Tiny cURL
API レファレンスマニュアル

このマニュアルでは Tiny cURL ライブラリの
提供する API について解説します。
内容は Tiny cURL v7.72.0 に基づいています。

1. curl_easy API：

tiny cURL API：
CURL *curl_easy_init();

この関数は easy セッションの最初に呼び出す関数で、easy インターフェイスの他の関数への入力として使用する CURL easy ハンドルを返します。この呼び出しには、操作の完了時に curl_easy_cleanup への対応する呼び出しが必要です。

curl_global_init をまだ呼び出していない場合は、curl_easy_init が自動的に呼び出します。 curl_global_init はスレッドセーフではないため、これはマルチスレッドの場合に致命的となる可能性があり、対応するクリーンアップがないためにリソースの問題が発生する可能性があります。

curl_global_init を適切に呼び出すことにより、この自動動作を許可しないことを強くお勧めします。

tiny cURL API：
CURLcode curl_easy_setopt(CURL *handle, CURLoption option, parameter);

curl_easy_setopt は、tiny-cURL に動作を指示するために使用されます。適切なオプションを設定することにより、アプリケーションは tiny-cURL の動作を変更できます。すべてのオプションは、オプションとそれに続くパラメーターで設定されます。そのパラメーターは、特定のオプションが何を期待するかに応じて、long、関数ポインター、オブジェクトポインター、または curl_off_t にすることができます。入力値が悪いと tiny-cURL の動作が悪くなる可能性があるため、このマニュアルを注意深くお読みください。各関数呼び出しで設定できるオプションは 1 つだけです。一般的なアプリケーションは、セットアップフェーズで多くの curl_easy_setopt 呼び出しを使用します。

この関数呼び出しで設定されたオプションは、このハンドルを使用して実行される今後のすべての転送に対して有効です。オプションは転送間でリセットされることは決してないため、異なるオプションを使用した後続の転送が必要な場合は、転送間でオ

プションを変更する必要があります。オプションで、curl_easy_reset を使用してすべてのオプションを内部デフォルトにリセットできます。

'char *'引数として tiny-cURL に渡される文字列は、ライブラリによってコピーされます。ポインタ引数に関連付けられた文字列ストレージは、curl_easy_setopt が戻った後に破棄または再利用できます。このルールの唯一の例外は実際には CURLOPT_POSTFIELDS ですが、文字列 CURLOPT_COPYPOSTFIELDS をコピーする代替手段には、理解する必要のあるいくつかの使用特性があります。この関数は、CURL_MAX_INPUT_LENGTH（8 MB）より長い入力文字列を受け入れません。

オプションが設定される順序は重要ではありません。

バージョン 7.17.0 より前では、文字列はコピーされませんでした。代わりに、tiny-cURL がそれらを必要としなくなるまで、ユーザーはそれらを利用可能に保つことを余儀なくされました。

ハンドルは、curl_easy_init または curl_easy_duphandle 呼び出しからの戻りコードです。

tiny cURL API：
CURLcode curl_easy_perform(CURL *easy_handle);

curl_easy_init およびすべての curl_easy_setopt 呼び出しが行われた後にこの関数を呼び出し、オプションで説明されているように転送を実行します。返された curl_easy_init 呼び出しと同じ easy_handle を入力として呼び出す必要があります。

curl_easy_perform は、リクエスト全体をブロックして実行し、完了すると、失敗した場合はそれ以前に戻ります。非ブロッキング動作については、curl_multi_perform を参照してください。

同じ easy_handle を使用しながら、curl_easy_perform に対して任意の数の呼び出しを行うことができます。複数のファイルを転送する場合は、転送することをお勧めします。次に、tiny-cURL は、次の転送で同じ接続を再利用しようとします。これにより、操作が高速化され、CPU の負荷が軽減され、ネットワークリソースの使用量が削減されます。次の curl_easy_perform のオプションを設定するには、呼び出しの間に curl_easy_setopt を使用する必要があることに注意してください。

同じ easy_handle を使用して、2 つの場所からこの関数を同時に呼び出さないでください。関数を最初に戻してから、もう一度呼び出すようにします。並列転送が必要な場合は、いくつかの curleasy_handles を使用する必要があります。

