

NPIGFガイド

● 概 要

ご使用にあたっての注意事項

● 関数概要



各関数の概要・使用方法を説明します。

ご注意

- ・ 本書の内容については、万全を期して作成いたしましたが、万一ご不審な点や誤り、記載もれなど、お気づきの点がございましたらご連絡ください。
- ・ 本書の内容については、予告なしに変更することがあります。最新の情報はお問い合わせください。
- ・ 本書の内容の一部または全部を無断で転載、複写、複製、改ざんすることは固くお断りします。
- ・ ご利用いただいた結果の影響については、一切の責任を負いかねますのでご了承ください。
- ・ 本製品がお客様により不適切に使用されたり、本書の内容に従わずに取り扱われたり、または弊社以外の第三者により修理・変更されたことなどに起因して発生した損害などにつきましては、責任を負いかねますのでご了承ください。

記号について

本書では以下の記号が使われています。それぞれの記号の意味をよく理解してから製品を取り扱ってください。

	ご使用上、必ずお守りいただきたいことを記載しています。 この表示を無視して間違った取り扱いをすると、製品の故障や動作不良の原因になる可能性があります。
	補足説明や関連事項を記載しています。

使用制限

本製品を航空機・列車・船舶・自動車などの運行に直接関わる装置・防災防犯装置・各種安全装置など機能・精度等において高い信頼性・安全性が必要とされる用途に使用される場合は、これらのシステム全体の信頼性および安全を維持するためにフェールセーフ設計や冗長設計の措置を行うなど、システム全体の安全設計にご配慮いただいた上で当社製品をご使用いただきますようお願いいたします。

本製品は、航空宇宙機器、幹線通信機器、原子力制御機器、医療機器など、きわめて高い信頼性・安全性が必要とされている用途への使用を意図しておりませんので、これらの用途には本製品の適合性をお客様において十分ご確認のうえ、ご判断頂きますようお願いいたします。

もくじ

・ 概要	
－システム構成	4
－各種設定 / 注意事項	5
－構造体定義	6
・ 関数概要	
－概要	7
－関数一覧	8
関数	
－NPrinterInitEpW／NPrinterInitEp	10
－NReadInfoEpW／NReadInfoEp	12
－NImagePrintEpW／NImagePrintEp	
【NImagePrintEpW_DX／NImagePrintEp_DX】	14
－NPrintEpW／NPrintEp 【NPrintEpW_DX／NPrintEp_DX】	17
－NGetStEpW／NGetStEp 【NGetStEpW_DX／NGetStEp_DX】	19
－NGetInfoEpW／NGetInfoEp	
【NGetInfoEpW_DX／NGetInfoEp_DX】	21
－NStartOnChangeStW／NStartOnChangeSt	
【NStartOnChangeStW_DX／NStartOnChangeSt_DX】	25
－NEndOnChangeSt	27
－NSetTextEpW／NSetTextEp	29
－NSetBarcodeEpW／ NSetBarcodeEp	31
－NSetImageEpW / NSetImageEp	33
－NSetCellEpW / NSetCellEp	35
－NGetTextEpW / NGetTextEp	37
－NGetBarcodeEpW / NGetBarcodeEp	39
－NGetImageEpW / NGetImageEp	41
－NGetCellEpW / NGetCellEp	43
－NSaveDataEpW / NSaveDataEp	45
－エラーコード一覧	47
－関数エラーコード一覧	49

システム構成

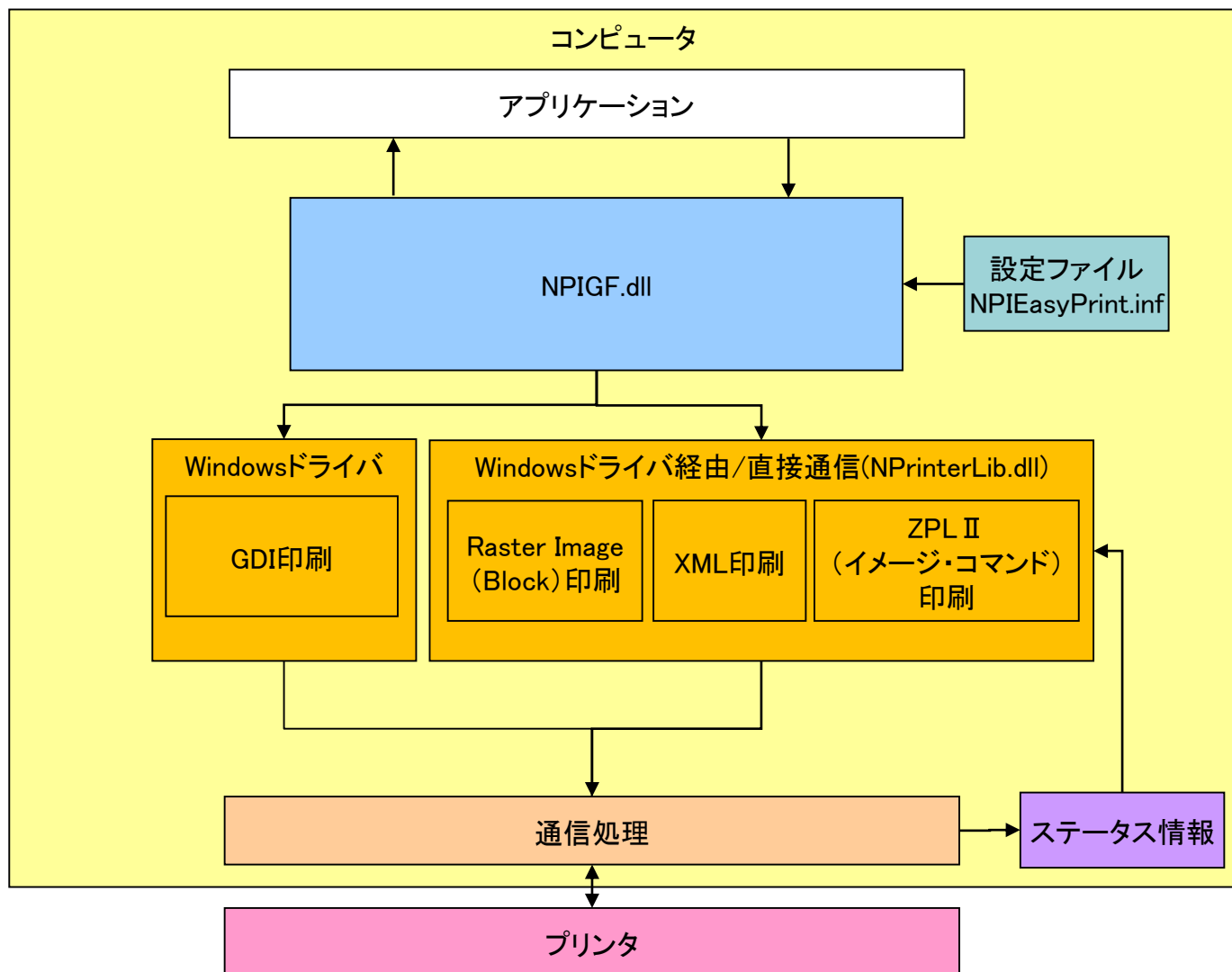
NPIGFを使用して、コンポーネント操作や描画、印刷機能、ステータス監視をアプリケーションに組み込むことができます。

