JUNGO

# WinDriver

# USB API リファレンス

エクセルソフト株式会社

**Jungo Ltd.**

POB 8493 Netanya Zip - 42504 Israel

Phone (USA) 1-877-514-0537 (WorldWide) +972-9-8859365

Fax (USA) 1-877-514-0538 (WorldWide) +972-9-8859366

## ご注意



**エクセルソフト株式会社**

〒108-0073 東京都港区三田 3-9-9 森伝ビル 6F

TEL 03-5440-7875 FAX 03-5440-7876

E-MAIL: xlsoftkk@xlsoft.com

Home Page: http://www.xlsoft.com/

Rev. 10.10 - 12/2009

# 目次

# 付録 A
# WinDriver USB PC Host API のリファレンス

**注意**

この関数リファレンスは、C 言語指向です。WinDriver の .NET、Visual Basic および Delphi API を C コードに似た形で表現することで、すべてのユーザーに対しての理解度を向上します。各言語の実装および使用例は、WinDriver .NET、VB および Delphi ソース コードを参照してください。

## A.1 WinDriver USB (WDU) ライブラリの概要

このセクションでは、以下の内容を含んだ WinDriver の USB ライブラリ (WDU) について説明します。

- `WDU_xxx` APIのコール順序の概要 - セクション A.1.1。
- 以前の WinDriver USB API (バージョン 5.22 以降) で開発されたコードのアップグレード方法 (向上した `WDU_xxx` API の使用) – セクション A.1.3。
  WinDriver の以前のバージョンで開発した USB ドライバ コードをアップグレードしない場合、このセクションをスキップしてください。

WDU ライブラリのインターフェイスは、WDU API を呼ぶソースファイルが含まれた **`WinDriver/include/wdu_lib.h`** および **`WinDriver/include/windrvr.h`** ヘッダー ファイルに保存されています。(**`wdu_lib.h`** は、既に **`windrvr.h`** に含まれています。)

### A.1.1 WD_DriverName()

#### 目的

- 呼び出し元アプリケーションにより使用される WinDriver カーネル モジュールの名前を指定します。

注意:

- デフォルトのドライバ名は **`windrvr6`** です。この関数が呼び出されない場合に使用されます。
- この関数は、サンプルおよび DriverWizard で生成される WinDriver アプリケーションのように、他の WinDriver 関数 (`WD_Open()` / `WDU_Init()` を含む) を呼び出す前に、アプリケーションの始めで 1 度だけ呼び出してください。サンプルおよび DriverWizard で生成される WinDriver アプリケーションでは、デフォルトのドライバ名 (**`windrvr6`**) でこの関数を呼び出しています。
- Windows および Linux では、WinDriver ユーザーズ ガイドのセクション 15.2 で説明するように、WinDriver カーネル モジュールの名前 (**`windrvr6.sys/.o/.ko`**) を変更する場合、アプリケーションによる `WD_DriverName()` の呼び出しに新しい名前が使用されていることを確認してください。
- `WD_DriverName()` 関数を使用するためには、`WD_DRIVER_NAME_CHANGE` プリプロセッサー フラグ (例: Visual Studio および gcc の場合は `-DWD_DRIVER_NAME_CHANGE`) を使用して、ユーザーモードのドライバ プロジェクトをビルドする必要があります。サンプルおよび DriverWizard で生成される Windows および Linux の WinDriver プロジェクトまたは makefile では、既にこのプリプロセッサ フラグが設定されています。

### プロトタイプ

```
const char*  DLLCALLCONV WD_DriverName(const char* sName);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ sName | const char* | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| sName | アプリケーションにより使用される WinDriver カーネル モジュールの名前<br>注意: ドライバ名には、ファイル拡張子を含めないでください。たとえば、**windrvr6.sys** や **windrvr6.o** ではなく、**windrvr6** とします。 |

### 戻り値

正常終了した場合、指定したドライバ名を返します。失敗した場合 (例: 同じアプリケーションから 2 度呼び出された場合)、NULL を返します。

### 注釈

- WinDriver ユーザーズ ガイドのセクション 15.2 で説明するように、WinDriver カーネル モジュールの名前変更は、Windows および Linux でサポートしています。
- Windows CE では、WD_DriverName() 関数は、常にデフォルトの WinDriver カーネル モジュール名 (**windrvr6**) で呼び出す必要があります。そうでない場合、この関数を呼び出さないでください。

## A.1.2 WinDriver USB のコール順序

WinDriver の WDU_xxx USB API は、ユーザー モード USB アプリケーションと USB デバイス間のイベントドリブン転送をサポートするよう設計されています。これは以前のバージョンにはありませんでした。以前のバージョンでは、USB デバイスは関数呼出しの特定の順序を使用して初期化およびコントロールされていました。

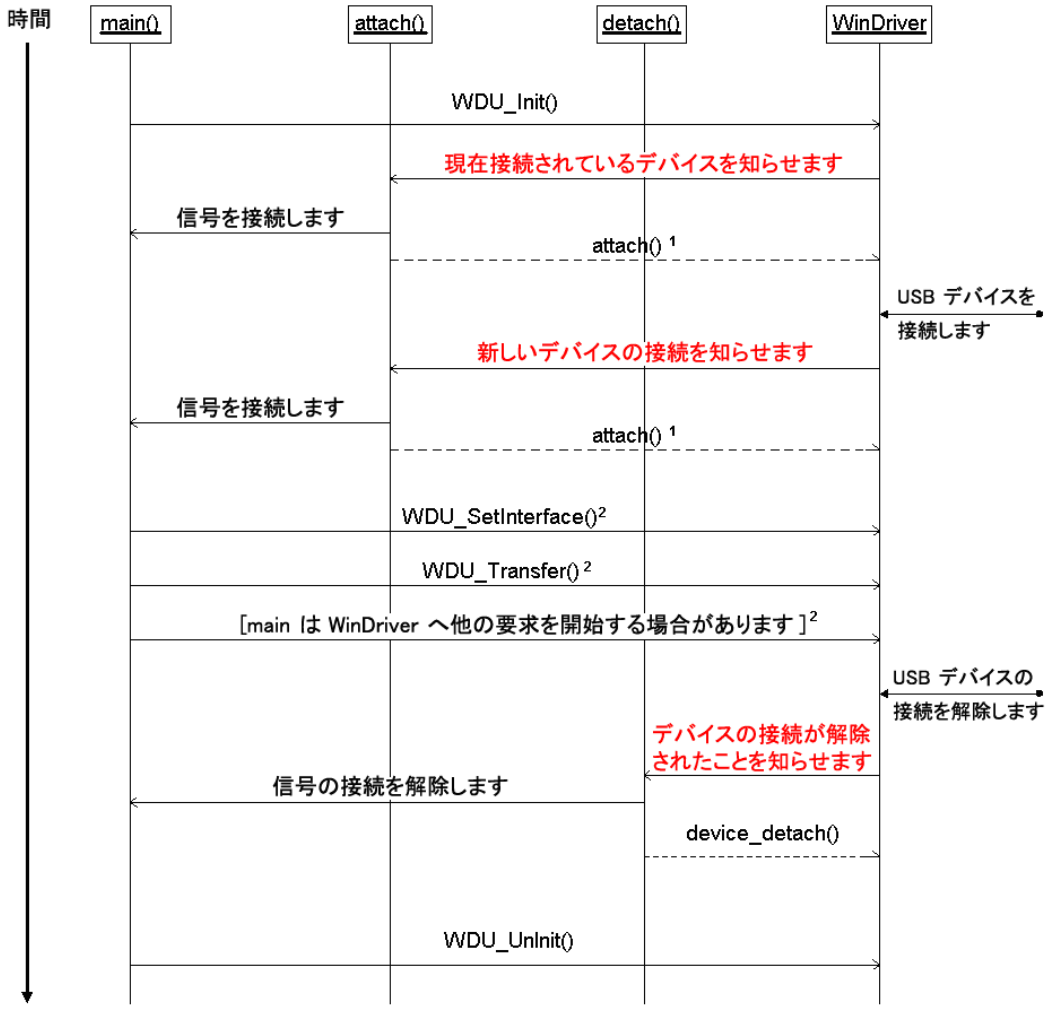
次のセクションでは、3 つのユーザー コールバック関数 (WDU_ATTACH_CALLBACK [A.2.1]、WDU_DETACH_CALLBACK [A.2.2]、および WDU_POWER_CHANGE_CALLBACK [A.2.3]) を実装できます。これらの関数は、USB デバイスの装着または検出などの、システム イベントの発生時に、アプリケーションに通知するのに使用されます。最善のパフォーマンスは、最小の実行が 3 つの関数で行われます。

アプリケーションが WDU_Init() [A.3.1] を呼び出し、デバイスが関連しているかしていないかをシステムが識別するための基準を設けます。WDU_Init()がユーザー コールバック関数へポインタを渡す必要があります。

アプリケーションはただイベントの通知を待機するだけです。通知を受信するまで、処理が続きます。アプリケーションは、下記の高レベルまたは低レベル API で定義したどの関数でも使用します。高レベル関数は低レベル関数 (WinDriver カーネル モジュールとユーザー モード アプリケーション間の通信を可能にする IOCTL を使用する) を使用します。

既存の場合、アプリケーションは指定の基準に一致するデバイスへのリスンを停止し、それらのデバイスへのコールバックの通知を解除するのに `WDU_Uninit()` [A.3.6] 関数を呼び出します。

次の図は、上記で説明したコール順序を示しています。縦線が、関数またはプロセスを表しています。横線の矢印が、信号または要求を表し、開始位置から受信までを描いています。上から下が時間軸です。




1 WDU_Init() の呼び出しで、WD_ACKNOWLEDGE フラグがセットされている場合、関数がデバイスの制御を許可する場合は attach() のコールバックが TRUE を返します。許可しない場合は、FALSE を返します。

2 attach() が TRUE を返した場合のみ可能です

**図 A.1: WinDriver USB の呼び出し順序**

次のコードをユーザー モード アプリケーションのコードのフレームワークとして使用できます。

```
attach()
{
    ...
    if this is my device
        /*
        Set the desired alternate setting ;
        Signal main() about the attachment of this device
        */

        return TRUE;
    else
        return FALSE;
}

detach()
{
    ...
    signal main() about the detachment of this device
    ...
}

main()
{
    WDU_Init(...);

    ...
    while (...)
    {
        /* wait for new devices */

        ...

        /* issue transfers */

        ...
    }
    ...
    WDU_Uninit();
}
```

### A.1.3 WD_xxx USB API から WDU_xxx API へのアップグレード

バージョン 6.00 から提供されている WinDriver の WDU_xxx USB API は、ユーザー モード USB アプリケーションと USB デバイス間のイベント ドリブン転送をサポートするよう設計されています。これは以前のバージョンにはありませんでした。以前のバージョンでは、USB デバイスは関数呼出しの特定の順序を使用して初期化およびコントロールされていました。

この変更の結果、Microsoft Windows だけではなく対応するすべてのプラット フォーム上で WinDriver 6.X で動作するように、WinDriver の以前のバージョンのインターフェイス用に設計された USB アプリケーションを修正する必要があります。セクション A.1.2 で説明した meta-code の一部のフレームワークと一致するようにアプリケーションのコードを作り直す必要があります。

更に、USB API を定義している関数が変更されています。次のセクションで説明しますが、新しい関数は、改良したユーザー モード USB アプリケーションと WinDriver カーネル モード間のインターフェイスを提供します。新しい関数は引数を直接、受信します。古い関数は、構造体を使用して引数を受信していました。

次の表に、左側の項目に古い関数を、右側の項目に各古い関数を置き換えた関数を表示しています。この表を使用して、新しいコードではどの新しい関数を使用するかを素早く判断します。

| 問題 | 解決方法 |
|---|---|
| **高レベル API** | |
| この関数が... | 次のように置き換えられました... |
| WD_Open()<br>WD_Version()<br>WD_UsbScanDevice() | WDU_Init() [A.3.1] |
| WD_UsbDeviceRegister() | WDU_SetInterface() [A.3.2] |
| WD_UsbGetConfiguration() | WDU_GetDeviceInfo() [A.3.4] |
| WD_UsbDeviceUnregister() | WDU_Uninit() [A.3.6] |
| **低レベル API** | |
| この関数が... | 次のように置き換えられました... |
| WD_UsbTransfer() | WDU_Transfer() [A.3.8.1]<br>WDU_TransferDefaultPipe() [A.3.8.3]<br>WDU_TransferBulk() [A.3.8.4]<br>WDU_TransferIsoch() [A.3.8.5]<br>WDU_TransferInterrupt() [A.3.8.6] |
| USB_TRANSFER_HALT option | WDU_HaltTransfer() [A.3.8.2] |
| WD_UsbResetPipe() | WDU_ResetPipe() [A.3.10] |
| WD_UsbResetDevice()<br>WD_UsbResetDeviceEx() | WDU_ResetDevice() [A.3.11] |

# A.2 USB – ユーザー コールバック関数

## A.2.1 WDU_ATTACH_CALLBACK()

### 目的

- WinDriver は、まだ他のドライバに制御されていない、指定した基準に一致した新しいデバイスが装着されたときに、この関数を呼び出します。
  この callback を各一致するインターフェイスに対して一度呼びます。

### プロトタイプ

