ''ECDH Key Exchange parameters'' in **EVP_KEYEXCH-ECDH**(7). These functions are obsolete and should not be required.

- **EVP_PKEY_CTX_set_rsa_keygen_pubexp()**

  Applications should use **EVP_PKEY_CTX_set1_rsa_keygen_pubexp**(3) instead.

- **EVP_PKEY_cmp()**, **EVP_PKEY_cmp_parameters()**

  Applications should use **EVP_PKEY_eq**(3) and **EVP_PKEY_parameters_eq**(3) instead. See **EVP_PKEY_copy_parameters**(3) for further details.

- **EVP_PKEY_encrypt_old()**, **EVP_PKEY_decrypt_old()**,

  Applications should use **EVP_PKEY_encrypt_init**(3) and **EVP_PKEY_encrypt**(3) or **EVP_PKEY_decrypt_init**(3) and **EVP_PKEY_decrypt**(3) instead.

- **EVP_PKEY_get0()**

  This function returns NULL if the key comes from a provider.

- **EVP_PKEY_get0_DH()**, **EVP_PKEY_get0_DSA()**, **EVP_PKEY_get0_EC_KEY()**, **EVP_PKEY_get0_RSA()**, **EVP_PKEY_get1_DH()**, **EVP_PKEY_get1_DSA()**, EVP_PKEY_get1_EC_KEY and **EVP_PKEY_get1_RSA()**, **EVP_PKEY_get0_hmac()**, **EVP_PKEY_get0_poly1305()**, **EVP_PKEY_get0_siphash()**

  See ''Functions that return an internal key should be treated as read only''.

- **EVP_PKEY_meth_*()**

  See ''Providers are a replacement for engines and low-level method overrides''.

- **EVP_PKEY_new_CMAC_key()**

  See ‘‘Deprecated low-level MAC functions’’.

- **EVP_PKEY_assign()**, **EVP_PKEY_set1_DH()**, **EVP_PKEY_set1_DSA()**, **EVP_PKEY_set1_EC_KEY()**, **EVP_PKEY_set1_RSA()**

  See ‘‘Deprecated low-level key object getters and setters’’

- **EVP_PKEY_set1_tls_encodedpoint() EVP_PKEY_get1_tls_encodedpoint()**

  These functions were previously used by libssl to set or get an encoded public key into/from an EVP_PKEY object. With OpenSSL 3.0 these are replaced by the more generic functions **EVP_PKEY_set1_encoded_public_key**(3) and **EVP_PKEY_get1_encoded_public_key**(3). The old versions have been converted to deprecated macros that just call the new functions.

- **EVP_PKEY_set1_engine()**, **EVP_PKEY_get0_engine()**

  See ‘‘Providers are a replacement for engines and low-level method overrides’’.

- **EVP_PKEY_set_alias_type()**

  This function has been removed. There is no replacement. See ‘‘**EVP_PKEY_set_alias_type()** method has been removed’’

- **HMAC_Init_ex()**, **HMAC_Update()**, **HMAC_Final()**, **HMAC_size()**

  See ‘‘Deprecated low-level MAC functions’’.

- **HMAC_CTX_new()**, **HMAC_CTX_free()**, **HMAC_CTX_copy()**, **HMAC_CTX_reset()**, **HMAC_CTX_set_flags()**, **HMAC_CTX_get_md()**

  See ‘‘Deprecated low-level MAC functions’’.

- **i2d_DHparams()**, **i2d_DHxparams()**

  See ‘‘Deprecated low-level key reading and writing functions’’ and ‘‘Migration’’ in **d2i_RSAPrivateKey**(3)

- **i2d_DSAparams()**, **i2d_DSAPrivateKey()**, **i2d_DSAPrivateKey_bio()**, **i2d_DSAPrivateKey_fp()**, **i2d_DSA_PUBKEY()**, **i2d_DSA_PUBKEY_bio()**, **i2d_DSA_PUBKEY_fp()**, **i2d_DSAPublicKey()**

  See ‘‘Deprecated low-level key reading and writing functions’’ and ‘‘Migration’’ in **d2i_RSAPrivateKey**(3)

- **i2d_ECParameters()**, **i2d_ECPrivateKey()**, **i2d_ECPrivateKey_bio()**, **i2d_ECPrivateKey_fp()**, **i2d_EC_PUBKEY()**, **i2d_EC_PUBKEY_bio()**, **i2d_EC_PUBKEY_fp()**, **i2o_ECPublicKey()**