NPIGFとして提供されるファイルはDynamic Link Library (DLL)となります。

提供ファイル

NPIGF.dll、32bit・64bit版 NPrinterLib.dll 32bit・64bit版

NPIGFを使用したときのシステム構成



※ドライバポートを使用する時のプリンタ名はプリンタドライバ名となります

開発言語

Win32 : Visual C++

Windows対応バージョン

Windows10 以降 32bit/64bit 対応

インターフェース

USB、RS232C

ライブラリインストール

ライブラリのみで使用する場合

提供ファイルのNPrinterLib.dll、NBarcodeLib.dll、NResetPrinter.exeをSystem32フォルダに保存してください。64bit環境で32bit版NPIGFを使用する場合は、32bit版dllをSysWOW64フォルダに保存してください。

※サンプルアプリケーションは32bit版で動作するため、32bit版NPrinterLib.dllをSysWOW64フォルダに配置するか実行ファイル(exe)と同じフォルダ内に配置する必要があります。

※NBarcodeLib.dll、NResetPrinter.exeはドライバフォルダ配下のInstallフォルダ内にファイルを同梱してあります。

ドライバと併用して使用する場合

ドライバをセットアップすると自動的にインストールされるので、特別設定を必要としません。ドライバのセットアップについてはドライバの「導入ガイド」を参照してください。

使用方法

1. ライブラリをインストール

2. 提供されているNPI EasyPrintのZipフォルダを展開します。

3. NPIGF.dllファイルをアプリケーションから読み込んで関数を呼び出します。

NPIGF.dllの配置場所はアプリケーションの実行ファイルと同じフォルダ内もしくはSystem32、SysWOW64となります。

アプリケーション実行ファイルの配置場所はSystem32、SysWOW64フォルダ内では使用することはできません。

※関数の実装方法、使い方につきましてはNPIGFSample(サンプルアプリケーション)と本資料と合わせてご参照ください。

NPIGF.dll 配置例

通常パターン1

Applicationフォルダ(C:¥Users¥User¥Application)

Application.exe

NPIGF.dll

通常パターン2

Applicationフォルダ(C:¥Users¥User¥Application)

Application.exe

SysWOW64フォルダ(C:¥Windows¥SysWOW64)

NPIGF.dll

NGパターン

Applicationフォルダ(C:¥Windows¥System32¥Application)

Application.exe

NPIGF.dll

※NPIGFは複数のスレッド呼び出しに対応していません。(スレッドフリー未対応)

構造体定義

プリンタ印字設定構造体

定義名	PRTSetInfo		
型	属性	意味	設定値
int	border	描画コンポーネント枠表示	0:枠なし 1:枠あり
int	makeSelect	描画・印字領域作成	0:印刷用 1:プレビュー用
wchar_t*	prtName	印字先プリンタ名	
int	printCnt	印字枚数	
int	copyCnt	複写枚数	
int	Interval	印刷間隔(ms)	

注意

- ・borderの枠表示はmakeSelectがプレビュー用時のみ有効となります。
- ・1枚だけ印刷する場合にはprintCnt(印字枚数)とcopyCnt(複写枚数)の両方に1を指定してください。
- ・Interval(印刷間隔)はプリンターへの印字要求の間隔となります。
印字が完了してからの間隔とはなりませんのでご注意ください。

利用方法

LoadLibrary関数にてNPIGF.dllを読み込みます。

関数が成功するとDLLのハンドルが返りますので、そのハンドルをGetProcAddress関数に渡すと、DLL 関数のアドレスを取得できます。

詳しくはリリース媒体のサンプルプログラム (Document/NPIGFSample) の NPIGFAPIファイルをご確認下さい。

ドライバ接続について

ドライバポートを使用する際はNEnumPrintersの呼び出しは必要ありません。

Bluetooth接続について

当NPIGFではBluetooth接続を行う前に、PCとプリンタのペアリングを済ませておく必要があります。

シリアル接続について

当NPIGFではシリアル接続を行う前に、使用するポートのポート設定を済ませておく必要があります。

プリンタとPCのポート設定が合わないと正しくデータの送受信を行えません。

直接通信時の注意事項

直接通信で使用するポートについては、ドライバをセットアップしないでください

テンポラリ及び一時ファイルの保管場所

デフォルトの状態では、OSセットアップドライブの直下にNPIフォルダが作成されます。

保管場所の変更はレジストリに下記のキーを作成し、その下にDirPathを文字列値として作成して使用されるフォルダを指定してください。

HKEY_LOCAL_MACHINE¥SOFTWARE¥NPI

例.