```
typedef BOOL (DLLCALLCONV *WDU_ATTACH_CALLBACK)(
    WDU_DEVICE_HANDLE hDevice,
    WDU_DEVICE *pDeviceInfo,
    PVOID pUserData);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ pDeviceInfo | WDU_DEVICE* | 入力 |

| | | |
|---|---|---|
| ➢ pUserData | PVOID | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| pDeviceInfo | USB デバイス情報の構造体 [A.4.3] へのポインタ。関数の終了まで有効。 |
| pUserData | コールバックするユーザーモード データ。イベント テーブル パラメータ (pEventTable->pUserData) で WDU_Init() [A.3.1] へ渡されます。 |

### 戻り値

WDU_Init() [A.3.1] (dwOptions 引数内で) を呼び出すように WD_ACKNOWLEDGE フラグを設定した場合、コールバック関数がデバイスを制御するかチェックし、制御する場合は、TRUE を返し、そうでない場合、FALSE を返します。
WDU_Init()への呼び出しで、WD_ACKNOWLEDGE フラグを設定しない場合、コールバック関数の戻り値は、意味がありません。

## A.2.2 WDU_DETACH_CALLBACK()

### 目的

- WinDriver は、デバイスがシステムから取り外されたときに、この関数を呼び出します。

### プロトタイプ

```
typedef void (DLLCALLCONV *WDU_DETACH_CALLBACK)(
    WDU_DEVICE_HANDLE hDevice,
    PVOID pUserData);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ pUserData | PVOID | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| pUserData | コールバックするユーザーモード データ。イベント テーブル パラメータ (pEventTable->pUserData) で WDU_Init() [A.3.1] へ渡されます。 |

### 戻り値

なし。

## A.2.3 WDU_POWER_CHANGE_CALLBACK()

### 目的

- WinDriver は、デバイスの電源の設定を変更したときに、この関数を呼び出します。

### プロトタイプ

```
typedef BOOL (DLLCALLCONV *WDU_POWER_CHANGE_CALLBACK)(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPowerState,
    PVOID pUserData);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ dwPowerState | DWORD | 入力 |
| ➢ pUserData | PVOID | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| dwPowerState | 選択した電源状態の番号。 |
| pUserData | コールバックするユーザーモード データ。イベント テーブル パラメータ (`pEventTable->pUserData`) で `WDU_Init()` [A.3.1] へ渡されます。 |

### 戻り値

TRUE / FALSE。現在、戻り値に重要な意味はありません。

### 注釈

- Windows 2000 以降の Windows オペレーティング システムでのみ、この callback をサポートします。

# A.3 USB – 関数

このセクションで説明する関数は、**WinDriver/include/wdu_lib.h** ヘッダー ファイルに宣言されています。

## A.3.1 WDU_Init()

### 目的

- 入力基準に一致するデバイスへリスンを開始し、デバイスの通知コールバックを登録します。

### プロトタイプ

```
DWORD WDU_Init(
    WDU_DRIVER_HANDLE *phDriver,
    WDU_MATCH_TABLE *pMatchTables,
    DWORD dwNumMatchTables,
    WDU_EVENT_TABLE *pEventTable,
    const char *sLicense,
    DWORD dwOptions);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ phDriver | WDU_DRIVER_HANDLE * | 出力 |
| ➢ pMatchTables | WDU_MATCH_TABLE* | 入力 |
| ➢ dwNumMatchTables | DWORD | 入力 |
| ➢ pEventTable | WDU_EVENT_TABLE* | 入力 |
| ➢ sLicense | const char* | 入力 |
| ➢ dwOptions | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| phDriver | イベントおよび基準の登録へのハンドル。 |
| pMatchTables | デバイスの基準を定義しているマッチ テーブル [A.4.1] の配列。 |
| dwNumMatchTables | `pMatchTables` のエレメントの番号。 |
| pEventTable | ユーザーモードのデバイス ステータス変更通知コールバック関数 [A.2] およびコールバック関数に渡されるデータのアドレスを保持するイベント テーブル構造体 [A.4.2] へのポインタ。 |
| sLicense | WinDriver のライセンス文字列。 |
| dwOptions | 0 または :<br>• **WD_ACKNOWLEDGE** - WDU_ATTACH_CALLBACK [A.2.1] に |

| | |
|---|---|
| | 値が戻ってくるときに、ユーザーはデバイスを制御できます。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.2 WDU_SetInterface()

### 目的

- 指定したインターフェイスの代替の設定を設定します。

### プロトタイプ

```
DWORD WDU_SetInterface(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwInterfaceNum,
    DWORD dwAlternateSetting);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwInterfaceNum | DWORD | 入力 |
| ➢ dwAlternateSetting | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| dwInterfaceNum | インターフェイスの番号。 |
| dwAlternateSetting | 要求した代替の設定値。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

**Windows CE** (Windows 2000 / XP / Server 2003 / Server 2008 / Vista / 7 とは対照的に) では、すべてのパイプが必要ない場合でも、`WDU_SetInterface()` は、指定した代替設定のすべてのパイプを開こうとします。この理由は、Windows CE には、デバイス上で同時に開くことができるパイプの総数に制限があります (参考: http://msdn.microsoft.com/en-us/library/ms919318.aspx)。すべてのパイプを開くことによって、必要に応じて、パイプが利用可能かドライバは確認します。
Windows CE USB ホスト コントローラ ドライバの `LPOPEN_PIPE` コールバックを使用して、パイプを開きま

す。このコールバックへの呼び出しの失敗で、WDU_SetInterface()も失敗するモバイル デバイスがあります。この問題を解決するには、デバイスの USB ホスト コントローラ ドライバをアップグレードしてください。

## A.3.3 WDU_GetDeviceAddr()

### 目的

- 指定されたデバイスの USB アドレスを取得します。

### プロトタイプ

```
DWORD WDU_GetDeviceAddr(
    WDU_DEVICE_HANDLE hDevice,
    ULONG *pAddress);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ pAddress | ULONG | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| pAddress | 関数により返されるアドレスへのポインタ |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- Windows 2000 およびそれ以降でのみ、この関数をサポートします。

## A.3.4 WDU_GetDeviceRegistryProperty()

### 目的

- 指定した USB デバイスの特定のレジストリ プロパティを取得します。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_GetDeviceRegistryProperty(
    WDU_DEVICE_HANDLE hDevice,
```

```
    PVOID pBuffer,
    PDWORD pdwSize,
    WD_DEVICE_REGISTRY_PROPERTY property);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ pBuffer | PVOID | 出力 |
| ➢ pdwSize | PDWORD | 入力 / 出力 |
| ➢ property | WD_DEVICE_REGISTRY_PROPERTY | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| pBuffer | 要求したレジストリ プロパティで満たされるユーザーが割り当てたバッファへのポインタ。**pdwSize** パラメータの入力値で示したように、バッファ サイズが十分な場合 (つまり、**pdwSize** で返るプロパティ サイズ以上の場合) のみ、この関数はバッファを満たします。<br>この関数を使用してレジストリ プロパティのサイズを取得する場合のみ、**pBuffer** を NULL に設定します。 |
| pdwSize | 入力の場合、ユーザーが指定したバッファ (**pBuffer**) のサイズを示す値へのポインタ。**pBuffer** に NULL を設定した場合、このパラメータの入力値は、無視されます。<br>出力の場合、レジストリ プロパティを格納するために必要なバッファ サイズを示す値へのポインタ。 |
| property | 取得されるレジストリ プロパティの ID (WD_DEVICE_REGISTRY_PROPERTY 列挙型の説明を参照してください)。<br>注意: 文字列のレジストリ プロパティは WCHAR フォーマットです。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- 指定したユーザーのバッファー (**pBuffer**) のサイズ (***pdwSize** – 入力) が、要求したレジストリ プロパティを確保するのに十分でない場合、この関数は WD_INVALID_PARAMETER を返します。
- Windows 2000 およびそれ以降でのみこの関数をサポートします。

## A.3.5 WDU_GetDeviceInfo()

### 目的

- すべてのデバイス ディスクリプタを含む、デバイスからの設定情報を取得します。

**注意:** 呼び出し元は、関数によって返される*ppDeviceInfo ポインタを解放するために、WDU_PutDeviceInfo() [A.3.5] を呼び出す必要があります。

### プロトタイプ

```
DWORD WDU_GetDeviceInfo(
    WDU_DEVICE_HANDLE hDevice,
    WDU_DEVICE **ppDeviceInfo);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ ppDeviceInfo | WDU_DEVICE** | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| ppDeviceInfo | USB デバイス情報の構造体 [A.4.3] へのポインタをポイントします。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.6 WDU_PutDeviceInfo()

### 目的

- 以前の WDU_GetDeviceInfo() [A.3.4] の呼び出しで割り当てられた、デバイス情報のポインタを受信します。

### プロトタイプ

```
void WDU_PutDeviceInfo(WDU_DEVICE *pDeviceInfo);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ pDeviceInfo | WDU_DEVICE* | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| pDeviceInfo | WDU_GetDeviceInfo() [A.3.4] への以前の呼び出しで返された USB デバイス情報の構造体へのポインタ [A.4.3] |

### 戻り値

なし。

## A.3.7 WDU_Uninit()

### 目的

- 入力基準に一致するデバイスへリスンを開始し、デバイスの通知コールバックを解除します。

### プロトタイプ

```
void WDU_Uninit(WDU_DRIVER_HANDLE hDriver);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDriver | WDU_DRIVER_HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDriver | WDU_Init() [A.3.1] から受信した登録をハンドルします。 |

### 戻り値

なし。

## A.3.8 シングル ブロッキング転送関数

このセクションでは、WinDriver のシングル ブロッキング データ転送関数について説明します。

詳細は、WinDriver ユーザーズ ガイドのセクション 9.5 を参照してください。

### A.3.8.1 WDU_Transfer()

### 目的

- デバイスへまたはデバイスからデータを転送します。

### プロトタイプ

```
DWORD WDU_Transfer(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum,
    DWORD fRead,
    DWORD dwOptions,
    PVOID pBuffer,
    DWORD dwBufferSize,
    PDWORD pdwBytesTransferred,
    PBYTE pSetupPacket,
    DWORD dwTimeout);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwPipeNum | DWORD | 入力 |
| ➢ fRead | DWORD | 入力 |
| ➢ dwOptions | DWORD | 入力 |
| ➢ pBuffer | PVOID | 入力 |
| ➢ dwBufferSize | DWORD | 入力 |
| ➢ pdwBytesTransferred | PDWORD | 出力 |
| ➢ pSetupPacket | PBYTE | 入力 |
| ➢ dwTimeout | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | `WDU_Init()` [A.3.1] から取得したデバイス / インターフェイスのユニークな識別子 |
| dwPipeNum | データ転送に使用するパイプの番号 |
| fRead | read (読み取り) の場合は TRUE、write (書き込み) の場合 FALSE |
| dwOptions | 以下のいずれかのフラグのビット マスク。<br>• **`USB_ISOCH_NOASAP`** - アイソクロナス転送。このオプションを設定することで、データ転送の実行中に下位 USB スタック ドライバ (**`usbd.sys`**) は (次に利用可能なフレームの代わりに) 現在のフレーム番号を使用するよう命令します。低スピードまたは高スピードのデバイス (USB 1.1 のみ) で転送中に未使用のフレームがある場合、このフラグを使用します (Windows のみ。ただし、Windows CE は除く)。<br>• **`USB_ISOCH_RESET`** -データ転送を行なう前にアイソクロナス パイプをリセットします。また、マイナーなエラーが発生した際にパイプをリセットします (データ転送は続行されます)。<br>• **`USB_ISOCH_FULL_PACKETS_ONLY`** -パケット サイズ以下の |

| | |
|---|---|
| | データをアイソクロナス パイプに転送しません。<br>• **USB_BULK_INT_URB_SIZE_OVERRIDE_128K** - URB (USB Request Block) のサイズを 128KB に制限します。 |
| pBuffer | データ バッファのアドレス |
| dwBufferSize | 転送するバイト数。バッファ サイズは、デバイスの最大パケット サイズに制限されません。このため、バッファ サイズを最大パケット サイズの倍数に設定して、より大きなバッファを使用できます。大きなバッファを使用してコンテキスト スイッチの数を減らし、パフォーマンスを向上します。 |
| pdwBytesTransferred | 実際に転送されたバイト数 |
| pSetupPacket | パイプを制御するために転送する 8 バイトのパケット |
| dwTimeout | 転送にかかるミリ秒 (ms) 単位での最大所要時間。0 の場合、タイムアウトせずに、無限に待ちます。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- タイムアウト (`dwTimeout` パラメータ) の最小単位は、オペレーティング システムのスケジューラのタイムスロットに依存します。たとえば、Windows の場合、タイムアウトの最小単位は 10 ミリ秒 (ms) です。

## A.3.8.2 WDU_HaltTransfer()

### 目的

- 指定されたパイプの転送を停止します (WinDriver では、1 つのパイプにつき、同時に 1 つの転送のみ許可されています)。

### プロトタイプ