ネットワーク転送は、データをピアに、またはピアから移動します。アプリケーションは、CURLOPT_WRITEFUNCTION および CURLOPT_WRITEDATA オプションを設定することにより、tiny-cURL にデータの受信方法を指示します。 tiny-cURL に送信するデータを指示するために、さらにいくつかの選択肢がありますが、2 つの一般的な選択肢は CURLOPT_READFUNCTION と CURLOPT_POSTFIELDS です。

easy_handle はマルチハンドルに追加されますが、curl_easy_perform では使用できません。

tiny cURL API：
void curl_easy_cleanup(CURL *handle);

この関数は、easy セッションの最後の関数として呼び出す必要があります。これは curl_easy_init 関数の反対であり、curl_easy_init 呼び出しが返した入力と同じハンドルで呼び出す必要があります。

これにより、このハンドルが使用していたすべての接続が閉じられ、転送中にマルチハンドルに接続されていない限り、これまで開いたままになっている可能性があります。より多くのファイルを転送する場合は、この関数を呼び出さないでください。ハ

ンドルを再利用することが、tiny-cURL のパフォーマンスを向上させるための鍵となります。

場合によっては、curl_easy_cleanup 内から進行状況コールバックまたはヘッダーコールバックが呼び出されることがあります（以前に curl_easy_setopt を使用してハンドルに設定されている場合）。 tiny-cURL が接続をシャットダウンすることを決定し、プロトコルが切断前にコマンド/応答シーケンスを必要とする種類のものである場合のように。このようなプロトコルの例は、FTP、POP3、および IMAP です。

この関数が呼び出されて戻った後のハンドルは使用できません。 curl_easy_cleanup は、ハンドルとそれに関連するすべてのメモリを強制終了します。

マルチインターフェースで使用されていたイージーハンドルを閉じるには、必ず最初に curl_multi_remove_handle を呼び出して、閉じる前にマルチハンドルからハンドルを削除してください。

ハンドルに NULL ポインターを渡すと、この関数はアクションなしですぐに戻ります。

tiny cURL API：
CURL_EXTERN CURLcode curl_easy_getinfo(CURL *curl, CURLINFO info, ...);

この関数を使用して、Curl セッションから内部情報をリクエストします。3 番目のアーギュメントは、long のポインター、char のポインター、または double へのポインターでなければなりません（ドキュメントが他の場所で説明しているように）。ポインターで示されたデータはそれに応じて埋められ、関数が CURL_OK を返す場合にのみ依存することができます。この関数は、実行された転送の後に使用されることを目的としています。この関数のすべての結果は、転送が完了するまで不定です。

tiny cURL API：
CURL_EXTERN CURL *curl_easy_duphandle(CURL *curl);

渡されたハンドル用に設定された同じオプションを備えた新しい cURL セッションハンドルを作成します。ハンドルの複製は、データとオプション、内部状態情報、および永続的な接続などのもののみの問題である可能性があります。すべてのスレッドで一連の同一の curl_easy_setopt（）を避けて、新しいスレッドごとに curl_easy_duphandle（）を実行できる場合、マルチスレッドアプリケーションで役立ちます。

tiny cURL API：

```
CURL_EXTERN void curl_easy_reset(CURL *curl);
```

cURL ハンドルをデフォルト値にリセットします。これにより、ハンドルが作成されたときと同じ状態にハンドルを戻します。ライブ接続、セッション ID キャッシュ、DNS キャッシュ、Cookie を保持します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_easy_recv(CURL *curl, void *buffer, size_t buflen,
                                    size_t *n);
```

接続されたソケットからデータを受信します。curlopt_connect_only オプションで curl_easy_perform（）が成功した後に使用します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_easy_send(CURL *curl, const void *buffer,
                                    size_t buflen, size_t *n);
```

接続されたソケットにデータを送信します。curlopt_connect_only オプションで curl_easy_perform（）が成功した後に使用します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_easy_upkeep(CURL *curl);
```

指定されたセッションハンドルの接続 UPKEEP を実行します。

2. cURL API

tiny cURL API：

```
CURL_EXTERN curl_mime *curl_mime_init(CURL *easy);
```

mime コンテキストを作成し、ハンドルを返します。easy 引数はターゲットハンドルです。

tiny cURL API：