[HKEY_LOCAL_MACHINE¥SOFTWARE¥NPI]

"DirPath"="d:¥¥Printer"

64bit環境で32bit版SDKを使用する場合は上記レジストリにアクセスできない恐れがあるため、

[HKEY_LOCAL_MACHINE¥SOFTWARE¥WOW6432Node¥NPI]

フォルダにDirPathを記述してください。

関数一覧(1/2)

以下の 関数 が用意されています。

種別	関数概要	関数名	備考
SDK	プリンタ情報管理ファイル作成	NEnumPrinters	ドライバ提供資料 SDKガイドを参照

種別	関数概要	関数名	備考
		UNICODE版／ASCII版	
制御系	プリンタ初期化	NPrinterInitEpW／NPrinterInitEp	
入力系	ファイル読込	NReadInfoEpW／NReadInfoEp	
出力系	イメージデータ出力	NImagePrintEpW／NImagePrintEp	
		NImagePrintEpW_DX／NImagePrintEp_DX	※1
	コマンドデータ送信 出力	NPrintEpW／NPrintEp	
		NPrintEpW_DX／NPrintEp_DX	※1
制御系	プリンタステータス 取得	NGetStEpW／NGetStEp	
		NGetStEpW_DX／NGetStEp_DX	※1
	拡張情報取得	NGetInfoEpW／NGetInfoEp	
		NGetInfoEpW_DX／NGetInfoEp_DX	※1
イベント系	ステータス変更 イベント監視開始	NStartOnChangeStW／NStartOnChangeSt	
		NStartOnChangeStW_DX／NStartOnChangeSt_DX	※1
	ステータス変更 イベント監視終了	NEndOnChangeSt	

※DX系関数について

引数のプリンタ名がない代わりに、
プリンタ初期化時に指定されたプリンタ名が使用されます。
プリンタ初期化以降プリンタ名が変更されず指定する必要がない場合に使用します。

※1: プリンタ初期化関数実行後使用可能

関数一覧(2/2)

以下の 関数 が用意されています。

種別	関数概要	関数名	備考
		UNICODE版／ASCII版	
編集系	テキストデータ編集設定	NSetTextEpW／NSetTextEp	
	バーコードデータ編集設定	NSetBarcodeEpW／NSetBarcodeEp	
	イメージデータ編集設定	NSetImageEpW／NSetImageEp	
	セルデータ編集設定	NSetCellEpW／NSetCellEp	
	テキストデータ情報取得	NGetTextEpW／NGetTextEp	
	バーコードデータ情報取得	NGetBarcodeEpW／NGetBarcodeEp	
	イメージデータ情報取得	NGetImageEpW／NGetImageEp	
	セルデータ情報取得	NGetCellEpW／NGetCellEp	
	最終データ保存設定	NSaveDataEpW／NSaveDataEp	

関数概要

関数名		NPrinterInitEpW / NPrinterInitEp	
引数名	IN/OUT	型	説明
i_pid	I	PWCHAR / PCHAR	プリンタ名 (NULL指定可)
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容			
<div>・引数i_pid で指定したプリンタのステータス監視を行います。 初回関数実行時プリンタがステータスが取得可能(プリンタオンライン状態)な場合、プリンタ接続時発行(下記備考参照)コマンドを実行します。 初期化後引数i_pid 指定されたプリンタ名を保存され関数名の語尾が【_DX】関数はすべて使用可能となります。 関数実行後プリンタがオフライン(例 USB未接続)からオンライン(例 USB接続)になった場合、プリンタ接続時発行コマンド(下記備考参照)が実行されます。</div> <div>・引数i_pidをNULL指定で実行するとステータス監視を停止します。 初期化時保存されているプリンタ名が破棄されます。</div> <div>・プリンタの出力形式がGDIもしくはXMLの場合、 プリンタステータス取得機能が未対応のためステータス監視、プリンタ接続時発行コマンドの実行は行いません。</div> <div>・設定ファイル(NPIEasyPrint.inf)が作成されていない場合、本関数を実行することで新規作成されます。 設定ファイルが既に作成されている場合は設定内容を更新します。</div>			
備考			
プリンタ接続時発行コマンドは 印字スピード、印字濃度、リセット等が設定できます。 詳細は easyprintHelp 3. 2. 1 コマンド発行一覧参照して下さい。			

使用例

```
PWCHAR pid = L"NPI Integration Driver";
BOOL debug = false;
PWCHAR dat = L"¥"test¥"0A";
DWORD size = (DWORD) wcslen(dat);
PINT jobID = NULL;
int res;

// プリンタ初期化・ステータス監視開始・infファイル作成
res = NPrinterInitEpW(pid);

if(res == 0)
{
    // 印字
    res = NPrintEpW_DX(debug, dat, size, jobID);
}

// プリンタ名破棄・ステータス監視停止
res = NPrinterInitEpW(NULL);
```

NPIEasyPrint.inf

[NPI Integration Driver]

dpi=0

FullCut=0

PartialCut=0

MaxWidth=0.00

HeadToCutter=0.00

PaperType=0

OutputFormat=2

ConnectType=0

PrinterInit=INIT__.;PRT_SPD:04

関数名	NReadInfoEpW／NReadInfoEp		
引数名	IN/OUT	型	説明
i_fname	I	PWCHAR/PCHAR	ファイル名(フルパス)
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fname(ファイル名(フルパス))で指定したXML形式ファイルを読み込み フォーマットの作成、各コンポーネントの作成を行います。</div> <div>※XML形式で記述されているファイルであれば拡張子がXMLでなくても読み込みが可能です。</div>		
<div><div>参考</div> ※XML仕様、説明は「XMLリファレンス」を参照</div>			
備考			

使用例

```
PWCHAR strFileName;  
int res;  
  
// 読み込み用ダイアログ表示  
if(SaveDialog1->Execute())  
{  
    // 読み込みファイルのフルパスとファイル名  
    strFileName = SaveDialog1->FileName;  
  
    // XMLファイルを読み込み  
    res = NReadInfoEpW(strFileName);  
}
```

関数概要

関数名		NImagePrintEpW／NImagePrintEp【NImagePrintEpW_DX／NImagePrintEp_DX】	
引数名	IN/OUT	型	説明
i_prtSetInfo	I	PVOID (PRTSetInfo)	プリンタ印字設定構造体
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・コンポーネント一覧をデバイスコンテキストへ描画のみか、描画と印字を行います。</div> <div>・印字する場合、設定ファイル(NPIEasyPrint.inf)を読み込んだ内容で印字します。</div>		
<div><div>参考</div> ※プリンタ印字設定構造体 詳細は別ページ 構造体定義を参照</div>			
備考			