  See ‘‘Deprecated low-level key reading and writing functions’’ and ‘‘Migration’’ in **d2i_RSAPrivateKey**(3)

- **i2d_RSAPrivateKey()**, **i2d_RSAPrivateKey_bio()**, **i2d_RSAPrivateKey_fp()**, **i2d_RSA_PUBKEY()**, **i2d_RSA_PUBKEY_bio()**, **i2d_RSA_PUBKEY_fp()**, **i2d_RSAPublicKey()**, **i2d_RSAPublicKey_bio()**, **i2d_RSAPublicKey_fp()**

  See ‘‘Deprecated low-level key reading and writing functions’’ and ‘‘Migration’’ in **d2i_RSAPrivateKey**(3)

- **IDEA_encrypt()**, **IDEA_set_decrypt_key()**, **IDEA_set_encrypt_key()**, **IDEA_cbc_encrypt()**, **IDEA_cfb64_encrypt()**, **IDEA_ecb_encrypt()**, **IDEA_ofb64_encrypt()**

  See ‘‘Deprecated low-level encryption functions’’. IDEA has been moved to the Legacy Provider.

- **IDEA_options()**

  There is no replacement. This function returned a constant string.

- **MD2()**, **MD2_Init()**, **MD2_Update()**, **MD2_Final()**

See ‘‘Deprecated low-level encryption functions’’. MD2 has been moved to the Legacy Provider.

- **MD2_options()**

  There is no replacement. This function returned a constant string.

- **MD4()**, **MD4_Init()**, **MD4_Update()**, **MD4_Final()**, **MD4_Transform()**

  See ‘‘Deprecated low-level encryption functions’’. MD4 has been moved to the Legacy Provider.

- **MDC2()**, **MDC2_Init()**, **MDC2_Update()**, **MDC2_Final()**

  See ‘‘Deprecated low-level encryption functions’’. MDC2 has been moved to the Legacy Provider.

- **MD5()**, **MD5_Init()**, **MD5_Update()**, **MD5_Final()**, **MD5_Transform()**

  See ‘‘Deprecated low-level encryption functions’’.

- **NCONF_WIN32()**

  This undocumented function has no replacement. See ‘‘HISTORY’’ in **config** (5) for more details.

- **OCSP_parse_url()**

  Use **OSSL_HTTP_parse_url** (3) instead.

- **OCSP_REQ_CTX** type and **OCSP_REQ_CTX_*()** functions

  These methods were used to collect all necessary data to form a HTTP request, and to perform the HTTP transfer with that request. With OpenSSL 3.0, the type is **OSSL_HTTP_REQ_CTX**, and the deprecated functions are replaced with **OSSL_HTTP_REQ_CTX_*()**. See **OSSL_HTTP_REQ_CTX** (3) for additional details.

- **OPENSSL_fork_child()**, **OPENSSL_fork_parent()**, **OPENSSL_fork_prepare()**

  There is no replacement for these functions. These pthread fork support methods were unused by OpenSSL.

- **OSSL_STORE_ctrl()**, **OSSL_STORE_do_all_loaders()**, **OSSL_STORE_LOADER_get0_engine()**, **OSSL_STORE_LOADER_get0_scheme()**, **OSSL_STORE_LOADER_new()**, **OSSL_STORE_LOADER_set_attach()**, **OSSL_STORE_LOADER_set_close()**, **OSSL_STORE_LOADER_set_ctrl()**, **OSSL_STORE_LOADER_set_eof()**, **OSSL_STORE_LOADER_set_error()**, **OSSL_STORE_LOADER_set_expect()**, **OSSL_STORE_LOADER_set_find()**, **OSSL_STORE_LOADER_set_load()**, **OSSL_STORE_LOADER_set_open()**, **OSSL_STORE_LOADER_set_open_ex()**, **OSSL_STORE_register_loader()**, **OSSL_STORE_unregister_loader()**, **OSSL_STORE_vctrl()**

  These functions helped applications and engines create loaders for schemes they supported. These are all deprecated and discouraged in favour of provider implementations, see **provider−storemgmt** (7).