```
DWORD WDU_HaltTransfer(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwPipeNum | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子 |
| dwPipeNum | パイプの番号 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.8.3 WDU_TransferDefaultPipe()

### 目的

- デフォルトのパイプを使用して、デバイスへ またはデバイスからデータを転送します。

### プロトタイプ

```
DWORD WDU_TransferDefaultPipe(
    WDU_DEVICE_HANDLE hDevice,
    DWORD fRead,
    DWORD dwOptions,
    PVOID pBuffer,
    DWORD dwBufferSize,
    PDWORD pdwBytesTransferred,
    PBYTE pSetupPacket,
    DWORD dwTimeout);
```

### パラメータ

`WDU_Transfer()` [A.3.8.1] のパラメータを参照してください。
`dwPipeNum` はこの関数のパラメータではありません。

### 説明

`WDU_Transfer()` [A.3.8.1] の説明を参照してください

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.8.4 WDU_TransferBulk()

### 目的

- デバイスへまたはデバイスからバルク データ転送を実行します。

### プロトタイプ

```
DWORD WDU_TransferBulk(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum,
    DWORD fRead,
    DWORD dwOptions,
    PVOID pBuffer,
    DWORD dwBufferSize,
    PDWORD pdwBytesTransferred,
    DWORD dwTimeout);
```

### パラメータ

`WDU_Transfer()` [A.3.8.1] のパラメータを参照してください。
`pSetupPacket` はこの関数のパラメータではありません。

### 説明

`WDU_Transfer()` [A.3.8.1] の説明を参照してください。

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.8.5 WDU_TransferIsoch()

### 目的

- デバイスへまたはデバイスからアイソクロナス データ転送を実行します。

### プロトタイプ

```
DWORD WDU_TransferIsoch(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum,
    DWORD fRead,
    DWORD dwOptions,
    PVOID pBuffer,
    DWORD dwBufferSize,
    PDWORD pdwBytesTransferred,
    DWORD dwTimeout);
```

### パラメータ

`WDU_Transfer()` [A.3.8.1] のパラメータを参照してください。
`pSetupPacket` はこの関数のパラメータではありません。

### 説明

`WDU_Transfer()` [A.3.8.1] の説明を参照してください。

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

#### A.3.8.6 WDU_TransferInterrupt()

### 目的

- デバイスへまたはデバイスからインタラプト データ転送を実行します。

### プロトタイプ

```
DWORD WDU_TransferInterrupt(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum,
    DWORD fRead,
    DWORD dwOptions,
    PVOID pBuffer,
    DWORD dwBufferSize,
    PDWORD pdwBytesTransferred,
    DWORD dwTimeout);
```

### パラメータ

`WDU_Transfer()` [A.3.8.1] のパラメータを参照してください。
`pSetupPacket` はこの関数のパラメータではありません。

### 説明

`WDU_Transfer()` [A.3.8.1] の説明を参照してください。

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.9 ストリーミング データ転送関数

このセクションでは、WinDriver のストリーミング データ転送関数について説明します。

ストリーム転送および Windriver を使用した実装に関する詳細は、WinDriver ユーザーズ ガイドのセクション 9.5 を参照してください。このセクションで説明する API は、現在 Windows と Windows CE でのみサポートしています。

#### A.3.9.1 WDU_StreamOpen()

### 目的

- 指定されたパイプの新しいストリームを開きます。

- コントロール パイプ (Pipe 0) を除く、すべてのパイプにストリームを関連付けることができます。
- ストリームのデータ転送方向 (読み取り / 書き込み) は、パイプの方向により決定されます。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamOpen(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum,
    DWORD dwBufferSize,
    DWORD dwRxSize,
    BOOL  fBlocking,
    DWORD dwOptions,
    DWORD dwRxTxTimeout,
    WDU_STREAM_HANDLE *phStream);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwPipeNum | DWORD | 入力 |
| ➢ dwBufferSize | DWORD | 入力 |
| ➢ dwRxSize | DWORD | 入力 |
| ➢ fBlocking | BOOL | 入力 |
| ➢ dwOptions | DWORD | 入力 |
| ➢ dwRxTxTimeout | DWORD | 入力 |
| ➢ phStream | WDU_STREAM_HANDLE* | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| dwPipeNum | ストリームを開くパイプの番号 |
| dwBufferSize | ストリーム データ バッファのバイト単位でのサイズ |
| dwRxSize | ストリームによりデバイスから読み取るデータ ブロックのバイト単位でのサイズ。このパラメータは、読み取りストリームに対してのみ使用され、`dwBufferSize` パラメータ の値以下でなければなりません。<br>注意: **`USB_STREAM_MAX_TRANSFER_SIZE_OVERWRITE`** **`dwOptions`** フラグを設定しても、最大転送サイズになります。 |
| fBlocking | ブロッキング I/O を使用するブロッキング ストリームの場合は TRUE。ノンブロッキング I/O を使用するノンブロッキング ストリームの場合は FALSE。詳細は、WinDriver ユーザーズ ガイドのセクション 9.5.3.1 を参照してください。 |
| dwOptions | 以下のいずれかのフラグのビット マスク。 |

| | |
|---|---|
| | **USB_ISOCH_NOASAP** - アイソクロナス データ転送。このオプションを設定することで、データ転送の実行中に下位 USB スタック ドライバ (**usbd.sys**) は (次に利用可能なフレームの代わりに) 現在のフレーム番号を使用するよう命令します。低スピードまたはフル スピードの USB 1.1 デバイスで転送中に未使用のフレームがある場合、このフラグを使用します。このフラグは Windows でのみ利用可能で、Windows CE では無視されます。<br>**USB_ISOCH_FULL_PACKETS_ONLY** - パケット サイズ以下のデータをアイソクロナス パイプに転送しません。<br>**USB_BULK_INT_URB_SIZE_OVERRIDE_128K** - URB (USB Request Block) のサイズを 128KB に制限します。このフラグは Windows でのみ利用可能です。<br>**USB_STREAM_OVERWRITE_BUFFER_WHEN_FULL** – 転送を完了するのに読み取りストリームのデータ バッファに十分な空き容量がない場合、バッファの古いデータを上書きします。このフラグは、読み取りストリームにのみ利用可能です。<br>**USB_STREAM_MAX_TRANSFER_SIZE_OVERRIDE** – Windows CE で、デフォルトの最大転送サイズを dwRxSize 転送サイズで上書きします。注意: このフラグを使用すると、大きい dwRxSize の値を設定するので、ホスト コントローラの制限によって、転送が失敗する場合があります。このフラグは、Windows CE で読み取りストリームにのみ利用可能です。 |
| dwRxTxTimeout | ストリームとデバイス間のデータ転送のミリ秒 (ms) 単位での最大所要時間。0 の場合、タイムアウトせずに、無限に待ちます。 |
| phStream | ストリームのユニークな識別子へのポインタ。この関数により返され、その他の `WDU_StreamXXX()` 関数に渡されます。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### A.3.9.2 WDU_StreamStart()

### 目的

- ストリーム (例: ストリームとデバイス間の転送) を開始します。
- データは、ストリームの方向 (読み取り / 書き込み) に転送されます。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamStart(
    WDU_STREAM_HANDLE hStream);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | WDU_StreamOpen() によって返されるストリームのユニークな識別子 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.9.3 WDU_StreamRead()

### 目的

- 読み取りストリームからアプリケーションへデータを読み取ります。
- ブロッキング ストリームの場合 (fBlocking=TRUE - WDU_StreamOpen() を参照)、指定されたデータ量 (バイト) を読み取るか、ストリームによるデバイスからの読み取りがタイムアウト (例: WDU_StreamOpen() の dwRxTxTimeout パラメータ [A.3.9.1] で設定されたストリームとデバイス間の転送のタイムアウトに到達した場合) になるまでこの関数の呼び出しはブロックされます。
- ノンブロッキング ストリームの場合 (fBlocking=FALSE)、この関数は要求されたデータをできるだけ多く (ストリーム データ バッファにある利用可能なデータ量により異なる) アプリケーションに転送して、直ちにリターンします。
- ブロッキング転送およびノンブロッキング転送の両方の場合で、この関数はストリームから実際に読み取ったバイト数を、pdwBytesRead パラメータに格納して返します。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamRead(
    HANDLE hStream,
    PVOID pBuffer,
    DWORD bytes,
    DWORD *pdwBytesRead);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |
| ➢ pBuffer | PVOID | 出力 |
| ➢ bytes | DWORD | 入力 |
| ➢ pdwBytesRead | DWORD* | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | WDU_StreamOpen() によって返されるストリームのユニークな識 |

| | |
|---|---|
| | 別子 |
| pBuffer | ストリームから読み取るデータを保持するためのデータ バッファへのポインタ |
| bytes | ストリームから読み取るバイト数 |
| pdwBytesRead | ストリームから実際に読み取ったバイト数へのポインタ |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### A.3.9.4 WDU_StreamWrite()

### 目的

- アプリケーションから書き込みストリームへデータを書き込みます。
- ブロッキング ストリームの場合 (`fBlocking=TRUE` - `WDU_StreamOpen()` を参照)、すべてのデータを書き込むか、ストリームによるデバイスへの書き込みがタイムアウト (例: `WDU_StreamOpen()` の `dwRxTxTimeout` パラメータ [A.3.9.1] で設定されたストリームとデバイス間の転送のタイムアウトに到達した場合) になるまでこの関数の呼び出しはブロックされます。
- ノンブロッキング ストリームの場合 (`fBlocking=FALSE`)、この関数はできるだけ多くのデータをストリーム データ バッファに書き込み、直ちにリターンします。
- ブロッキング転送およびノンブロッキング転送の両方の場合で、この関数はストリームへ実際に書き込んだバイト数を、`pdwBytesWritten` パラメータに格納して返します。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamWrite(
    HANDLE hStream,
    const PVOID pBuffer,
    DWORD bytes,
    DWORD *pdwBytesWritten);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |
| ➢ pBuffer | PVOID | 入力 |
| ➢ bytes | DWORD | 入力 |
| ➢ pdwBytesWritten | DWORD* | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | `WDU_StreamOpen()` によって返されるストリームのユニークな識 |

| | |
|---|---|
| | 別子 |
| pBuffer | ストリームへ書き込むデータを保持するデータ バッファへのポインタ |
| bytes | ストリームへ書き込むバイト数 |
| pdwBytesWritten | ストリームへ実際に書き込んだバイト数へのポインタ |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### A.3.9.5 WDU_StreamFlush()

### 目的

- 書き込みストリームをフラッシュします (例: ストリーム データ バッファのすべてのコンテンツをデバイスに書き込みます)。
- ストリームのすべての待機中 I/O が処理されるまでブロックします。
- この関数は、ブロッキング ストリームとノンブロッキング ストリームの両方で呼び出すことができます。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamFlush(
    WDU_STREAM_HANDLE hStream);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | `WDU_StreamOpen()` によって返されるストリームのユニークな識別子 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### A.3.9.6 WDU_StreamGetStatus()