使用例

NPIEasyPrint.inf

[NPI Integration Driver]

dpi=0

FullCut=0

PartialCut=0

MaxWidth=0.00

HeadToCutter=0.00

PaperType=0

OutputFormat=2

ConnectType=0

PrinterInit=

//プリンタ情報構造体定義

```
typedef struct tmpPRTSetInfo
```

```
{
```

```
    int border;           // 0:コンポーネント枠なし 1:コンポーネント枠あり
```

```
    int makeSelect;       // 0:印刷用 1:プレビュー用
```

```
    PWCHAR prtName;      // 印字先プリンタ名
```

```
    int printCnt;         // 印字枚数
```

```
    int copyCnt;          // 複写枚数(ZPL)
```

```
    int Interval;         // 印刷間隔(ms)
```

```
} PRTSetInfo;
```


使用例

test.xml

```
<?xml version="1.0" encoding="Shift_JIS" standalone="yes" ?>
<npixml>
<request>
<print>
<page Name = "Format1" Length = "120.00" Width = "120.00" >
  <text Name = "Component1" Kind = "5" LayoutOrder = "1" Left = "10.00" Top = "10.00"
    Width = "50.00" Height = "30.00" AutoSize = "1" FontName = "font_d" Size = "32" >
    8365834c83588367</text>
</page>
</print>
</request>
</npixml>
```

```
PRTSetInfo prtSetInfo;
prtSetInfo.border = 0;
prtSetInfo.makeSelect = 1;
prtSetInfo.prtName = L"NPI Integration Driver";
prtSetInfo.printCnt = 2;
prtSetInfo.copyCnt = 1;
prtSetInfo.Interval = 1000;
```

// XML読み込み

```
NReadInfoEpW (L"test.xml");
```

// コンポーネント描画、印字

```
NImagePrintEpW(&prtSetInfo);
```

※イメージ



関数概要

関数名		NPrintEpW / NPrintEp 【NPrintEpW_DX / NPrintEp_DX】	
引数名	IN/OUT	型	説明
i_pid	I	PWCHAR / PCHAR	プリンタ名
i_debug	I	BOOL	コマンド文字列ファイル出力 (true:出力有／false:出力無)
i_dat	I	PWCHAR / PCHAR	送信データ(16進文字列)
i_size	I	DWORD	出力文字数
o_jobid	O	PINT	プリントジョブID(NULL指定可)
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<p>・指定したプリンタへ出力文字数分16進文字データを送信します。</p> <p>”(ダブルコーテーション)、<> (大なり、小なり)、[] (大括弧)で括られたデータと ‘(シングルコーテーション)が先頭にあるデータは、以下の通り変換して送信します</p> <p>1. ”(ダブルコーテーション)で括られた文字列 ⇒ 文字列として変換します。(”ABC” ⇒ 0x41,0x42,0x43)</p> <p>2. <> (大なり、小なり)で括られたファイル名(パス込み) ⇒ ファイル内容(バイナリデータ)を出力します。</p> <p>3. [] (大括弧)で括られた画像ファイル名(パス込み、bmp形式) ⇒ 画像をラスタビットイメージコマンドに変換して出力します。</p> <p>4. ‘(シングルコーテーション)が先頭にある文字列 ⇒ コメントとして処理されます(出力されない)</p>		
備考	<p>・引数 i_debug(コマンド文字列ファイル出力) trueの場合、 引数 i_dat(送信データ)と出力した日時を記録したログファイルを出力します。 出力先: NPIGF.dllと同じフォルダ内 ファイル名: debug.txt</p>		

使用例

```
PWCHAR pid = L"NPI Integration Driver";  
BOOL debug = true;  
PWCHAR dat = L"¥"test¥"0A";  
DWORD size = (DWORD) wcslen(dat);  
PINT jobID = NULL;  
int res;  
  
// プリント印字  
res = NPrintEpW(pid, debug, dat, size, jobID);
```

関数概要

関数名		NGetStEpW / NGetStEp 【NGetStEpW_DX / NGetStEp_DX】	
引数名	IN/OUT	型	説明
i_pid	I	PWCHAR / PCHAR	プリンタ名
o_st	O	PVOID	ステータス情報 (NULL 指定可)
o_size	O	PDWORD	ステータス情報取得格納エリアサイズ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_pid に指定したプリンタのステータス情報を取得します。</div> <div>・引数 o_st をNULL指定することでステータス情報格納に必要なバイト数(引数o_size)を取得します。</div>		
備考			

参考

ステータス情報の詳細は対象プリンタの製品仕様書「エラーの処理」参照

使用例

```
PWCHAR pid = L"NPI Integration Driver";
PVOID o_st;
DWORD o_size = 0;
int res;

// プリンタステータス情報エリアサイズ取得
res = NGetStEpW(pid, NULL, &o_size) ;

if(res == 0)
{
    // プリンタステータス情報エリア確保
    o_st = (PBYTE)malloc(o_size*sizeof(BYTE));
    memset(o_st, 0x00, o_size*sizeof(BYTE));

    // プリンタステータス情報取得
    res = NGetStEpW(pid, o_st, &o_size) ;

    free(o_st);
}
```

関数概要

関数名		NGetInfoEpW / NGetInfoEp 【NGetInfoEpW_DX / NGetInfoEp_DX】	
引数名	IN/OUT	型	説明
i_pid	I	PWCHAR / PCHAR	プリンタ名
i_id	I	BYTE	拡張情報ID
o_dat	O	PVOID	拡張情報(NULL指定可)
o_size	O	PDWORD	拡張情報格納エリアサイズ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	・引数 i_pid に指定したプリンタと引数 i_id(拡張情報ID)で指定したIDの拡張情報を取得します。 ・引数 o_dat をNULL指定することで拡張情報格納に必要なバイト数(引数o_size)を取得します。		
備考			

使用例

```
PWCHAR pid = L"NPI Integration Driver";
BYTE id = (BYTE)2;
PVOID o_dat;
DWORD o_size = 0;
int res;

//拡張情報格納エリアサイズ取得
res = NGetInfoEpW(pid, id, NULL, &o_size) ;

if(res == 0)
{
    //拡張情報格納エリア確保
    o_dat = (PBYTE)malloc(o_size*sizeof(BYTE));
    memset(o_dat, 0x00, o_size*sizeof(BYTE));
    //拡張情報取得
    res = NGetInfoEpW(pid, id, o_dat, &o_size) ;

    free(o_dat);
}
```