- **PEM_read_DHparams()**, **PEM_read_bio_DHparams()**, **PEM_read_DSAparams()**, **PEM_read_bio_DSAparams()**, **PEM_read_DSAPrivateKey()**, **PEM_read_DSA_PUBKEY()**, PEM_read_bio_DSAPrivateKey and **PEM_read_bio_DSA_PUBKEY()**, **PEM_read_ECPKParameters()**, **PEM_read_ECPrivateKey()**, **PEM_read_EC_PUBKEY()**, **PEM_read_bio_ECPKParameters()**, **PEM_read_bio_ECPrivateKey()**, **PEM_read_bio_EC_PUBKEY()**, **PEM_read_RSAPrivateKey()**, **PEM_read_RSA_PUBKEY()**, **PEM_read_RSAPublicKey()**, **PEM_read_bio_RSAPrivateKey()**, **PEM_read_bio_RSA_PUBKEY()**, **PEM_read_bio_RSAPublicKey()**, **PEM_write_bio_DHparams()**, **PEM_write_bio_DHxparams()**, **PEM_write_DHparams()**, **PEM_write_DHxparams()**, **PEM_write_DSAparams()**, **PEM_write_DSAPrivateKey()**, **PEM_write_DSA_PUBKEY()**, **PEM_write_bio_DSAparams()**, **PEM_write_bio_DSAPrivateKey()**, **PEM_write_bio_DSA_PUBKEY()**, **PEM_write_ECPKParameters()**, **PEM_write_ECPrivateKey()**, **PEM_write_EC_PUBKEY()**, **PEM_write_bio_ECPKParameters()**, **PEM_write_bio_ECPrivateKey()**,

PEM_write_bio_EC_PUBKEY(), **PEM_write_RSAPrivateKey()**, **PEM_write_RSA_PUBKEY()**, **PEM_write_RSAPublicKey()**, **PEM_write_bio_RSAPrivateKey()**, **PEM_write_bio_RSA_PUBKEY()**, **PEM_write_bio_RSAPublicKey()**,

See ''Deprecated low-level key reading and writing functions''

- **PKCS1_MGF1()**

  See ''Deprecated low-level encryption functions''.

- **RAND_get_rand_method()**, **RAND_set_rand_method()**, **RAND_OpenSSL()**, **RAND_set_rand_engine()**

  Applications should instead use **RAND_set_DRBG_type** (3), **EVP_RAND** (3) and **EVP_RAND** (7). See **RAND_set_rand_method** (3) for more details.

- **RC2_encrypt()**, **RC2_decrypt()**, **RC2_set_key()**, **RC2_cbc_encrypt()**, **RC2_cfb64_encrypt()**, **RC2_ecb_encrypt()**, **RC2_ofb64_encrypt()**, **RC4()**, **RC4_set_key()**, **RC4_options()**, **RC5_32_encrypt()**, **RC5_32_set_key()**, **RC5_32_decrypt()**, **RC5_32_cbc_encrypt()**, **RC5_32_cfb64_encrypt()**, **RC5_32_ecb_encrypt()**, **RC5_32_ofb64_encrypt()**

  See ''Deprecated low-level encryption functions''. The Algorithms ''RC2'', ''RC4'' and ''RC5'' have been moved to the Legacy Provider.

- **RIPEMD160()**, **RIPEMD160_Init()**, **RIPEMD160_Update()**, **RIPEMD160_Final()**, **RIPEMD160_Transform()**

  See ''Deprecated low-level digest functions''. The RIPE algorithm has been moved to the Legacy Provider.

- **RSA_bits()**, **RSA_security_bits()**, **RSA_size()**

  Use **EVP_PKEY_get_bits** (3), **EVP_PKEY_get_security_bits** (3) and **EVP_PKEY_get_size** (3).

- **RSA_check_key()**, **RSA_check_key_ex()**

  See ''Deprecated low-level validation functions''

- **RSA_clear_flags()**, **RSA_flags()**, **RSA_set_flags()**, **RSA_test_flags()**, **RSA_setup_blinding()**, **RSA_blinding_off()**, **RSA_blinding_on()**

  All of these RSA flags have been deprecated without replacement:

  RSA_FLAG_BLINDING, **RSA_FLAG_CACHE_PRIVATE**, **RSA_FLAG_CACHE_PUBLIC**, **RSA_FLAG_EXT_PKEY**, **RSA_FLAG_NO_BLINDING**, **RSA_FLAG_THREAD_SAFE** **RSA_METHOD_FLAG_NO_CHECK**

- **RSA_generate_key_ex()**, **RSA_generate_multi_prime_key()**

  See ''Deprecated low-level key generation functions''.