```
CURL_EXTERN void curl_mime_free(curl_mime *mime);
```

mime ハンドルとその下部構造を解放します。

tiny cURL API：

```
CURL_EXTERN curl_mimepart *curl_mime_addpart(curl_mime *mime);
```

指定された mime コンテキストに新しい空の部分を追加し、作成された部品にハンドルを返します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_name(curl_mimepart *part, const char *name);
```

MIME/フォーム part 名を設定します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_filename(curl_mimepart *part,
                                        const char *filename);
```

part のリモートファイル名を設定します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_type(curl_mimepart *part, const char *mimetype);
```

part のタイプを設定します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_encoder(curl_mimepart *part,
                                       const char *encoding);
```

mime データ転送エンコーダーを設定します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_data(curl_mimepart *part,
                                    const char *data, size_t datasize);
```

メモリデータから part データソースを設定します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_filedata(curl_mimepart *part,
                                        const char *filename);
```

名前付きファイルから mimepart データソースを設定します。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_mime_data_cb(curl_mimepart *part,
                                       curl_off_t datasize,
                                       curl_read_callback readfunc,
                                       curl_seek_callback seekfunc,
                                       curl_free_callback freefunc,
                                       void *arg);
```

コールバック関数から part データソースを設定します。

tiny cURL API：

```
CURL_EXTERN CURLFORMcode curl_formadd(struct curl_httppost **httppost,
                                      struct curl_httppost **last_post,
```

...);

マルチパートフォームポストを構築するためのかなり高度な機能。それぞれの呼び出しは、完全な投稿を構築するための部分を追加します。次に、curlopt_httppost を使用して tiny-cURL に送信します。

tiny cURL API：
CURL_EXTERN int curl_formget(struct curl_httppost *form, void *arg,
curl_formget_callback append);

curl_formadd（）で構築された curl_httppost 構造体をシリアル化します。
curl_formget_callback 関数に渡される 2 番目の引数としてボイドポインターを受け入れます。成功に 0 を返します。以前に curl_formadd（）で構築されたマルチパートフォームポストを開放。

tiny cURL API：
CURL_EXTERN void curl_formfree(struct curl_httppost *form);

url_formadd()でビルドされたリソースを解放します。

tiny cURL API：
CURL_EXTERN char *curl_getenv(const char *variable);

使用状況が完了した後、curl_free（）されなければならない malloc された文字列を返します。非推奨-Lib/readme.curlx を参照してください

tiny cURL API：
CURL_EXTERN char *curl_version(void);

tiny-cURL バージョンの静的 ASCII 文字列を返します。

tiny cURL API：

```
CURL_EXTERN char *curl_easy_escape(CURL *handle,
                                   const char *string,
                                   int length);
```

URL 文字列をエスケープ形式に変換します（URL に不正な文字チェックしすべての文字を変換して、%XX の形式に変換します）。この関数は、エラーが発生した場合、新しい割り当てられた文字列または null を返します。

tiny cURL API：

```
CURL_EXTERN char *curl_easy_unescape(CURL *handle,
                                     const char *string,
                                     int length,
                                     int *outlength);
```

エスケープ形式の URL をエンコード（すべての％xx コードを 8 ビットバージョンに変換します）。この関数は、エラーが発生した場合、新しい割り当てられた文字列または null を返します。コンバージョン注：ASCII 以外のプラットフォームでは、ASCII％XX コードがホストエンコーディングに変換されます。

tiny cURL API：

```
CURL_EXTERN void curl_free(void *p);
```

割り当てを行った同じ翻訳単位のリソースをすべて解放。

tiny cURL API：

```
CURL_EXTERN CURLcode curl_global_init(long flags);
```

curl_global_init（）は、tiny-cURL を使用するアプリケーションごとに、および他の tiny-cURL 関数の呼び出しの前に 1 回だけ呼び出す必要があります。この関数はスレッドセーフではありません！

tiny cURL API：

```
CURL_EXTERN CURLcode curl_global_init_mem(long flags,
                                          curl_malloc_callback m,
                                          curl_free_callback f,
                                          curl_realloc_callback r,
                                          curl_strdup_callback s,
                                          curl_calloc_callback c);
```