### 目的

- ストリームの現在の状況を返します。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamGetStatus(
    WDU_STREAM_HANDLE hStream,
    BOOL *pfIsRunning,
    DWORD *pdwLastError,
    DWORD *pdwBytesInBuffer);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |
| ➢ pfIsRunning | BOOL* | 出力 |
| ➢ pdwLastError | DWORD* | 出力 |
| ➢ pdwBytesInBuffer | DWORD* | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | WDU_StreamOpen() によって返されるストリームのユニークな識別子 |
| pfIsRunning | ストリームの現在の状況を示す値へのポインタ。<br>• TRUE – ストリームが現在起動中です。<br>• FALSE – ストリームが現在停止しています。 |
| pdwLastError | ストリームに関連した最後のエラーへのポインタ。<br>注意: この関数の呼び出すことで、ストリームの最後のエラーをリセットします。 |
| pdwBytesInBuffer | ストリームのデータ バッファの現在のバイト カウントへのポインタ。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.9.7 WDU_StreamStop()

### 目的

- アクティブなストリーム (例: ストリームとデバイス間の転送) を停止します。
- 書き込みストリームの場合、この関数はストリームを停止する前にフラッシュ (ストリームのコンテンツをデバイスに書き込むなど) します。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamStop(
    WDU_STREAM_HANDLE hStream);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | `WDU_StreamOpen()` によって返されるストリームのユニークな識別子 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.9.8 WDU_StreamClose()

### 目的

- 開かれているストリームを閉じます。
- この関数はストリームを閉じる前に、ストリームを停止し、データをデバイスにフラッシュします (書き込みストリームの場合)。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_StreamClose(
    WDU_STREAM_HANDLE hStream);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hStream | WDU_STREAM_HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hStream | `WDU_StreamOpen()` によって返されるストリームのユニークな識別子 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.10 WDU_ResetPipe()

### 目的

- パイプのホスト側の停止状態とエンドポイントのストール状態の両方をクリアして、パイプをリセットします。
- pipe00 を除くすべてのパイプで有効です。

### プロトタイプ

```
DWORD WDU_ResetPipe(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwPipeNum);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwPipeNum | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子 |
| dwPipeNum | パイプの番号 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- この関数は、パイプの停止状態をクリアするために使用します。

## A.3.11 WDU_ResetDevice()

### 目的

- デバイスをリセットします。

### プロトタイプ

```
DWORD WDU_ResetDevice(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwOptions);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwOptions | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子 |
| dwOptions | 0 または:<br>• **WD_USB_HARD_RESET** -デバイスが無効に設定されていない場合でもリセットします。このオプションを使用した後に、WDU_SetInterface() [A.3.2] を使用して、インターフェイス デバイスを設定することを推奨します。<br>• **WD_USB_CYCLE_PORT** - デバイスをリセットせずに再度列挙するようにオペレーティング システムに求め、デバイスの取り外しおよび再装着のシミュレーションを行います。このオプションは、Windows XP 以降でのみサポートしています。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- WDU_ResetDevice() は Windows および Windows CE 5.0 以降でサポートしています。WD_USB_CYCLE_PORT オプションは、Windows XP 以降でサポートしています。
- この関数は、Windows USB ドライバから、ハブ ポートのリセット要求を発行します (Windows USB ドライバでこの機能がサポートされていることを前提としています)。

## A.3.12 WDU_SelectiveSuspend()

### 目的

- 指定されたデバイスのサスペンド要求を送信 (選択的サスペンド)、または送信済みサスペンド要求をキャンセルします。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_SelectiveSuspend(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwOptions);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwOptions | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子 |
| dwOptions | 以下のいずれかの WDU_SELECTIVE_SUSPEND_OPTIONS の値を指定できます。<br>**WDU_SELECTIVE_SUSPEND_SUBMIT** - デバイスのサスペンド要求を送信します。<br>**WDU_SELECTIVE_SUSPEND_CANCEL** - 送信済みのデバイスのサスペンド要求をキャンセルします。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

サスペンド要求を送信した際にデバイスがビジーな場合 (dwOptions = WDU_SELECTIVE_SUSPEND_SUBMIT)、WD_OPERATION_FAILED を返します。

### 注釈

- WDU_SelectiveSuspend() は、Windows XP 以降でサポートしています。

## A.3.13 WDU_Wakeup()

### 目的

- ウェイクアップ機能を有効 / 無効にします。

### プロトタイプ

```
DWORD WDU_Wakeup(
    WDU_DEVICE_HANDLE hDevice,
    DWORD dwOptions);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ dwOptions | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子。 |
| dwOptions | 以下のいずれか:<br>• **WDU_WAKEUP_ENABLE** - ウェイクアップ機能を有効にします。<br>または<br>• **WDU_WAKEUP_DISABLE** - ウェイクアップ機能を無効にします。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

## A.3.14 WDU_GetLangIDs()

### 目的

- デバイスからサポートされている言語 ID のリストと数を読み取ります。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_GetLangIDs(
    WDU_DEVICE_HANDLE hDevice,
    PBYTE pbNumSupportedLangIDs,
    WDU_LANGID *pLangIDs,
    BYTE bNumLangIDs);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ pbNumSupportedLangIDs | PBYTE | 出力 |
| ➢ pLangIDs | WDU_LANGID* | 出力 |
| ➢ bNumLangIDs | BYTE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイス用のユニークな識別子 |
| pbNumSupportedLangIDs | サポートされている言語 ID の数を受け取るパラメータ |
| pLangIDs | 言語 ID の配列。**bNumLangIDs** が 0 でない場合、デバイスでサポートされている言語 ID が格納されます。 |

| | |
|---|---|
| bNumLangIDs | pLangIDs 配列に格納されている ID の数 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- **dwNumLangIDs** が 0 の場合、この関数はサポートされている言語 ID の数 (**pbNumSupportedLangIDs**)のみを返し、実際の ID で **pLangIDs** 配列を更新しません。このため、**pLangIDs** は (参照されたないため) NULL にすることができまが、**pbNumSupportedLangIDs** は NULL にしないでください。ただし、ユーザーがサポートされている言語 ID の数ではなく、そのリストだけを必要とする場合、**pbNumSupportedLangIDs** を NULL にすることができます。この場合、**bNumLangIDs** を 0 にしたり、**pLangIDs** を NULL にすることはできません。
- デバイスがどの言語 ID もサポートしていない場合、この関数は正常終了します。このため、この関数がリターンした後に、呼び出し元で ***pbNumSupportedLangIDs** の値を確認する必要があります。
- **pLangIDs** 配列のサイズ (**bNumLangIDs**) がデバイスでサポートされている ID の数 (***pbNumSupportedLangIDs**) よりも小さい場合、この関数はサポートされている言語 ID の中から最初の **bNumLangIDs** のみ読み取って返します。

## A.3.15 WDU_GetStringDesc()

### 目的

- 文字列インデックスによりデバイスから文字列ディスクリプタを読み取ります。

### プロトタイプ

```
DWORD DLLCALLCONV WDU_GetStringDesc(
    WDU_DEVICE_HANDLE hDevice,
    BYTE bStrIndex,
    PBYTE pbBuf,
    DWORD dwBufSize,
    WDU_LANGID langID,
    PDWORD pdwDescSize);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hDevice | WDU_DEVICE_HANDLE | 入力 |
| ➢ bStrIndex | BYTE | 入力 |
| ➢ pbBuf | PBYTE | 出力 |
| ➢ dwBufSize | DWORD | 入力 |
| ➢ langID | WDU_LANGID | 入力 |

| | | |
|---|---|---|
| ➢ pdwDescSize | PDWORD | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hDevice | デバイス / インターフェイスのユニークな識別子 |
| bStrIndex | 読み取り文字列ディスクリプタのインデックス |
| pbBuf | 文字列ディスクリプタで満たされるバッファへのポインタ |
| dwBufSize | **pbBuf** のバイト サイズ |
| langID | 文字列ディスクリプタの取得 (*get string descriptor*) の要求で使用する言語 ID。このパラメータが 0 の場合、この関数はデバイスから返されるサポートされている言語 ID の中から最初のものを使用します。 |
| pdwDescSize | デバイスから読み取った文字列ディスクリプタのサイズで満たされたオプションの DWORD ポインタ。NULL の場合、文字列ディスクリプタのサイズは返されません。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- **pbBuf** バッファのサイズが文字列ディスクリプタを保持するのに十分でない場合 (**dwBufSize** < ***pdwDescSize**)、デバイスから返されたディスクリプタは **dwBufSize** のサイズに切り捨てられます。

# A.4 USB データ型

このセクションで説明する型 (タイプ) は、ドキュメント内で特別な指定がない限り、
**WinDriver/include/windrvr.h** ヘッダー ファイル内で宣言されています。

## A.4.1 WD_DEVICE_REGISTRY_PROPERTY 列挙型

デバイスのレジストリ プロパティの識別子の列挙型。
NULL 終端 WCHAR 配列フォーマットで文字列プロパティを返します。

注意: この列挙型で説明するプロパティに関する詳細情報は、MSDN (Microsoft Development Network) のドキュメントの Windows の IoGetDeviceProperty() 関数の DeviceProperty パラメータの説明を参照してください。

| 列挙値 | 説明 |
|---|---|
| WdDevicePropertyDeviceDescription | デバイスの説明 |
| WdDevicePropertyHardwareID | デバイスのハードウェア ID |

| | |
|---|---|
| WdDevicePropertyCompatibleIDs | デバイスの互換 ID |
| WdDevicePropertyBootConfiguration | ファームウェアによってデバイスへ割り当てられたハードウェアのリソース、生データ フォーム |
| WdDevicePropertyBootConfigurationTranslated | ファームウェアによってデバイスへ割り当てられたハードウェアのリソース、変換済みのフォーム |
| WdDevicePropertyClassName | デバイスのセットアップ クラスの名前、テキスト フォーマット |
| WdDevicePropertyClassGuid | デバイスのセットアップ クラスの GUID (文字列フォーマット) |
| WdDevicePropertyDriverKeyName | ドライバ独自のレジストリ キーの名前 |
| WdDevicePropertyManufacturer | デバイス製造元 (マニュファクチャ) の文字列 |
| WdDevicePropertyFriendlyName | 使いやすいデバイスの名前 (基本的にはクラス インストーラで定義)、2 つの同様のデバイスを区別するのに使用 |
| WdDevicePropertyLocationInformation | バス上のデバイスの場所に関する情報 (文字列フォーマット)<br>この情報の解釈は、バス独自です。 |
| WdDevicePropertyPhysicalDeviceObjectName | デバイスの PDO (Physical Device Object) の名前 |
| WdDevicePropertyBusTypeGuid | デバイスを接続するバスの GUID |
| WdDevicePropertyLegacyBusType | バス タイプ (PCIBus または PCMCIABus) |
| WdDevicePropertyBusNumber | デバイスを接続するバスのレガシー バス番号 |
| WdDevicePropertyEnumeratorName | デバイスの列挙型の名前 (PCI またはルート) |
| WdDevicePropertyAddress | デバイスのバス アドレス<br>このアドレスの解釈は、バス独自です |
| WdDevicePropertyUINumber | ユーザー インターフェイスで表示できるデバイスに関連する番号 |
| WdDevicePropertyInstallState | デバイスのインストール状況 |
| WdDevicePropertyRemovalPolicy | デバイスの現在の取り外しポリシー (Windows XP およびそれ以降) |

# A.5 USB – 構造

次の図は、WinDriver の USB API が使用する構造階層を表しています。階層の各レベルに位置する配列は、図で表されている以上に要素を持っています。矢印はポインタを表しています。分かり易くするために、階層の各レベルの 1 つの構造のみを、詳細に説明しています (リストされたすべての要素とポインタ)。