- **RSA_get0_engine()**

  See ''Providers are a replacement for engines and low-level method overrides''

- **RSA_get0_crt_params()**, **RSA_get0_d()**, **RSA_get0_dmp1()**, **RSA_get0_dmq1()**, **RSA_get0_e()**, **RSA_get0_factors()**, **RSA_get0_iqmp()**, **RSA_get0_key()**, **RSA_get0_multi_prime_crt_params()**, **RSA_get0_multi_prime_factors()**, **RSA_get0_n()**, **RSA_get0_p()**, **RSA_get0_pss_params()**, **RSA_get0_q()**, **RSA_get_multi_prime_extra_count()**

  See ''Deprecated low-level key parameter getters''

- **RSA_new()**, **RSA_free()**, **RSA_up_ref()**

  See ''Deprecated low-level object creation''.

- **RSA_get_default_method()**, RSA_get_ex_data and **RSA_get_method()**

  See ''Providers are a replacement for engines and low-level method overrides''.

- **RSA_get_version()**

  There is no replacement.

- **RSA_meth_*()**, **RSA_new_method()**, RSA_null_method and **RSA_PKCS1_OpenSSL()**

  See ‘‘Providers are a replacement for engines and low-level method overrides’’.

- **RSA_padding_add_*()**, **RSA_padding_check_*()**

  See ‘‘Deprecated low-level signing functions’’ and ‘‘Deprecated low-level encryption functions’’.

- **RSA_print()**, **RSA_print_fp()**

  See ‘‘Deprecated low-level key printing functions’’

- **RSA_public_encrypt()**, **RSA_private_decrypt()**

  See ‘‘Deprecated low-level encryption functions’’

- **RSA_private_encrypt()**, **RSA_public_decrypt()**

  This is equivalent to doing sign and verify recover operations (with a padding mode of none). See ‘‘Deprecated low-level signing functions’’.

- **RSAPrivateKey_dup()**, **RSAPublicKey_dup()**

  There is no direct replacement. Applications may use **EVP_PKEY_dup** (3).

- **RSAPublicKey_it()**, **RSAPrivateKey_it()**

  See ‘‘Deprecated low-level key reading and writing functions’’

- **RSA_set0_crt_params()**, **RSA_set0_factors()**, **RSA_set0_key()**, **RSA_set0_multi_prime_params()**

  See ‘‘Deprecated low-level key parameter setters’’.

- **RSA_set_default_method()**, **RSA_set_method()**, **RSA_set_ex_data()**

  See ‘‘Providers are a replacement for engines and low-level method overrides’’

- **RSA_sign()**, **RSA_sign_ASN1_OCTET_STRING()**, **RSA_verify()**, **RSA_verify_ASN1_OCTET_STRING()**, **RSA_verify_PKCS1_PSS()**, **RSA_verify_PKCS1_PSS_mgf1()**

  See ‘‘Deprecated low-level signing functions’’.

- **RSA_X931_derive_ex()**, **RSA_X931_generate_key_ex()**, **RSA_X931_hash_id()**

  There are no replacements for these functions. X931 padding can be set using ‘‘Signature Parameters’’ in **EVP_SIGNATURE−RSA** (7). See **OSSL_SIGNATURE_PARAM_PAD_MODE**.

- **SEED_encrypt()**, **SEED_decrypt()**, **SEED_set_key()**, **SEED_cbc_encrypt()**, **SEED_cfb128_encrypt()**, **SEED_ecb_encrypt()**, **SEED_ofb128_encrypt()**

  See ‘‘Deprecated low-level encryption functions’’. The SEED algorithm has been moved to the Legacy Provider.

- **SHA1_Init()**, **SHA1_Update()**, **SHA1_Final()**, **SHA1_Transform()**, **SHA224_Init()**, **SHA224_Update()**, **SHA224_Final()**, **SHA256_Init()**, **SHA256_Update()**, **SHA256_Final()**, **SHA256_Transform()**, **SHA384_Init()**, **SHA384_Update()**, **SHA384_Final()**, **SHA512_Init()**, **SHA512_Update()**, **SHA512_Final()**, **SHA512_Transform()**

  See ‘‘Deprecated low-level digest functions’’.

- **SRP_Calc_A()**, **SRP_Calc_B()**, **SRP_Calc_client_key()**, **SRP_Calc_server_key()**, **SRP_Calc_u()**, **SRP_Calc_x()**, **SRP_check_known_gN_param()**, **SRP_create_verifier()**, **SRP_create_verifier_BN()**, **SRP_get_default_gN()**, **SRP_user_pwd_free()**, **SRP_user_pwd_new()**, **SRP_user_pwd_set0_sv()**, **SRP_user_pwd_set1_ids()**,

SRP_user_pwd_set_gN(), **SRP_VBASE_add0_user()**, **SRP_VBASE_free()**, **SRP_VBASE_get1_by_user()**, **SRP_VBASE_init()**, **SRP_VBASE_new()**, **SRP_Verify_A_mod_N()**, **SRP_Verify_B_mod_N()**

There are no replacements for the SRP functions.