プリンタ拡張情報一覧

- 種別1 : 4 バイト(固定) : 更新フラグ(4バイト) <拡張ステータス> 1Byte: 7~0、2Byte: 15~8、
3Byte: 23~16、4Byte: 31~24
- 種別2 : 32バイト(テリタ) : 更新フラグ(4バイト) <モデル名>
- 種別3 : 8 バイト(固定) : 更新フラグ(4バイト) <F/Wバージョン>
- 種別4 : 8 バイト(固定) : 更新フラグ(4バイト) <Bootバージョン>
- 種別5 : 4 バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別6 : 4 バイト(固定) : 更新フラグ(4バイト) <ヘッド通電ドットライン数>
- 種別7 : 4 バイト(固定) : 更新フラグ(4バイト) <走行ドットライン数>
- 種別8 : 4 バイト(固定) : 更新フラグ(4バイト) <カット回数>
- 種別9 : 16バイト(固定) : 更新フラグ(4バイト) <ユーザメンテナンスカウンタ:
ヘッド通電ドットライン数,
走行ドットライン数,
カット回数,
予備>
- 種別10: 16バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別11: 64バイト(テリタ) : 更新フラグ(4バイト)
- 種別12: 32バイト(テリタ) : 更新フラグ(4バイト)
- 種別13: 32バイト(固定) : 更新フラグ(4バイト) <NV登録状況>
- 種別14: 32バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別15: 16バイト(固定) : 更新フラグ(4バイト)
- 種別16: 16バイト(固定) : 更新フラグ(4バイト)
- 種別17: 16バイト(固定) : 更新フラグ(4バイト)
- 種別18: 16バイト(固定) : 更新フラグ(4バイト)
- 種別19: 8 バイト(固定) : 更新フラグ(4バイト) <印字完了通知: 印字開始/終了コマンドの指定に
より、終了コマンド処理時に任意ID、終了ステータス記載>
- 種別20: 8 バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別21: 8 バイト(固定) : 更新フラグ(4バイト)
- 種別22: 8 バイト(固定) : 更新フラグ(4バイト)
- 種別23: 8 バイト(固定) : 更新フラグ(4バイト)
- 種別24: 4 バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別25: 4 バイト(固定) : 更新フラグ(4バイト) <転送完了通知: 転送の完了したジョブID記載>
- 種別26: 4 バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別27: 4 バイト(固定) : 更新フラグ(4バイト) <予約>
- 種別28: 2 バイト(固定) : 更新フラグ(4バイト) <F/Wチェックサム>
- 種別29: 2 バイト(固定) : 更新フラグ(4バイト)
- 種別30: 2 バイト(固定) : 更新フラグ(4バイト)
- 種別31: 2 バイト(固定) : 更新フラグ(4バイト) <通信状態情報: USB: 0x0000固定
COM: 1バイト目CTS 2バイト目DSR
※更新フラグに最終ステータス取得タイムスタンプ>

※種別25、31を除きプリンタにて機能実装していない情報に付いては取得できた内容に妥当性はありません。
※ここで記載した内容が全てのプリンタで使用出来るとは限りません。

種別32:8 バイト16進文字列/文字列
種別33:4 バイト16進文字列/文字列
種別34:2 バイト16進文字列/文字列
種別35:8 バイト16進文字列
種別36:8 バイト16進文字列
種別37:4 バイト16進文字列
種別38:4 バイト16進文字列
種別39:2 バイト16進文字列
種別40:2 バイト16進文字列

※種別11/32/33/34はサブIDによって返信される情報は変更されます

ネットワーク関連拡張情報種別一覧

MACアドレス(6バイト)	種別32-0	応答:16進文字列
IPアドレス(4バイト)	種別33-0	応答:16進文字列
サブネットマスク(4バイト)	種別33-1	応答:16進文字列
デフォルトゲートウェイ(4バイト)	種別33-2	応答:16進文字列
DNSサーバアドレス(4バイト×5)	種別33-3	応答:16進文字列
印刷タイムアウト(ミリ秒単位:4バイト)	種別33-14	応答:16進文字列
通信モード(インフラストラクチャー,アドホック)	種別34-4	応答:16進文字列
バンド(2.4GHz/5GHz)	種別34-5	応答:16進文字列
送信パワーレベル(Low/Medium/High)	種別34-6	応答:16進文字列
チャンネル	種別34-7	応答:16進文字列
セキュリティタイプ(OPEN/WPA/WPA2/WEP)	種別34-8	応答:16進文字列
暗号化タイプ(OPEN/TKIP/AES)	種別34-9	応答:16進文字列
IP設定方法(自動(DHCP/APIPA)/手動)	種別34-10	応答:16進文字列

関数概要

関数名		NStartOnChangeStW／NStartOnChangeSt 【NStartOnChangeStW_DX／NStartOnChangeSt_DX】	
引数名	IN/OUT	型	説明
i_pid	I	PWCHAR / PCHAR	プリンタ名
i_adr	I	STADDRESS	ステータスイベント関数アドレス
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容		<div>・引数 i_pid で指定したプリンタのステータスを常時監視を行い、ステータス変化時に引数 i_adr に指定されたステータスイベント関数を実行します。</div> <div>・引数 i_pidをNULL指定することでステータス常時監視を停止します。</div> <div>・ステータス監視はプリンタ1台のみ可能となります。</div> <div>・引数 i_adr STADDRESS型は以下の通り宣言されています。</div> <div>typedef void(_stdcall *STADDREES)(PVOID, PVOID);</div> <div>引数 :</div> <div>o_bst(PVOID型) : 前回値ステータス格納エリア</div> <div>o_ast(PVOID型) : 今回値ステータス格納エリア</div> <div>戻り値 :</div> <div>VOID型</div>	
備考			

使用例

//コールバック関数

```
static void _stdcall StatusCallBack(PVOID o_bst, PVOID o_ast)
{
    PDWORD bst = (PDWORD)o_bst; // 前回値ステータス格納エリア
    PDWORD ast = (PDWORD)o_ast; // 今回値ステータス格納エリア
}
```