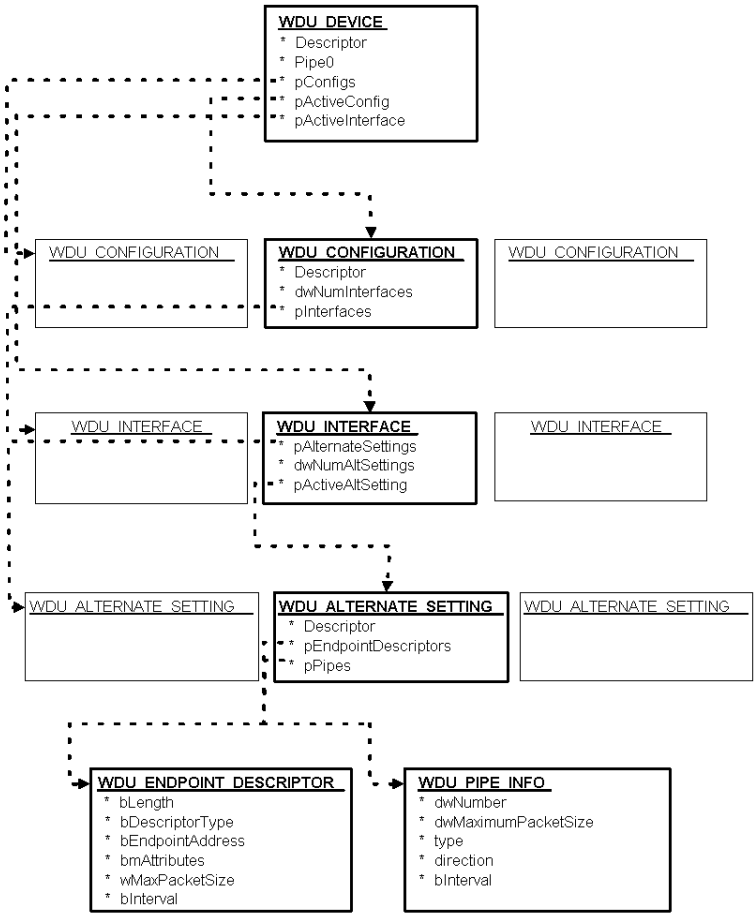



**図 A.2: WinDriver USB の構造**

## A.5.1 WDU_MATCH_TABLE 構造体

USB 一致テーブルは、以下の要素で構成されています。

**注意**

すべての項目の (*) は、値を 0 に設定すると、すべて一致します。

| 名前 | 型 | 説明 |
|---|---|---|
| wVendorId | WORD | USB-IF に割り当てられる、検出に必要な USB ベンダー ID。(*) |
| wProductId | WORD | 製造元に割り当てられる、検出に必要な USB プロダクト ID。(*) |
| bDeviceClass | BYTE | USB-IF に割り当てられる、デバイスのクラス コード。(*) |
| bDeviceSubClass | BYTE | USB-IF に割り当てられる、デバイスのサブ クラス コード。(*) |
| bInterfaceClass | BYTE | USB-IF に割り当てられる、インターフェイスのクラス コード。(*) |
| bInterfaceSubClass | BYTE | USB-IF に割り当てられる、インターフェイスのサブ クラス コード。(*) |
| bInterfaceProtocol | BYTE | USB-IF に割り当てられる、インターフェイスのプロトコル コード。(*) |

## A.5.2 WDU_EVENT_TABLE 構造体

USB イベント テーブルは、以下の要素で構成されています。
この構造体は、**WinDriver/include/wdu_lib.h** ヘッダー ファイルに定義されています。

| 名前 | 型 | 説明 |
|---|---|---|
| pfDeviceAttach | WDU_ATTACH_CALLBACK | デバイスを装着したときに WinDriver に呼ばれます。 |
| pfDeviceDetach | WDU_DETACH_CALLBACK | デバイスを取り外したときに WinDriver に呼ばれます。 |
| pfPowerChange | WDU_POWER_CHANGE_CALLBACK | デバイスの電源状態が変化すると WinDriver から呼ばれます。 |
| pUserData | PVOID | コールバックへ渡されるユーザーモード データへのポインタ。 |

## A.5.3 WDU_DEVICE 構造体

USB デバイス情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| Descriptor | WDU_DEVICE_DESCRIPTOR | デバイス ディスクリプタ情報の構造体 [A.4.7]。 |
| Pipe0 | WDU_PIPE_INFO | デバイスのデフォルトのパイプ (Pipe 0) に関する情報の構造体 [A.4.11] 。 |
| pConfigs | WDU_CONFIGURATION* | デバイスの設定情報の構造体 [A.4.4] へのポインタ。 |
| pActiveConfig | WDU_CONFIGURATION* | デバイスのアクティブな設定に関する情報の構造体 [A.4.4] へのポインタ。 |
| pActiveInterface | WDU_INTERFACE* | デバイスのアクティブなインターフェイスに関する情報の構造体 [A.4.5] へのポインタの配列。 |

## A.5.4 WDU_CONFIGURATION 構造体

設定情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| Descriptor | WDU_CONFIGURATION_DESCRIPTOR | 設定ディスクリプタ情報の構造体 [A.4.8] 。 |
| dwNumInterfaces | DWORD | この設定でサポートされるインターフェイスの数。 |
| pInterfaces | WDU_INTERFACE* | 設定のインターフェイスに関する情報の構造体 [A.4.5] 配列の初めへのポインタ。 |

## A.5.5 WDU_INTERFACE 構造体

インターフェイス情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| pAlternateSettings | WDU_ALTERNATE_SETTING* | インターフェイスの代替設定に関する情報の構造体 [A.4.6] 配列の初めへのポインタ。 |
| dwNumAltSettings | DWORD | このインターフェイスでサポートされている代替設定の数。 |
| pActiveAltSetting | WDU_ALTERNATE_SETTING* | インターフェイスのアクティブな代替設定に関する情報の構造体 [A.4.6] へのポインタ。 |

## A.5.6 WDU_ALTERNATE_SETTING 構造体

代替設定情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| Descriptor | WDU_INTERFACE_DESCRIPTOR | インターフェイス ディスクリプタ情報の構造体 [A.4.9]。 |
| pEndpointDescriptors | WDU_ENDPOINT_DESCRIPTOR* | 代替設定のエンドポイントディスクリプタ情報の構造体 [A.4.10] 配列の初めへのポインタ |
| pPipes | WDU_PIPE_INFO* | 代替設定のパイプ情報の構造体 [A.4.11] 配列の初めへのポインタ |

## A.5.7 WDU_DEVICE_DESCRIPTOR 構造体

USB デバイス ディスクリプタ情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| bLength | UCHAR | ディスクリプタのバイト サイズ (18 バイト)。 |
| bDescriptorType | UCHAR | デバイス ディスクリプタ (0x01)。 |
| bcdUSB | USHORT | デバイスが対応する USB 仕様の数。 |
| bDeviceClass | UCHAR | デバイスのクラス。 |
| bDeviceSubClass | UCHAR | デバイスのサブ クラス。 |
| bDeviceProtocol | UCHAR | デバイスのプロトコル。 |
| bMaxPacketSize0 | UCHAR | 転送されるパケットの最大サイズ。 |
| idVendor | USHORT | USB-IF に割り当てられるベンダー ID。 |
| idProduct | USHORT | 製造元に割り当てられるプロダクト ID。 |
| bcdDevice | USHORT | デバイス リリース番号。 |
| iManufacturer | UCHAR | 製造元文字列ディスクリプタのインデックス。 |
| iProduct | UCHAR | 製品文字列ディスクリプタのインデックス。 |

| | | |
|---|---|---|
| iSerialNumber | UCHAR | シリアル番号文字列ディスクリプタのインデックス。 |
| bNumConfigurations | UCHAR | 設定可能な数。 |

## A.5.8 WDU_CONFIGURATION_DESCRIPTOR 構造体

USB 設定ディスクリプタ情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| bLength | UCHAR | ディスクリプタのバイト サイズ。 |
| bDescriptorType | UCHAR | デバイス ディスクリプタ (0x02)。 |
| wTotalLength | USHORT | 必要なデータの合計バイト長。 |
| bNumInterfaces | UCHAR | インターフェイスの数。 |
| bConfigurationValue | UCHAR | 設定番号。 |
| iConfiguration | UCHAR | この設定を記述する文字列ディスクリプタのインデックス。 |
| bmAttributes | UCHAR | この設定の電源設定:<br>• *D6* - 電源内臓の場合<br>• *D5* - リモート ウェイクアップの場合 (デバイスはホストをウェイク アップできます)。 |
| MaxPower | UCHAR | 2*mA* ユニットで、この設定の最大電力消費量。 |

## A.5.9 WDU_INTERFACE_DESCRIPTOR 構造体

USB インターフェイス ディスクリプタ情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| bLength | UCHAR | ディスクリプタのバイト サイズ (9 バイト)。 |
| bDescriptorType | UCHAR | デバイス ディスクリプタ (0x04)。 |
| bInterfaceNumber | UCHAR | インターフェイス番号。 |
| bAlternateSetting | UCHAR | 代替設定番号。 |
| bNumEndpoints | UCHAR | このインターフェイスで使用するエンドポイントの数。 |
| bInterfaceClass | UCHAR | USB-IF に割り当てられるインターフェイスのクラス コード。 |
| bInterfaceSubClass | UCHAR | USB-IF に割り当てられるインターフェイスのサブ クラス コード。 |
| bInterfaceProtocol | UCHAR | USB-IF に割り当てられるインターフェイスのプロトコル コード。 |
| iInterface | UCHAR | このインターフェイスを記述する文字列ディスクリプタのインデックス。 |

## A.5.10 WDU_ENDPOINT_DESCRIPTOR 構造体

USB エンドポイント ディスクリプタ情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| bLength | UCHAR | ディスクリプタのバイト サイズ (7 バイト)。 |
| bDescriptorType | UCHAR | エンドポイント ディスクリプタ (0x05)。 |
| bEndpointAddress | UCHAR | エンドポイント アドレス: エンドポイント番号のビット 0-3 を使用します。ビット 4-6 を 0 に設定します。アウトバウンド データの 0 およびインバウンド データの 1 にビット 7 を設定します (コントロール エンドポイントのために無視されます)。 |
| bmAttributes | UCHAR | このエンドポイントの転送タイプを指定します (コントロール、インタラプト、アイソクロナスまたはバルク)。詳細は、USB の仕様を参照してください。 |
| wMaxPacketSize | USHORT | このエンドポイントが送受信可能なパケットの最大サイズ。 |
| bInterval | UCHAR | フレーム カウントのエンドポイント データ転送のポーリング間隔。<br>バルクおよびコントロール エンドポイントのため無視されます。<br>アイソクロナス エンドポイントの 1 に等しいです。<br>インタラプト エンドポイントの 1 から 255 の範囲です。 |

### A.5.11 WDU_PIPE_INFO 構造体

USB パイプ情報の構造体は、以下の要素で構成されています。

| 名前 | 型 | 説明 |
|---|---|---|
| dwNumber | DWORD | パイプ番号; デフォルトのパイプ番号は 0 です。 |
| dwMaximumPacketSize | DWORD | このパイプを使用して転送できるパケットの最大サイズ。 |
| type | DWORD | このパイプの転送タイプ。 |
| direction | DWORD | 転送の方法:<br>• アイソクロナス、バルクまたはインタラプト パイプの場合、**USB_DIR_IN** または **USB_DIR_OUT**。<br>• コントロール パイプの場合、**USB_DIR_IN_OUT**。 |
| dwInterval | DWORD | ミリ秒 (*ms*) 間隔。<br>インタラプト パイプにのみ適応されます。 |

## A.6 一般的な WD_xxx 関数

### A.6.1 WinDriver のコール順序 – 一般用法

次に WinDriver API の一般的なコール順序を示します。

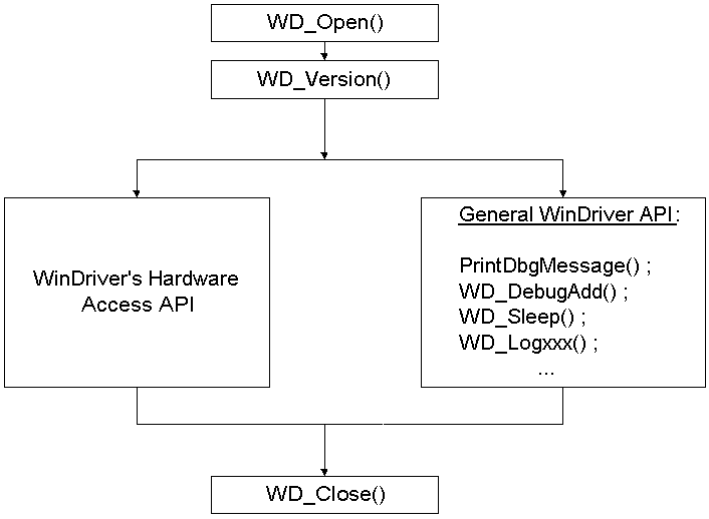



**図 A.3: WinDriver API の呼び出し順序**

**注意**

- `WD_Version()` [A.5.3] は、`WD_Open()` [A.5.2] をコールした後で他の WinDriver 関数をコールする前にコールすることを推奨します。その目的は、WinDriver カーネル モジュール (`windrvr6.sys/.dll/.o/.ko`) バージョン番号を返すことでアプリケーションおよび WinDriver カーネル モジュールのバージョンが互換であることを認識させます。
- `WD_Open()` のあと、`WD_DebugAdd()` [A.5.6] および `WD_Sleep()` [A.5.8] をどこからでもコールすることができます。