curl_global_init（）または curl_global_init_mem（）は、tiny-cURL を使用する各アプリケーションに対して正確に 1 回呼び出す必要があります。この関数は、Tiny-cURL の初期化とユーザー定義のメモリ管理コールバック関数を設定するために使用できます。ユーザーは、メモリ管理ルーチンを実装してメモリリークをチェックし、カールライブラリなどの誤用を確認できます。ユーザー登録コールバックルーチンは、Malloc、free などのシステムメモリ管理ルーチンではなく、このライブラリによって呼び出されます。

tiny cURL API：

```
CURL_EXTERN void curl_global_cleanup(void);
```

curl_global_cleanup（）は、tiny-cURL を使用する各アプリケーションに対して正確に 1 回呼び出す必要があります

tiny cURL API：

```
CURL_EXTERN CURLsslset curl_global_sslset(curl_sslbackend id, const char *name,
                                          const curl_ssl_backend ***avail);
```

複数の SSL バックエンドで構築されると、curl_global_sslset（）を使用すると、1 つを選択できます。この関数は 1 回しか呼ばれず、前に* curl_global_init（）と呼ぶ必要があります。バックエンドは ID で識別できます（例：curlsslbackend_openssl）。バックエンドは、名前パラメーター（ID として-1 を渡す）を介して指定することもできます。ID と名前の両方が指定されている場合、名前は無視されます。ID も名前も指定されていない場合、関数は curlsslset_unknown_backend で失敗し、利用可能なバックエン

ドのヌル終了リストに「avail」ポインターを設定します。成功すると、関数は curlsslset_ok を返します。指定された SSL バックエンドが使用できない場合、関数は curlsslset_unknown_backend を返し、利用可能な SSL バックエンドのヌル終端リストに「avail」ポインターを設定します。SSL バックエンドは一度だけ設定できます。すでに設定されている場合、それを変更しようとするその後の試みは、curlsslset_too_late になります。

tiny cURL API：

```
CURL_EXTERN struct curl_slist *curl_slist_append(struct curl_slist *,
                                                 const char *);
```

文字列をリンクリストに追加します。リストが存在しない場合、最初に作成されます。アプリがある後、新しいリストを返します。

tiny cURL API：

```
CURL_EXTERN void curl_slist_free_all(struct curl_slist *);
```

以前に作成された curl_slist を解放します。

tiny cURL API：

```
CURL_EXTERN time_t curl_getdate(const char *p, const time_t *unused);
```

最初の引数で与えられた時間文字列の 1970 年 1 月 1 日から数秒で時間を返します。2 番目のパラメーターの時間引数は未使用であり、null に設定する必要があります。

tiny cURL API：

```
CURL_EXTERN curl_version_info_data *curl_version_info(CURLversion);
```

この関数は、バージョン情報構造の静的コピーにポインターを返します。

tiny cURL API：

```
CURL_EXTERN const char *curl_easy_strerror(CURLcode);
```

curl_easy_strerror 関数を使用して、curlcode 値を同等の人間の読み取り可能なエラー文字列に変えることができます。これは、意味のあるエラーメッセージを印刷するのに役立ちます。

tiny cURL API：
CURL_EXTERN const char *curl_share_strerror(CURLSHcode);

curl_share_strerror 関数を使用して、curlshcode 値を同等の人間の読み取り可能なエラー文字列に変えることができます。これは、意味のあるエラーメッセージを印刷するのに役立ちます。

tiny cURL API：
CURL_EXTERN CURLcode curl_easy_pause(CURL *handle, int bitmask);

curl_easy_pause 関数は、透過を一時停止または整理します。ビットマスクを設定して新しい状態を選択し、以下の定義を使用します。

```
#define CURLPAUSE_RECV      (1<<0)
#define CURLPAUSE_RECV_CONT (0)

#define CURLPAUSE_SEND      (1<<2)
#define CURLPAUSE_SEND_CONT (0)

#define CURLPAUSE_ALL       (CURLPAUSE_RECV|CURLPAUSE_SEND)
#define CURLPAUSE_CONT      (CURLPAUSE_RECV_CONT|CURLPAUSE_SEND_CONT)
```