- **SSL_CTX_set_tmp_dh_callback()**, **SSL_set_tmp_dh_callback()**, **SSL_CTX_set_tmp_dh()**, **SSL_set_tmp_dh()**

  These are used to set the Diffie-Hellman (DH) parameters that are to be used by servers requiring ephemeral DH keys. Instead applications should consider using the built-in DH parameters that are available by calling **SSL_CTX_set_dh_auto** (3) or **SSL_set_dh_auto** (3). If custom parameters are necessary then applications can use the alternative functions **SSL_CTX_set0_tmp_dh_pkey** (3) and **SSL_set0_tmp_dh_pkey** (3). There is no direct replacement for the ''callback'' functions. The callback was originally useful in order to have different parameters for export and non-export ciphersuites. Export ciphersuites are no longer supported by OpenSSL. Use of the callback functions should be replaced by one of the other methods described above.

- **SSL_CTX_set_tlsext_ticket_key_cb()**

  Use the new **SSL_CTX_set_tlsext_ticket_key_evp_cb** (3) function instead.

- **WHIRLPOOL()**, **WHIRLPOOL_Init()**, **WHIRLPOOL_Update()**, **WHIRLPOOL_Final()**, **WHIRLPOOL_BitUpdate()**

  See ''Deprecated low-level digest functions''. The Whirlpool algorithm has been moved to the Legacy Provider.

- **X509_certificate_type()**

  This was an undocumented function. Applications can use **X509_get0_pubkey** (3) and **X509_get0_signature** (3) instead.

- **X509_http_nbio()**, **X509_CRL_http_nbio()**

  Use **X509_load_http** (3) and **X509_CRL_load_http** (3) instead.

Using the FIPS Module in applications

See **fips_module** (7) and **OSSL_PROVIDER-FIPS** (7) for details.

OpenSSL command line application changes

New applications

openssl kdf uses the new **EVP_KDF** (3) API. **openssl kdf**uses the ne w **EVP_MAC** (3) API.

Added options

-provider_path and **-provider** are available to all apps and can be used multiple times to load any providers, such as the 'legacy' provider or third party providers. If used then the 'default' provider would also need to be specified if required. The **-provider_path** must be specified before the **-provider** option.

The **list** app has many new options. See **openssl-list** (1) for more information.

-crl_lastupdate and **-crl_nextupdate** used by **openssl ca** allows explicit setting of fields in the generated CRL.

Removed options

Interactive mode is not longer available.

The **-crypt** option used by **openssl passwd**. The**-c** option used by **openssl x509**, **openssl dhparam**, **openssl dsaparam**, and **openssl ecparam**.

Other Changes

The output of Command line applications may have minor changes. These are primarily changes in capitalisation and white space. However, in some cases, there are additional differences. For example, the DH parameters output from **openssl dhparam** now lists 'P', 'Q', 'G' and 'pcounter' instead of 'prime',

’generator’, ’subgroup order’ and ’counter’ respectively.

The **openssl** commands that read keys, certificates, and CRLs now automatically detect the PEM or DER format of the input files so it is not necessary to explicitly specify the input format anymore. However if the input format option is used the specified format will be required.

openssl speed no longer uses low-level API calls. This implies some of the performance numbers might not be comparable with the previous releases due to higher overhead. This applies particularly to measuring performance on smaller data chunks.

b<openssl dhparam>, **openssl dsa**, **openssl gendsa**, **openssl dsaparam**, **openssl genrsa** and **openssl rsa** have been modified to use PKEY APIs. **openssl genrsa** and **openssl rsa** no w write PKCS #8 keys by default.

Default settings

‘‘SHA256’’ is now the default digest for TS query used by **openssl ts**.

Deprecated apps

openssl rsautl is deprecated, use **openssl pkeyutl** instead. **openssl dhparam**, **openssl dsa**, **openssl gendsa**, **openssl dsaparam**, **openssl genrsa**, **openssl rsa**, **openssl genrsa** and **openssl rsa** are now in maintenance mode and no new features will be added to them.

TLS Changes

- TLS 1.3 FFDHE key exchange support added

  This uses DH safe prime named groups.