PWCHAR pid = L"NPI Integration Driver";

int res;

// プリントステータス監視開始

res = NStartOnChangeStW(pid, StatusCallBack);

// プリントステータス監視停止

res = NStartOnChangeStW(NULL, StatusCallBack);

関数概要

関数名		NEndOnChangeSt		
引数名	IN/OUT	型	説明	
なし				
戻り値	INT			
・エラー(-値)、警告(+値)、正常終了(0)				
処理内容	<div>・プリンタのステータス常時監視を停止します。</div> <div>・NStartOnChangeStWの引数 i_pidをNULL指定で実行した場合もステータス常時監視は停止することが可能です。</div>			
備考				

使用例

//コールバック関数

```
static void __stdcall StatusCallBack(PVOID o_bst, PVOID o_ast)
```

```
{
```

```
    PDWORD bst = (PDWORD)o_bst; // 前回値ステータス格納エリア
```

```
    PDWORD ast = (PDWORD)o_ast; // 今回値ステータス格納エリア
```

```
}
```

```
PWCHAR pid = L"NPI Integration Driver";
```

```
int res;
```

// プリンタステータス監視開始

```
res = NStartOnChangeStW(pid, StatusCallBack);
```

// プリンタステータス監視停止

```
res = NEndOnChangeSt();
```

関数概要

関数名		NSSetTextEpW / NSSetTextEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_txtid	I	PWCHAR / PCHAR	テキストID
i_dat	I	PWCHAR / PCHAR	テキストデータ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_txtidを検索、対象のテキストコンポーネントのテキストデータを引数 i_datへ変更します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";  
PWCHAR txtid = L"Component1";  
PWCHAR dat = L"test data";  
int res;  
  
// テキストコンポーネント変更  
res = NSetTextEpW (fid, txtid, dat) ;
```

関数概要

関数名		NSSetBarcodeEpW / NSetBarcodeEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_barid	I	PWCHAR / PCHAR	バーコードID
i_dat	I	PBYTE	バーコードデータ
i_size	I	DWORD	バーコードデータサイズ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_baridを検索、対象のバーコードコンポーネントのバーコードデータを引数 i_datへ変更します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";
PWCHAR barid = L"Component1";
PBYTE dat;
int size;
int res;

// 16進文字列からバイト変換 ※サンプルアプリケーション参照
dat = GetPBYTE(Edit->Text, &size);

// バーコードコンポーネント変更
res = NSetBarcodeEpW (fid, barid, dat, size);
```

関数概要

関数名		NSSetImageEpW / NSetImageEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_imgid	I	PWCHAR / PCHAR	イメージID
i_hdc	I	HDC	デバイスコンテキストハンドル
i_w	I	DWORD	イメージ幅
i_h	I	DWORD	イメージ高さ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_imgidを検索、対象のイメージコンポーネントのイメージデータを引数 i_hdc デバイスコンテキストを読み込んで変更します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";
PWCHAR imgid = L"Component1";
DWORD width;
DWORD height;
int res;
HDC hdc;
HBITMAP hbitmap;
LPDWORD lpPixel;    // ビットマップピクセルへのポインタ
BITMAPINFO bmpInfo; // DIB形式の画像情報

// イメージ幅、高さ
width = 200;
height = 200;

// DIBの情報を設定する
bmpInfo.bmiHeader.biSize = sizeof(BITMAPINFOHEADER); // BITMAPINFO構造体サイズ
bmpInfo.bmiHeader.biWidth = width;                  // 画像の横幅
bmpInfo.bmiHeader.biHeight = height;                 // 画像の縦幅
bmpInfo.bmiHeader.biPlanes = 1;                      // プレーン数
bmpInfo.bmiHeader.biBitCount = 24;                   // ビット数
bmpInfo.bmiHeader.biCompression = BI_RGB;           // 圧縮コード

// ビットマップハンドル作成
hbitmap = CreateDIBSection(NULL, &bmpInfo, DIB_RGB_COLORS, (void**)&lpPixel, NULL, 0);
// デバイスコンテキスト作成
hdc = CreateCompatibleDC(NULL);

// 画面イメージ情報からデバイスコンテキストへ変換
BitBlt(hdc, 0, 0, width, height, Img->Canvas->Handle, 0, 0, SRCCOPY);

// イメージコンポーネントへ画像読み込み
res = NSetImageEpW(fid, imgid, hdc, width, height);
```

関数概要

関数名		NSSetCellEpW / NSetCellEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_tableid	I	PWCHAR / PCHAR	テーブルID
i_col	I	INT	列
i_row	I	INT	行
i_TextContent	I	PCHAR	テキストデータ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_tableidを検索、対象のテーブルコンポーネント内の指定した引数 i_col(列)、i_row (行)に該当するセルのテキストデータを引数 i_TextContentへ変更します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";  
PWCHAR tableid = L"Component1";  
int res;
```

```
// テーブルコンポーネント2列1行目のセルコンポーネント変更  
res = NSetCellEpW(fid, tableid, 2, 1, "1234");
```

関数概要

関数名		NGetTextEpW / NGetTextEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_txtid	I	PWCHAR / PCHAR	テキストID
o_dat	O	PWCHAR / PCHAR	テキストデータ格納エリア (NULL指定可)
o_size	O	PDWORD	テキストデータ格納エリアサイズ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_txtidを検索、対象のテキストコンポーネントのテキストデータを引数 o_datへ取得します。</div> <div>・引数 o_datをNULL指定することでテキストデータ格納に必要なバイト数(引数o_size)を取得します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";
PWCHAR txtid = L"Component1";
PWCHAR o_dat;
DWORD o_size = 0;
int res;

// 必要サイズ取得
res = NGetTextEpW(fid, txtid, NULL, &o_size);

