NAME

OSSL_LIB_CTX, OSSL_LIB_CTX_new, OSSL_LIB_CTX_new_from_dispatch, OSSL_LIB_CTX_new_child, OSSL_LIB_CTX_free, OSSL_LIB_CTX_load_config, OSSL_LIB_CTX_get0_global_default, OSSL_LIB_CTX_set0_default - OpenSSL library context

SYNOPSIS

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

typedef struct ossl_lib_ctx_st OSSL_LIB_CTX;

OSSL_LIB_CTX *OSSL_LIB_CTX_new(void);
OSSL_LIB_CTX *OSSL_LIB_CTX_new_from_dispatch(const OSSL_CORE_HANDLE *handle,
                                             const OSSL_DISPATCH *in);
OSSL_LIB_CTX *OSSL_LIB_CTX_new_child(const OSSL_CORE_HANDLE *handle,
                                     const OSSL_DISPATCH *in);
int OSSL_LIB_CTX_load_config(OSSL_LIB_CTX *ctx, const char *config_file);
void OSSL_LIB_CTX_free(OSSL_LIB_CTX *ctx);
OSSL_LIB_CTX *OSSL_LIB_CTX_get0_global_default(void);
OSSL_LIB_CTX *OSSL_LIB_CTX_set0_default(OSSL_LIB_CTX *ctx);
```

DESCRIPTION

OSSL_LIB_CTX is an internal OpenSSL library context type. Applications may allocate their own, but may also use NULL to use a default context with functions that take an **OSSL_LIB_CTX** argument.

When a non default library context is in use care should be taken with multi-threaded applications to properly clean up thread local resources before the OSSL_LIB_CTX is freed. See **OPENSSL_thread_stop_ex**(3) for more information.

OSSL_LIB_CTX_new() creates a new OpenSSL library context.

OSSL_LIB_CTX_new_from_dispatch() creates a new OpenSSL library context initialised to use callbacks from the OSSL_DISPATCH structure. This is primarily useful for provider authors. The *handle* and dispatch structure arguments passed should be the same ones as passed to a provider's OSSL_provider_init function. Some OpenSSL functions, such as **BIO_new_from_core_bio**(3), require the library context to be created in this way in order to work.

OSSL_LIB_CTX_new_child() is only useful to provider authors and does the same thing as **OSSL_LIB_CTX_new_from_dispatch()** except that it additionally links the new library context to the application library context. The new library context is a full library context in its own right, but will

have all the same providers available to it that are available in the application library context (without having to reload them). If the application loads or unloads providers from the application library context then this will be automatically mirrored in the child library context.

In addition providers that are not loaded in the parent library context can be explicitly loaded into the child library context independently from the parent library context. Providers loaded independently in this way will not be mirrored in the parent library context and will not be affected if the parent library context subsequently loads the same provider.

A provider may call the function **OSSL_PROVIDER_load**(3) with the child library context as required. If the provider already exists due to it being mirrored from the parent library context then it will remain available and its reference count will be increased. If **OSSL_PROVIDER_load**(3) is called in this way then **OSSL_PROVIDER_unload**(3) should be subsequently called to decrement the reference count. **OSSL_PROVIDER_unload**(3) must not be called for a provider in the child library context that did not have an earlier **OSSL_PROVIDER_load**(3) call for that provider in that child library context.

In addition to providers, a child library context will also mirror the default properties (set via **EVP_set_default_properties**(3)) from the parent library context. If **EVP_set_default_properties**(3) is called directly on a child library context then the new properties will override anything from the parent library context and mirroring of the properties will stop.

When **OSSL_LIB_CTX_new_child()** is called from within the scope of a provider's **OSSL_provider_init** function the currently initialising provider is not yet available in the application's library context and therefore will similarly not yet be available in the newly constructed child library context. As soon as the **OSSL_provider_init** function returns then the new provider is available in the application's library context and will be similarly mirrored in the child library context.

OSSL_LIB_CTX_load_config() loads a configuration file using the given *ctx*. This can be used to associate a library context with providers that are loaded from a configuration.

OSSL_LIB_CTX_free() frees the given *ctx*, unless it happens to be the default OpenSSL library context.

OSSL_LIB_CTX_get0_global_default() returns a concrete (non NULL) reference to the global default library context.

OSSL_LIB_CTX_set0_default() sets the default OpenSSL library context to be *ctx* in the current thread. The previous default library context is returned. Care should be taken by the caller to restore the previous default library context with a subsequent call of this function. If *ctx* is NULL then no change is made to the default library context, but a pointer to the current library context is still

returned. On a successful call of this function the returned value will always be a concrete (non NULL) library context.

Care should be taken when changing the default library context and starting async jobs (see **ASYNC_start_job**(3)), as the default library context when the job is started will be used throughout the lifetime of an async job, no matter how the calling thread makes further default library context changes in the mean time. This means that the calling thread must not free the library context that was the default at the start of the async job before that job has finished.

RETURN VALUES

OSSL_LIB_CTX_new(), **OSSL_LIB_CTX_get0_global_default()** and **OSSL_LIB_CTX_set0_default()** return a library context pointer on success, or NULL on error.

OSSL_LIB_CTX_free() doesn’t return any value.

OSSL_LIB_CTX_load_config() returns 1 on success, 0 on error.

HISTORY

All of the functions described on this page were added in OpenSSL 3.0.

COPYRIGHT