## A.6.2 WD_Open()

### 目的

- WinDriver カーネル モジュールにアクセスするためにハンドルをオープンします。ハンドルはすべての WinDriver API によって使用されるため、他の WinDriver API がコールされる前にハンドルをコールしなければなりません。

### プロトタイプ

```
HANDLE WD_Open(void);
```

### 戻り値

WinDriver カーネル モジュールへのハンドル。
デバイスをオープンできない場合は `INVALID_HANDLE_VALUE` を返します。

### 注釈

- 登録版を使用する場合、WinDriver のライセンスの登録方法に関しては、`WD_License()` [A.5.9] 関数の説明を参照してください。

### 例

```
HANDLE hWD;

hWD = WD_Open();
if (hWD == INVALID_HANDLE_VALUE)
{
    printf("Cannot open WinDriver device\n");
}
```

## A.6.3 WD_Version()

### 目的

- 起動中の WinDriver カーネル モジュールのバージョン番号を返します。

### プロトタイプ

```
DWORD WD_Version(
    HANDLE hWD,
    WD_VERSION *pVer);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |
| ➢ pVer | WD_VERSION* | |
| ☐ dwVer | DWORD | 出力 |
| ☐ cVer | CHAR[128] | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | WD_Open() [A.5.2] から返される WinDriver のカーネル モード ドライバへのハンドル。 |
| pVer | WinDriver バージョン情報の構造体へのポインタ。 |
| dwVer | バージョン番号。 |
| cVer | バージョン情報文字列。<br>バージョン文字列のサイズは、128 文字までに制限されています (NULL 終端文字を含む)。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 例

```
WD_VERSION ver;

BZERO(ver);
WD_Version(hWD, &ver);
printf("%s\n", ver.cVer);
if (ver.dwVer < WD_VER)
{
    printf("Error - incorrect WinDriver version\n");
}
```

## A.6.4 WD_Close()

### 目的

- WinDriver カーネル モジュールへのアクセスを終了します。

### プロトタイプ

```
void WD_Close(HANDLE hWD);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | WD_Open() [A.5.2] から返される WinDriver のカーネル モード ドライバへのハンドル。 |

### 戻り値

なし。

### 注釈

- WinDriver カーネル モジュールの使用を終了したときは、この関数を必ずコールします。

### 例

```
WD_Close(hWD);
```

## A.6.5 WD_Debug()

### 目的

- デバッグメッセージを収集するレベルにデバッグレベルを設定 します。

### プロトタイプ

```
DWORD WD_Debug(
    HANDLE hWD,
    WD_DEBUG *pDebug);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |
| ➢ pDebug | WD_DEBUG* | 入力 |
| ☐ dwCmd | DWORD | 入力 |
| ☐ dwLevel | DWORD | 入力 |
| ☐ dwSection | DWORD | 入力 |
| ☐ dwLevelMessageBox | DWORD | 入力 |
| ☐ dwBufferSize | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | WD_Open() [A.5.2] から返される WinDriver のカーネル モード ドライバへのハンドル。 |
| pDebug | デバッグ情報の構造体へのポインタ。 |
| dwCmd | デバッグ コマンド: フィルタの設定、バッファのクリア等。<br>詳細は **windrvr.h** の DEBUG_COMMAND を参照してください。 |
| dwLevel | dwCmd=DEBUG_SET_FILTER に使用されます。デバッグ レベルを Error、Warning、Info、Trace に設定します。<br>詳細は **windrvr.h** の DEBUG_LEVEL を参照してください。 |
| dwSection | dwCmd=DEBUG_SET_FILTER に使用されます。セクションを I/O、Memory、Interrupt などに設定し、収集します (すべての場合は S_ALL を使用します)。<br>詳細は **windrvr.h** の DEBUG_SECTION を参照してください。 |
| dwLevelMessageBox | dwCmd=DEBUG_SET_FILTER に使用されます。デバッグレベルをメッセージボックスに出力するように設定します。<br>詳細は **windrvr.h** の DEBUG_LEVEL を参照してください。 |
| dwBufferSize | dwCmd=DEBUG_SET_BUFFER に使用されます。カーネルにあるバッファのサイズです。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 例

```
WD_DEBUG dbg;

BZERO(dbg);
dbg.dwCmd = DEBUG_SET_FILTER;
dbg.dwLevel = D_ERROR;
dbg.dwSection = S_ALL;
dbg.dwLevelMessageBox = D_ERROR;

WD_Debug(hWD, &dbg);
```

## A.6.6 WD_DebugAdd()

### 目的

- デバッグ ログへデバッグ メッセージを送ります。ドライバ コードで使用します。

### プロトタイプ

```
DWORD WD_DebugAdd(
    HANDLE hWD,
    WD_DEBUG_ADD *pData);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |
| ➢ pData | WD_DEBUG_ADD* | |
| ☐ dwLevel | DWORD | 入力 |
| ☐ dwSection | DWORD | 入力 |
| ☐ pcBuffer | CHAR [256] | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | `WD_Open()` [A.5.2] から返される WinDriver のカーネル モード ドライバへのハンドル。 |
| pData | 追加するデバッグ情報の構造体へのポインタ。 |
| dwLevel | 宣言されるデータに、Debug Monitor でレベルを割り当てます。<br>`dwLevel` が 0 の場合、`D_ERROR` が宣言されます。<br>詳細は **windrvr.h** の `DEBUG_LEVEL` を参照してください。 |
| dwSection | 宣言されるデータに、Debug Monitor でセクションを割り当てます。<br>`dwSection` が 0 の場合、`S_MISC` が宣言されます。<br>詳細は **windrvr.h** の `DEBUG_SECTION` を参照してください。 |
| pcBuffer | メッセージ ログにコピーする文字列。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 例

```
WD_DEBUG_ADD add;

BZERO(add);
add.dwLevel = D_WARN;
add.dwSection = S_MISC;
sprintf(add.pcBuffer, "This message will be displayed in "
    "the Debug Monitor\n");
WD_DebugAdd(hWD, &add);
```

## A.6.7 WD_DebugDump()

### 目的

- デバッグ メッセージ バッファを取り出します。

### プロトタイプ

```
DWORD WD_DebugDump(
    HANDLE hWD,
    WD_DEBUG_DUMP *pDebugDump);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |
| ➢ pDebug | WD_DEBUG_DUMP* | 入力 |
| ☐ pcBuffer | PCHAR | 入力 / 出力 |
| ☐ dwSize | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | WD_Open() [A.5.2] から返される WinDriver のカーネル モード ドライバへのハンドル。 |
| pDebugDump | デバッグ ダンプ情報の構造体へのポインタ。 |
| pcBuffer | デバッグ メッセージを受け取るバッファ。 |
| dwSize | バッファ サイズ (Byte)。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 例

```
char buffer[1024];
WD_DEBUG_DUMP dump;
dump.pcBuffer=buffer;
dump.dwSize = sizeof(buffer);
WD_DebugDump(hWD, &dump);
```

## A.6.8 WD_Sleep()

### 目的

- 指定した時間分、実行を遅らせます。

### プロトタイプ

```
DWORD WD_Sleep(
    HANDLE hWD,
    WD_SLEEP *pSleep);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |
| ➢ pSleep | WD_SLEEP* | |
| ☐ dwMicroSeconds | DWORD | 入力 |
| ☐ dwOptions | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | WD_Open() [A.5.2] から返される WinDriver のカーネル モード ドライバへのハンドル。 |
| pSleep | スリープ情報の構造体へのポインタ。 |
| dwMicroSeconds | 実行を遅らせる時間 (マイクロ秒単位)。- 1/1,000,000 秒 |
| dwOptions | 以下のいずれかのフラグのビット マスク。<br>• **Zero (0)** - ビジー スリープ (デフォルト)<br>または、<br>• **SLEEP_NON_BUSY** - CPU リソースを消費せずに実行を遅らせます。(17,000 マイクロ秒以下では無関係です。ビジー スリープよりも精度が落ちます。) |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- 使用例: 応答の遅いハードウェアへのアクセス。

### 例

```
WD_Sleep slp;

BZERO(slp);
slp.dwMicroSeconds = 200;
WD_Sleep(hWD, &slp);
```

## A.6.9 WD_License()

### 目的

- ライセンス文字列を WinDriver カーネル モジュールに転送して指定したライセンス文字列のライセンスの種類に関する情報を返します。

注意: WDU USB APIs [A.1] を使用する場合、WinDriver ライセンスの登録は`WDU_Init()` [A.3.1] への呼び出しで実行されるため、コードから直接 `WD_License()` を呼び出す必要はありません。

### プロトタイプ

```
DWORD WD_License(
    HANDLE hWD,
    WD_LICENSE *pLicense);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hWD | HANDLE | 入力 |
| ➢ pLicense | WD_LICENSE* | |
| ☐ cLicense | CHAR[] | 入力 |
| ☐ dwLicense | DWORD | 出力 |
| ☐ dwLicense2 | DWORD | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| hWD | `WD_Open()` [A.5.2] から渡された WinDriver のカーネル モード ドライバへのハンドル。 |
| pLicense | WinDriver ライセンス情報の構造体へのポインタ。 |

| | |
|---|---|
| cLicense | WinDriver カーネル モジュールへ転送されるライセンス文字列を含むためのバッファ。空の文字列が転送された場合、WinDriver カーネル モジュールはパラメータ `dwLicense` にライセンスの種類を返します。 |
| dwLicense | ライセンス文字列 (`cLicnese`) が許可したライセンスの種類を返します。戻り値は、**windrvr.h** で enum として定義したライセンスのフラグのビット マスクです。0 は無効なライセンスを示します。必要な場合、ライセンスの種類を決定する追加のフラグは、**`dwLicense2`** に返されます。 |
| dwLicense2 | `dwLicense` が対応するすべての情報を持てない場合 (その他の場合は 0)、ライセンスの種類の決定に追加のフラグを返します。 |

### 戻り値

正常終了した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、エラーコードを返します [A.7]。

### 注釈

- 登録版を使用する際に、コードからライセンスを登録するには、`WD_Open()` [A.5.2] 以外の WinDriver API関数を呼ぶ前に、この関数を呼ぶ必要があります。

### 例

使用例: アプリケーションに登録用のルーチンを追加する。

```
DWORD RegisterWinDriver()
{
    HANDLE hWD;
    WD_LICENSE lic;
    DWORD dwStatus = WD_INVALID_HANDLE;

    hWD = WD_Open();
    if (hWD!=INVALID_HANDLE_VALUE)
    {
        BZERO(lic);
        /* Replace the following string with your license string: */
        strcpy(lic.cLicense, "12345abcde12345.CompanyName");
        dwStatus = WD_License(hWD, &lic);
        WD_Close(hWD);
    }

    return dwStatus;
}
```

# A.7 ユーザーモード ユーティリティ関数

このセクションでは、さまざまな作業を実装するのに役に立つユーザーモード ユーティリティ関数を説明します。これらのユーティリティ関数をマルチ プラットフォーム (WinDriver がサポートしているすべてのオペレーティング システム) で実装します。

## A.7.1 Stat2Str()

### 目的

- ステータス コードに対応するステータス文字列を取得します。

### プロトタイプ

```
const char *Stat2Str(DWORD dwStatus);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ dwStatus | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| dwStatus | ステータス コードの番号。 |

### 戻り値

指定したステータス コードの番号に対応するステータスの説明 (文字列) を返します。

### 注釈

ステータス コードと文字列の一覧は、セクション A.7 を参照してください。

## A.7.2 get_os_type()

### 目的

- オペレーティング システムの種類を取得します。

### プロトタイプ

```
OS_TYPE get_os_type(void);
```

### 戻り値

オペレーティング システムの種類を返します。
オペレーティング システムの種類を検出できなかった場合、OS_CAN_NOT_DETECT を返します。

## A.7.3 ThreadStart()

### 目的

- スレッドを作成します。

### プロトタイプ

```
DWORD ThreadStart(
    HANDLE *phThread,
    HANDLER_FUNC pFunc,
    void *pData);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ phThread | HANDLE* | 出力 |
| ➢ pFunc | Typedef void (*HANDLER_FUNC)(void *pData); | 入力 |
| ➢ pData | VOID* | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| phThread | 生成したスレッドへのハンドルを返します。 |
| pFunc | 新しいスレッドを実行する際のコードの開始アドレス。(このハンドラーのプロトタイプ (HANDLER_FUNC) は、**utils.h** で定義されています)。 |
| pData | 新しいスレッドへ渡されるデータへのポインタ。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.4 ThreadWait()

### 目的

- 終了するスレッドを待機します。

### プロトタイプ