// テキストデータ取得
o_dat = (PWCHAR)malloc((o_size + 1)*sizeof(WCHAR));
memset(o_dat, 0x00, (o_size + 1)*sizeof(WCHAR));
res = NGetTextEpW(fid, txtid, o_dat, &o_size);

free(o_dat);
```

関数概要

関数名		NGetBarcodeEpW / NGetBarcodeEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_barid	I	PWCHAR / PCHAR	バーコードID
o_dat	O	PBYTE	バーコードデータ格納エリア (NULL指定可)
o_size	O	PDWORD	バーコードデータ格納エリアサイズ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_baridを検索、対象のバーコードコンポーネントのバーコードデータを引数 o_datへ取得します。</div> <div>・引数 o_datをNULL指定することでバーコードデータ格納に必要なバイト数(引数o_size)を取得します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";
PWCHAR barid = L"Component1";
PBYTE o_dat;
DWORD o_size = 0;
int res;

//バーコードデータ格納エリアサイズ取得
res = NGetBarcodeEpW (fid, barid, NULL, &o_size) ;

//バーコードデータエリア確保
o_dat = (PBYTE)malloc((o_size+1)*sizeof(BYTE));
memset(o_dat, 0x00, (o_size+1)*sizeof(BYTE));

//バーコードデータ取得
res = NGetBarcodeEpW (fid, barid, o_dat, &o_size);

free(o_dat);
```

関数概要

関数名		NGetImageEpW / NGetImageEp	
引数名	IN/OUT	型	説明
i_fid	I	PWCHAR / PCHAR	フォーマットID
i_imgid	I	PWCHAR / PCHAR	イメージID
o_hdc	O	HDC	デバイスコンテキストハンドル
o_w	O	PDWORD	イメージ幅
o_h	O	PDWORD	イメージ高さ
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_imgidを検索、対象のイメージコンポーネントのイメージデータをデバイスコンテキスト、幅、高さに変換して引数 o_hdc、o_w、o_hへ取得します。</div>		
備考			

使用例

```
PWCHAR fid = L"Format1";
PWCHAR imgid = L"Component1";
DWORD width = 0;
DWORD height = 0;
int res;
HDC hdc;
HBITMAP hbitmap;
LPDWORD lpPixel;    // ビットマップピクセルへのポインタ
BITMAPINFO bmpInfo; // DIB形式の画像情報

// DIBの情報を設定する
bmpInfo.bmiHeader.biSize = sizeof(BITMAPINFOHEADER); // BITMAPINFO構造体サイズ
bmpInfo.bmiHeader.biWidth = 200;                    // 画像の横幅
bmpInfo.bmiHeader.biHeight = 200;                   // 画像の縦幅
bmpInfo.bmiHeader.biPlanes = 1;                     // プレーン数
bmpInfo.bmiHeader.biBitCount = 24;                  // ビット数
bmpInfo.bmiHeader.biCompression = BI_RGB; // 圧縮コード

// ビットマップハンドル作成
hbitmap = CreateDIBSection(NULL, &bmpInfo, DIB_RGB_COLORS, (void**)&lpPixel, NULL, 0);
// デバイスコンテキスト作成
hdc = CreateCompatibleDC(NULL);

// イメージコンポーネントから画像取得
res = NGetImageEpW (fid, imgid, hdc, &width, &height);
```

関数概要

関数名		NGetCellEpW / NGetCellEp		
引数名		IN/OUT	型	説明
i_fid		I	PWCHAR / PCHAR	フォーマットID
i_tableid		I	PWCHAR / PCHAR	テーブルID
i_col		I	INT	列
i_row		I	INT	行
o_TextContent		O	PCHAR	テキストデータ(NULL指定可)
o_TCnt		O	PINT	テキストデータサイズ
戻り値	INT			
・エラー(-値)、警告(+値)、正常終了(0)				
処理内容	<div>・引数 i_fid(フォーマットID)が指定されている場合、対象のフォーマット名が存在するか確認します。存在しなければエラーとなります。</div> <div>引数 i_fid(フォーマットID)がNULLもしくは空白の場合、フォーマットの確認は行いません。</div> <div>・コンポーネント一覧から引数 i_tableidを検索、対象のテーブルコンポーネント内の指定した引数 i_col(列)、i_row(行)に該当するセルのテキストデータを引数 o_TextContento_TCnt へ取得します。</div> <div>・引数 o_TextContentをNULL指定することでテキストデータ格納に必要なバイト数(引数o_TCnt)を取得します。</div>			
備考				

使用例

```
PWCHAR fid = L"Format1";
PWCHAR tableid = L"Component1";
PBYTE o_text;
INT o_size = 0;
int res;

//テーブル1列1行目のセルデータ格納エリアサイズ取得
res = NGetCellEpW (fid, tableid, 1, 1, NULL, &o_size) ;

//セルデータエリア確保
o_text = (PCHAR)malloc((o_size + 1) * sizeof(CHAR));
memset(o_text, 0x00, (o_size + 1) * sizeof(CHAR));

//テーブル1列1行目のセルデータ取得
res = NGetCellEpW (fid, tableid, 1, 1, o_text, &o_size) ;

free(o_text);
```

関数概要

関数名		NSaveDataEpW / NSaveDataEp	
引数名	IN/OUT	型	説明
i_type	I	INT	XML形式(0:EasyPrint用 1:プリンタXML用)
i_fname	I	PWCHAR / PCHAR	保存先ファイルパス・ファイル名
戻り値	INT		
・エラー(-値)、警告(+値)、正常終了(0)			
処理内容	<div>・引数 i_type が 0 の場合 EasyPrint用にXML形式(<npixml>・<request>・<print>タグ等)で作成されます。</div> <div>引数 i_type が 1 の場合 XML対応プリンター印字用のXML形式 (<npixml>・<request>・<template>タグなし <print>タグあり)で作成されます。</div> <div>・引数 i_fname(保存先ファイルパス・ファイル名)へXMLファイル出力します。 ・保存先にファイルがない場合は、新規作成します。 ・保存先にファイルがある場合は、上書きをします。</div>		
<div><div>参考</div>※XML仕様、説明は「XMLリファレンス」を参照</div>			
備考			

使用例

```
PWCHAR strFileName;  
int res;  
  
// 保存用ダイアログ表示  
if(SaveDialog1->Execute())  
{  
    // 保存先フルパスとファイル名  
    strFileName = SaveDialog1->FileName;  
  
    // EasyPrint用XMLをファイル出力  
    res = NSaveDataEpW(0, strFileName);  
}
```