- Support for fully ‘‘pluggable’’ TLSv1.3 groups.

  This means that providers may supply their own group implementations (using either the ‘‘key exchange’’ or the ‘‘key encapsulation’’ methods) which will automatically be detected and used by libssl.

- SSL and SSL_CTX options are now 64 bit instead of 32 bit.

  The signatures of the functions to get and set options on SSL and SSL_CTX objects changed from ‘‘unsigned long’’ to ‘‘uint64_t’’ type.

  This may require source code changes. For example it is no longer possible to use the **SSL_OP_** macro values in preprocessor `#if` conditions. However it is still possible to test whether these macros are defined or not.

  See **SSL_CTX_get_options** (3), **SSL_CTX_set_options** (3), **SSL_get_options** (3) and **SSL_set_options** (3).

- **SSL_set1_host()** and **SSL_add1_host()** Changes

  These functions now take IP literal addresses as well as actual hostnames.

- Added SSL option SSL_OP_CLEANSE_PLAINTEXT

  If the option is set, openssl cleanses (zeroizes) plaintext bytes from internal buffers after delivering them to the application. Note, the application is still responsible for cleansing other copies (e.g.: data received by **SSL_read** (3)).

- Client-initiated renegotiation is disabled by default.

  To allow it, use the **−client_renegotiation** option, the **SSL_OP_ALLOW_CLIENT_RENEGOTIATION** flag, or the `ClientRenegotiation` config parameter as appropriate.

- Secure renegotiation is now required by default for TLS connections

  Support for RFC 5746 secure renegotiation is now required by default for SSL or TLS connections to succeed. Applications that require the ability to connect to legacy peers will need to explicitly set SSL_OP_LEGACY_SERVER_CONNECT. Accordingly, SSL_OP_LEGACY_SERVER_CONNECT is no

longer set as part of SSL_OP_ALL.

- Combining the Configure options no-ec and no-dh no longer disables TLSv1.3

  Typically if OpenSSL has no EC or DH algorithms then it cannot support connections with TLSv1.3. However OpenSSL now supports ''pluggable'' groups through providers. Therefore third party providers may supply group implementations even where there are no built-in ones. Attempting to create TLS connections in such a build without also disabling TLSv1.3 at run time or using third party provider groups may result in handshake failures. TLSv1.3 can be disabled at compile time using the ''no-tls1_3'' Configure option.

- **SSL_CTX_set_ciphersuites()** and **SSL_set_ciphersuites()** changes.

  The methods now ignore unknown ciphers.

- Security callback change.

  The security callback, which can be customised by application code, supports the security operation SSL_SECOP_TMP_DH. This is defined to take an EVP_PKEY in the ''other'' parameter. In most places this is what is passed. All these places occur server side. However there was one client side call of this security operation and it passed a DH object instead. This is incorrect according to the definition of SSL_SECOP_TMP_DH, and is inconsistent with all of the other locations. Therefore this client side call has been changed to pass an EVP_PKEY instead.

- New SSL option SSL_OP_IGNORE_UNEXPECTED_EOF

  The SSL option SSL_OP_IGNORE_UNEXPECTED_EOF is introduced. If that option is set, an unexpected EOF is ignored, it pretends a close notify was received instead and so the returned error becomes SSL_ERROR_ZERO_RETURN.

- The security strength of SHA1 and MD5 based signatures in TLS has been reduced.

  This results in SSL 3, TLS 1.0, TLS 1.1 and DTLS 1.0 no longer working at the default security level of 1 and instead requires security level 0. The security level can be changed either using the cipher string with `@SECLEVEL`, or calling **SSL_CTX_set_security_level**(3). This also means that where the signature algorithms extension is missing from a ClientHello then the handshake will fail in TLS 1.2 at security level 1. This is because, although this extension is optional, failing to provide one means that OpenSSL will fallback to a default set of signature algorithms. This default set requires the availability of SHA1.

- X509 certificates signed using SHA1 are no longer allowed at security level 1 and above.

  In TLS/SSL the default security level is 1. It can be set either using the cipher string with `@SECLEVEL`, or calling **SSL_CTX_set_security_level**(3). If the leaf certificate is signed with SHA-1, a call to **SSL_CTX_use_certificate**(3) will fail if the security level is not lowered first. Outside TLS/SSL, the default security level is -1 (effectively 0). It can be set using **X509_VERIFY_PARAM_set_auth_level**(3) or using the **-auth_level** options of the commands.

SEE ALSO

fips_module(7)

COPYRIGHT