```
void ThreadWait(HANDLE hThread);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hThread | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hThread | 終了するのを待たれているスレッドへのハンドル。 |

### 戻り値

なし。

## A.7.5 OsEventCreate()

### 目的

- イベント オブジェクトを生成します。

### プロトタイプ

```
DWORD OsEventCreate(HANDLE *phOsEvent);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ phOsEvent | HANDLE* | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| phOsEvent | 新しく生成したイベント オブジェクトへのハンドルを受信する変数へのポインタ。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します[A.7]。

## A.7.6 OsEventClose()

### 目的

- イベント オブジェクトへのハンドルを閉じます。

### プロトタイプ

```
void OsEventClose(HANDLE hOsEvent);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsEvent | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsEvent | 閉じられるイベント オブジェクトへのハンドル。 |

### 戻り値

なし。

## A.7.7 OsEventWait()

### 目的

- 指定したイベント オブジェクトがシグナル状態になるかまたはタイムアウトになるまで待機します。

### プロトタイプ

```
DWORD OsEventWait(
    HANDLE hOsEvent,
    DWORD dwSecTimeout);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsEvent | HANDLE | 入力 |
| ➢ dwSecTimeout | DWORD | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsEvent | イベント オブジェクトへのハンドル。 |
| dwSecTimeout | イベントのタイムアウト インターバル (秒単位)。<br>タイムアウトを INFINITE に設定すると、無限に待ちます。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.8 OsEventSignal()

### 目的

- 指定したイベント オブジェクトをシグナル状態に設定します。

### プロトタイプ

```
DWORD OsEventSignal(HANDLE hOsEvent);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsEvent | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsEvent | イベント オブジェクトへのハンドル。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.9 OsEventReset()

### 目的

- 指定したイベント オブジェクトを非シグナル状態に設定します。

### プロトタイプ

```
DWORD OsEventReset(HANDLE hOsEvent);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsEvent | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsEvent | イベント オブジェクトへのハンドル。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.10 OsMutexCreate()

### 目的

- mutex オブジェクトを生成します。

### プロトタイプ

```
DWORD OsMutexCreate(HANDLE *phOsMutex);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ phOsMutex | HANDLE* | 出力 |

### 説明

| 名前 | 説明 |
|---|---|
| phOsMutex | 新たに生成した mutex オブジェクトへのハンドルを受信する変数へのポインタ。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.11 OsMutexClose()

### 目的

- mutex オブジェクトへのハンドルを閉じます。

### プロトタイプ

```
void OsMutexClose(HANDLE hOsMutex);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsMutex | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsMutex | 閉じられる mutex オブジェクトへのハンドル。 |

### 戻り値

なし。

## A.7.12 OsMutexLock()

### 目的

- 指定した mutex オブジェクトをロックします。

### プロトタイプ

```
DWORD OsMutexLock(HANDLE hOsMutex);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsMutex | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsMutex | ロックされる mutex オブジェクトへのハンドル。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.13 OsMutexUnlock()

### 目的

- ロックした mutex オブジェクトを解放 (アンロック) します。

### プロトタイプ

```
DWORD OsMutexUnlock(HANDLE hOsMutex);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ hOsMutex | HANDLE | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| hOsMutex | アンロックされる mutex オブジェクトへのハンドル。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

## A.7.14 PrintDbgMessage()

### 目的

- Debug Monitor へデバッグメッセージを送信します。

### プロトタイプ

```
void PrintDbgMessage(
    DWORD dwLevel,
    DWORD dwSection,
    const char *format
    ...);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ dwLevel | DWORD | 入力 |
| ➢ dwSection | DWORD | 入力 |
| ➢ format | const char* | 入力 |
| ➢ argument | | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| dwLevel | Debug Monitor (データを宣言します) のレベルを指定します。0 の場合、D_ERROR を宣言します。<br>詳細は **windrvr.h** の DEBUG_LEVEL を参照してください。 |
| dwSection | Debug Monitor (データを宣言します) のレベルを指定します。0 の場合、S_MISC を宣言します。 |

| | |
|---|---|
| | 詳細は **windrvr.h** の `DEBUG_SECTION` を参照してください。 |
| format | 書式を管理する文字列 |
| argument | オプション引数、最大 256 バイト。 |

### 戻り値

なし。

## A.7.15 WD_LogStart()

### 目的

- ログ ファイルを開きます。

### プロトタイプ

```
DWORD WD_LogStart(
    const char *sFileName,
    const char *sMode);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ sFileName | const char* | 入力 |
| ➢ sMode | const char* | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| sFileName | 開くログ ファイル名。 |
| sMode | アクセスの種類。<br>たとえば、NULL または **w** の場合、書き込み用の空のファイルを開きます。指定したファイルが存在する場合、その内容を壊します。<br>**a** の場合、ファイルの終わりに書き込む (append: 追加する) ように開きます。 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

### 注釈

- ログ ファイルを開くと、すべての API 呼び出しをこのファイルに記録します。
  `WD_LogAdd()` [A.6.17] を呼んで、ログ ファイルへ出力内容を追加します。

## A.7.16 WD_LogStop()

### 目的

- ログ ファイルを閉じます。

### プロトタイプ

```
VOID WD_LogStop(void);
```

### 戻り値

なし。

## A.7.17 WD_LogAdd()

### 目的

- ログ ファイルに出力内容を追加します。

### プロトタイプ

```
VOID DLLCALLCONV WD_LogAdd(
    const char *sFormat
    ...);
```

### パラメータ

| 名前 | 型 | 入出力 |
|---|---|---|
| ➢ sFormat | const char* | 入力 |
| ➢ argument | | 入力 |

### 説明

| 名前 | 説明 |
|---|---|
| sFormat | 書式を管理する文字列。 |
| argument | 書式文字列用のオプション引数 |

### 戻り値

成功した場合、WD_STATUS_SUCCESS (0) を返します。その他の場合、対応するエラーコードを返します [A.7]。

# A.8 WinDriver ステータス コード

## A.8.1 はじめに

多くの WinDriver 関数はステータス コードを返します。正常終了した場合は 0 (WD_STATUS_SUCCESS) を、異常終了した場合は 0 以外の値を返します。
指定したステータス コードの意味を調べるのに Stat2Str() を使用します。ステータス コードとその説明を以下に示します。

## A.8.2 WinDriver が返すステータス コード

| ステータス コード | 説明 |
|---|---|
| WD_STATUS_SUCCESS | Success (成功) |
| WD_STATUS_INVALID_WD_HANDLE | Invalid WinDriver handle (無効な WinDriver のハンドル) |
| WD_WINDRIVER_STATUS_ERROR | Error (エラー) |
| WD_INVALID_HANDLE | Invalid handle (無効なハンドル) |
| WD_INVALID_PIPE_NUMBER | Invalid pipe number (無効なパイプ番号) |
| WD_READ_WRITE_CONFLICT | Conflict between read and write operations (読み込みと書き込み処理の競合) |
| WD_ZERO_PACKET_SIZE | Packet size is zero (パケット サイズが 0) |
| WD_INSUFFICIENT_RESOURCES | Insufficient resources (不十分なリソース) |
| WD_UNKNOWN_PIPE_TYPE | Unknown pipe type (未知のパイプの種類) |
| WD_SYSTEM_INTERNAL_ERROR | Internal system error (内部システム エラー) |
| WD_DATA_MISMATCH | Data mismatch (データの不整合) |
| WD_NO_LICENSE | No valid license (有効なライセンスがありません) |
| WD_NOT_IMPLEMENTED | Function not implemented (実装されていない関数) |
| WD_FAILED_ENABLING_INTERRUPT | Failed enabling interrupt (失敗した有効な割り込み) |
| WD_INTERRUPT_NOT_ENABLED | Interrupt not enabled (有効ではない割り込み) |
| WD_RESOURCE_OVERLAP | Resource overlap (リソースの重複) |
| WD_DEVICE_NOT_FOUND | Device not found (デバイスが見つかりません) |
| WD_WRONG_UNIQUE_ID | Wrong unique ID (間違った任意の ID) |
| WD_OPERATION_ALREADY_DONE | Operation already done (処理済の処理) |
| WD_USB_DESCRIPTOR_ERROR | USB descriptor error (USB ディスクリプタのエラー) |
| WD_SET_CONFIGURATION_FAILED | Set configuration operation failed (失敗した設定処理を設定) |
| WD_CANT_OBTAIN_PDO | Cannot obtain PDO (PDO の取得ができません) |
| WD_TIME_OUT_EXPIRED | Timeout expired (期限切れのタイムアウト) |
| WD_IRP_CANCELED | IRP operation cancelled (キャンセルされた IRP 処理) |

| | |
|---|---|
| WD_FAILED_USER_MAPPING | Failed to map in user space (ユーザー スペースへの割り当ての失敗) |
| WD_FAILED_KERNEL_MAPPING | Failed to map in kernel space (カーネル スペースへの割り当ての失敗) |
| WD_NO_RESOURCES_ON_DEVICE | No resources on the device (デバイスにリソースがありません) |
| WD_NO_EVENTS | No events (イベントがありません) |
| WD_INVALID_PARAMETER | Invalid parameter (不正な引数) |
| WD_INCORRECT_VERSION | Incorrect WinDriver version installed (不正な WinDriver のバージョンがインストールされています) |
| WD_TRY_AGAIN | Try again (再試行) |
| WD_INVALID_IOCTL | Received an invalid IOCTL (不正な IOCTL を受信しました) |
| WD_OPERATION_FAILED | Operation failed (操作が失敗しました) |
| WD_INVALID_32BIT_APP | Received an invalid 32-bit IOCTL (不正な 32 ビット IOCTL を受信しました) |
| WD_TOO_MANY_HANDLES | No room to add handle (ハンドルを追加する場所がありません) |
| WD_NO_DEVICE_OBJECT | Driver not installed (ドライバがインストールされていません) |

## A.8.3 USBD が返すステータス コード

以下に、WinDriver ステータス コードは USB スタック ドライバから返される USBD_XXX ステータス コードに対応します。

| **ステータス コード** | **説明** |
|---|---|
| USBD ステータスの種類 | |
| WD_USBD_STATUS_SUCCESS | USBD: Success (成功) |
| WD_USBD_STATUS_PENDING | USBD: Operation pending (処理待ち) |
| WD_USBD_STATUS_ERROR | USBD: Error (エラー) |
| WD_USBD_STATUS_HALTED | USBD: Halted (停止) |
| USBD ステータス コード (**注意:** 上記のステータスの種類とエラー コードで構成されています。たとえば、0XXYYYYYYL の X はステータスの種類、Y はエラーコード。同じエラー コードが、他のステータスの種類で現れる場合があります。) | |
| HC (ホスト コントローラ) ステータス コード (注意: WD_USBD_STATUS_HALTED のステータスの種類を使用します。) | |
| WD_USBD_STATUS_CRC | HC status: CRC |
| WD_USBD_STATUS_BTSTUFF | HC status: Bit stuffing (ビット スタッフ) |
| WD_USBD_STATUS_DATA_TOGGLE_MISMATCH | HC status: Data toggle mismatch (データ トグルの不整合) |
| WD_USBD_STATUS_STALL_PID | HC status: PID stall (PID 停止) |
| WD_USBD_STATUS_DEV_NOT_RESPONDING | HC status: Device not responding (デバイスから |

| | |
|---|---|
| | の応答がありません) |
| WD_USBD_STATUS_PID_CHECK_FAILURE | HC status: PID check failed (PID のチェック失敗) |
| WD_USBD_STATUS_UNEXPECTED_PID | HC status: Unexpected PID (予期せぬ PID) |
| WD_USBD_STATUS_DATA_OVERRUN | HC status: Data overrun (データの超過) |
| WD_USBD_STATUS_DATA_UNDERRUN | HC status: Data underrun (データの不足) |
| WD_USBD_STATUS_RESERVED1 | HC status: Reserved1 (予約 1) |
| WD_USBD_STATUS_RESERVED2 | HC status: Reserved2 (予約 2) |
| WD_USBD_STATUS_BUFFER_OVERRUN | HC status: Buffer overrun (バッファの超過) |
| WD_USBD_STATUS_BUFFER_UNDERRUN | HC status: Buffer Underrun (バファの不足) |
| WD_USBD_STATUS_NOT_ACCESSED | HC status: Not accessed (未アクセス) |
| WD_USBD_STATUS_FIFO | HC status: FIFO |
| Windows の場合のみ: | |
| WD_USBD_STATUS_XACT_ERROR | HC status: The host controller has set the Transaction Error (XactErr) bit in the transfer descriptor's status field (転送ディスクリプタのステータス フィールドに、ホスト コントローラが Transaction Error (XacctErr) ビットを設定しました) |
| WD_USBD_STATUS_BABBLE_DETECTED | HC status: Babble detected (バブルを検出しました) |
| WD_USBD_STATUS_DATA_BUFFER_ERROR | HC status: Data buffer error (データ バッファ エラー) |
| Windows CE の場合のみ: | |
| WD_USBD_STATUS_ISOCH | USBD: Isochronous transfer failed (アイソクロノス転送に失敗しました) |
| WD_USBD_STATUS_NOT_COMPLETE | USBD: Transfer not completed (転送が終了しませんでした) |
| WD_USBD_STATUS_CLIENT_BUFFER | USBD: Cannot write to buffer (バッファへ書き込みできません) |
| すべてのプラットフォームの場合のみ: | |
| WD_USBD_STATUS_CANCELED | USBD: Transfer cancelled (転送がキャンセルされました) |
| 転送が停止されたエンドポイントへ送信された場合、HCD (ホスト コントロール ドライバ)が返します: | |
| WD_USBD_STATUS_ENDPOINT_HALTED | HCD: Transfer submitted to stalled endpoint (停止されたエンドポイントへ送信された転送) |
| ソフトウェア ステータス コード (**注意:** エラー ビットのみ設定): | |
| WD_USBD_STATUS_NO_MEMORY | USBD: Out of memory (メモリーがありません) |
| WD_USBD_STATUS_INVALID_URB_FUNCTION | USBD: Invalid URB function (無効な URB 関数) |

| | |
|---|---|
| WD_USBD_STATUS_INVALID_PARAMETER | USBD: Invalid parameter (無効な引数) |
| クライアントのドライバが、未解決の転送の設定またはエンドポイント / インターフェイスを閉じる場合に返されます。 | |
| WD_USBD_STATUS_ERROR_BUSY | USBD: Attempted to close endpoint/interface/configuration with outstanding transfer (未解決の転送のエンドポイント / インターフェイス / 設定を閉じます) |
| URB の要求を完了できない場合に USBD に返されます。基本的に、これは特定のエラー コードを持つ URB のステータス項目 (IRQ が完成時) に返されます。IRQ ステータスは、WinDriver の Monitor Debug メッセージ (**wddebug_gui** / **wddebug**) ツールで表示されます | |
| WD_USBD_STATUS_REQUEST_FAILED | USBD: URB request failed (URB の要求失敗) |
| WD_USBD_STATUS_INVALID_PIPE_HANDLE | USBD: Invalid pipe handle (無効なパイプ ハンドル) |
| 要求したエンドポイントを開くのに十分な帯域幅が無い場合に返されます: | |
| WD_USBD_STATUS_NO_BANDWIDTH | USBD: Not enough bandwidth for endpoint (エンドポイントに十分な帯域幅がありません) |
| 汎用 HC (ホスト コントローラ)エラー: | |
| WD_USBD_STATUS_INTERNAL_HC_ERROR | USBD: Host controller error (ホスト コントロール エラー) |
| 短いパケットが転送を終了したときに返されます。たとえば、USBD_SHORT_TRANSFER_OK ビットが設定されていない場合: | |
| WD_USBD_STATUS_ERROR_SHORT_TRANSFER | USBD: Transfer terminated with short packet (短いパケットで終了された転送) |
| 要求した開始フレームが、USB フレームの USBD_ISO_START_FRAME_RANGE 内に無い場合、返されます。(**注意:** 停止ビットが設定されます): | |
| WD_USBD_STATUS_BAD_START_FRAME | USBD: Start frame outside range (開始フレームが範囲外です) |
| アイソクロナス転送のすべてのパケットがエラーを起こして終了した場合、HCD (ホスト コントローラ デバイス)が返します: | |
| WD_USBD_STATUS_ISOCH_REQUEST_FAILED | HCD: Isochronous transfer completed with error (エラーを起こして終了したアイソクロナス転送) |
| 指定した HC (ホスト コントローラ)のフレーム長コントロールが既に他のドライバに使用されている場合、USBD が返します: | |
| WD_USBD_STATUS_FRAME_CONTROL_OWNED | USBD: Frame length control already taken (既に使用されているフレーム長コントロール) |
| 呼出し元が自分自身のフレーム長コントロールを持っておらず、HC フレーム長を解放または修正する場合に、USBD が返します: | |
| WD_USBD_STATUS_FRAME_CONTROL_NOT_OWNED | USBD: Attempted operation on frame length control not owned by caller (呼出し元が持っていないフレーム長コントロールの処理) |
| USB 2.0 用に追加したエラー コードです (Windows の場合のみ) : | |
| WD_USBD_STATUS_NOT_SUPPORTED | USBD: API not supported/implemented (サポートされていない / 実装されない API) |

| | |
|---|---|
| WD_USBD_STATUS_INAVLID_CONFIGURATION_DESCRIPTOR | USBD: Invalid configuration descriptor (不正な設定ディスクリプタ) |
| WD_USBD_STATUS_INSUFFICIENT_RESOURCES | USBD: Insufficient resources (リソースが足りません) |
| WD_USBD_STATUS_SET_CONFIG_FAILED | USBD: Set configuration failed (設定に失敗しました) |
| WD_USBD_STATUS_BUFFER_TOO_SMALL | USBD: Buffer too small (バッファが小さ過ぎます) |
| WD_USBD_STATUS_INTERFACE_NOT_FOUND | USBD: Interface not found (インターフェイスが見つかりません) |
| WD_USBD_STATUS_INAVLID_PIPE_FLAGS | USBD: Invalid pipe flags (不正なパイプ フラグ) |
| WD_USBD_STATUS_TIMEOUT | USBD: Timeout (タイムアウト) |
| WD_USBD_STATUS_DEVICE_GONE | USBD: Device gone (デバイスがありません) |
| WD_USBD_STATUS_STATUS_NOT_MAPPED | USBD: Status not mapped (ステータスがマップされていません) |
| USBD から返る拡張アイソクロナス エラー コード。<br>これらのエラーは、アイソクロナス転送のパケット ステータス フィールドに表示します: | |
| WD_USBD_STATUS_ISO_NOT_ACCESSED_BY_HW | USBD: The controller did not access the TD associated with this packet (コントローラが、このパケットに関連付けた TD にアクセスしませんでした) |
| WD_USBD_STATUS_ISO_TD_ERROR | USBD: Controller reported an error in the TD (コントローラが TD でエラーを報告しました) |
| WD_USBD_STATUS_ISO_NA_LATE_USBPORT | USBD: The packet was submitted in time by the client but failed to reach the miniport in time (クライアントがパケットを送信しましたが、ミニポートに届きませんでした) |
| WD_USBD_STATUS_ISO_NOT_ACCESSED_LATE | USBD: The packet was not sent because the client submitted it too late to transmit (クライアントの転送が遅くて、パケットが届きませんでした) |

# 付録 B
# トラブルシューティングとサポート

開発者向けの技術情報が Web サイト http://www.xlsoft.com/jp/products/windriver/products.html より参照できます。以下の文書がありますので、参考にしてください。

- テクニカル ドキュメント
- FAQ
- サンプル コード
- クイック スタート ガイド

# 付録 C
# 評価版 (Evaluation Version) の制限

## C.1 WinDriver Windows

- 毎回 WinDriver を起動すると評価版であることを示すメッセージが表示されます。
- DriverWizard を使用する際に、評価版を起動していることを知らせるメッセージのダイアログ ボックスが、ハードウェアと相互作用するたびに表示されます。
- DriverWizard:
  - 毎回 DriverWizard を起動すると評価版であることを示すメッセージが表示されます。
  - DriverWizard を使用してハードウェアと相互作用するたびに、評価版であることを示すメッセージが表示されます。
- WinDriver は最初のインストールから 30 日間だけ使用可能です。

## C.2 WinDriver Windows CE

- 毎回 WinDriver を起動すると評価版であることを示すメッセージが表示されます。
- WinDriver CE Kernel (**windrvr6.dll**) は 1 度に 60 分間まで動作します。
- DriverWizard (Windows 2000 / XP / Server 2003 / Server 2008 / Vista / 7 PC ホストで使用する場合):
  - 毎回 DriverWizard を起動すると評価版であることを示すメッセージが表示されます。
  - DriverWizard を使用してハードウェアと相互作用するたびに、評価版であることを示すメッセージが表示されます。

## C.3 WinDriver Linux

- 毎回 WinDriver を起動すると評価版であることを示すメッセージが表示されます。
- DriverWizard:
  - 毎回 DriverWizard を起動すると評価版であることを示すメッセージが表示されます。
  - DriverWizard を使用してハードウェアと相互作用するたびに、評価版であることを示すメッセージが表示されます。
- WinDriver のカーネル モジュールは 1 度に 60 分間まで動作します。
  WinDriver を継続して使用するには、次のコマンドを使用して WinDriver カーネル モジュールをリロード (モジュールのアンロードおよびロード) します。

  **注意:** root 権限で以下のコマンドを実行する必要があります。

  アンロードするには:
  ```
  /sbin/modprobe –r windrvr6
  ```

ロードするには:
**<path to wdreg>/wdreg windrvr6**

**wdreg** は **WinDriver/util/** ディレクトリ以下にあります。

# 付録 D
# WinDriver の購入

Windows の [スタート] メニューの [プログラム] - [WinDriver] - [Order Form] にある申込用紙に記入し、電子メール、ファックス、または郵送してください。

WinDriver のライセンスを電子メール、またはファックスで返信致します。

## お問い合わせ先:

**エクセルソフト株式会社**

〒108 - 0073
東京都港区三田 3-9-9 森伝ビル 6F
Phone: 03 - 5440 - 7875 Fax: 03 - 5440 - 7876
メール: xlsoftkk@xlsoft.com
ホームページ: http://www.xlsoft.com/

# 付録 E
# ドライバの配布 - 法律問題

WinDriver は、開発者ごとにライセンスされます。WinDriver の Node-Locked ライセンスは一台のマシンで一人の開発者が無制限の数のドライバを開発し、ロイヤリティなしで作成したドライバを配布することを許可します。

windrvr.h ファイル は配布できません。WinDriver の機能を説明した如何なるソース ファイルの配布もできません。WinDriver のライセンス契約書については、**WinDriver/docs/license.pdf** ファイルを参照してください。

# 付録 F
# その他のドキュメント

### 最新マニュアル

最新版の WinDriver ユーザーズ ガイドは、Jungo 社の Web サイトより入手可能です: 最新版の WinDriver 日本語ユーザーズ ガイドは、下記Jungo 社の Web サイトより入手可能です:
http://www.xlsoft.com/jp/products/download/download.html

### バージョン履歴

WinDriver のバージョン履歴は、http://www.xlsoft.com/jp/products/windriver/wdversion.html を参照してくださいこの Web サイトでは、WinDriver の各バージョンで追加されたすべての新機能、強化および修正リストを参照することができます。

### テクニカル ドキュメント

次の Web サイトから、テクニカル ドキュメント データベースも利用可能です:
http://www.xlsoft.com/jp/products/windriver/support/tech_docs_indexes/main_index.html
テクニカル ドキュメント データベースには、WinDriver の機能、ユーティリティ、API とその正しい使用方法についての詳細な説明、一般的な問題のトラブルシューティング、役立つヒント、よくある質問が含まれています。

# WinDriver

## ユーザーズ ガイド

2009 年 12 月 10 日

発行　**エクセルソフト株式会社**

〒108-0073 東京都港区三田3-9-9 森伝ビル6F

TEL 03-5440-7875　FAX 03-5440-7876

E-MAIL: xlsoftkk@xlsoft.com

ホームページ: http://www.xlsoft.com/



Translated by

米国 XLsoft Corporation

12K Mauchly

Irvine, CA 92618  USA

URL: http://www.xlsoft.com/

E-Mail: sales@xlsoft.com