エラーコード一覧(1/2)

NPIGFの使用しているエラーコードの一覧になります。

定義名	定義値	説明
N_SUCCESS	0	正常
N_WRN_STATUS	300	ステータス監視実行中
N_ERR_ARGUMENT	-301	NULL無効
N_ERR_ARGUMENT_FILE	-302	ファイル指定無
N_ERR_OUTPUT	-303	出力失敗
N_ERR_STATUS	-304	ステータスエラー
N_ERR_PFD_NOT_OPEN	-320	PFDファイル未展開
N_ERR_PRINTER_NOT_REGISTER	-321	Printer OR Format情報未登録
N_ERR_PRINTER_ID	-322	Printer IDエラー
N_ERR_FORMAT_ID	-323	Format IDエラー
N_ERR_READ_PRINTER_INFO	-324	Printer情報読込失敗
N_ERR_READ_FORMAT_INFO	-325	Format情報読込失敗
N_ERR_TEXT_NOT_REGISTER	-326	Text情報未登録 OR Text IDエラー
N_ERR_SET_COMPONENT	-329	コンポーネント情報設定失敗
N_ERR_LIBRARY_LOAD	-340	DLL Loadエラー
N_ERR_READ_FILE	-341	ファイル読込失敗
N_ERR_RESOURCE	-343	リソース不足
N_ERR_USB	-344	USB認証失敗
N_ERR_WININET	-345	WININET初期化失敗
N_ERR_URL_OPEN	-346	URLオープン失敗
N_ERR_DEFAULTDATA	-350	デフォルトデータ作成失敗
N_ERR_STARTTHREAD	-351	スレッド作成失敗
N_ERR_STOPTHREAD	-352	スレッド停止失敗
N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
N_ERR_NOTKIND	-354	コンポーネント種別エラー
N_ERR_CONVERT	-355	データ変換エラー
N_ERR_SIZE	-356	サイズエラー
N_ERR_DATA	-357	データ不良エラー
N_ERR_BUFFERSIZE	-358	コマンドバッファサイズエラー
N_ERR_ADD_COMPONENT	-359	コンポーネント追加エラー
N_ERR_ATTR_ANALYSIS	-360	属性情報解析エラー
N_ERR_NAME_DUPLICATE	-361	コンポーネント名重複エラー
N_ERR_NOTLAYOUTORDER	-362	表示順序不明エラー
N_ERR_CELL_POSITION	-363	セル位置不正エラー
N_ERR_NOT_GET_STATUS	-364	ステータス取得対象外
N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー

エラーコード一覧(2/2)

N_WRN_ALREADY_PRTOPEN	10	既に接続されています
N_ERR_HANDLE	-1	ハンドルエラー
N_ERR_PRTOPEN	-2	プリンターオープンエラー
N_ERR_SEND_ERROR	-3	送信エラー発生中
N_ERR_OFFLINE	-5	オフライン
N_ERR_PRTCLOSE	-6	プリンタークローズエラー
N_ERR_STATUSTIMEOUT	-7	ステータス取得時にタイムアウトしました
N_ERR_FILEOPEN	-10	ファイルオープンエラー
N_ERR_NOT_MAPPING	-11	拡張情報取得エラー
N_ERR_NOT_OPEN_MAPFILE	-12	マップファイルオープンエラー
N_ERR_PRTOUTPUT	-13	プリンター出力エラー
N_ERR_PRTILLEGAL	-16	接続中のプリンター情報が不正です
N_ERR_NONE_PRTLIST	-21	使用できるプリンターが存在しません
N_ERR_NOHANDLE	-22	プリンターがオープンされていない
N_ERR_LACKRESOURCE	-31	リソース不足
N_ERR_LOADFROMFILE	-50	画像ファイルの読み込みに失敗した
N_ERR_IMAGESIZE	-51	イメージサイズ不正
N_ERR_DOCNOTSTARTED	-71	ドキュメント開始状態でない
N_ERR_ALREADYSTARTDOC	-72	既にドキュメント開始状態となっている
N_ERR_ARGUMENT	-90	引数不正値エラー
N_ERR_ARGUMENT_01	-91	引数の1番目が不正
N_ERR_ARGUMENT_02	-92	引数の2番目が不正
N_ERR_ARGUMENT_03	-93	引数の3番目が不正
N_ERR_UDPTHREADSTOPPED	-112	UDPスレッドが起動していません
N_ERR_PRTINFO_CREATE	-130	プリンター情報ファイルの作成に失敗しました
N_ERR_PRTINFO_READ	-131	プリンター情報ファイルの読み込みに失敗しました
N_ERR_PRTINFO_WRITE	-132	プリンター情報ファイルの書き込みに失敗しました
N_ERR_PRTNAME_ALLOC	-133	プリンター名の割り当てに失敗しました
N_ERR_PRTINFO_GET	-136	プリンター情報が取得できませんでした
N_ERR_PRTINFO_ILLEGAL	-137	プリンター情報ファイルが不正です
N_ERR_PRTINFO_NOTFOUND	-139	プリンター名が存在しませんでした
N_ERR_DEVICE_NOTSUPPORT	-150	接続種別がサポートされていません

関数別エラーコード一覧(1/5)

関数名	定義名	定義値	備考
NReadInfoEpW NReadInfoEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_READ_FILE	-341	ファイル読込失敗
	N_ERR_ATTR_ANALYSIS	-360	属性情報解析エラー
NImagePrintEpW NImagePrintEp NImagePrintEpW_DX NImagePrintEp_DX	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_READ_PRINTER_INFO	-324	プリンタ情報取得エラー
	N_ERR_PRINTER_ID	-322	Printer IDエラー
	N_ERR_DEFAULTDATA	-350	デフォルトデータ作成失敗
	N_WRN_PRTALREADYOPEN	10	関数戻り値
	N_ERR_HANDLE	-1	関数戻り値
	N_ERR_PRTOPEN	-2	関数戻り値
	N_ERR_FILEOPEN	-10	関数戻り値
	N_ERR_PRTOUTPUT	-13	関数戻り値
	N_ERR_NONE_PRTLST	-21	関数戻り値
	N_ERR_NOHANDLE	-22	関数戻り値
	N_ERR_LACKRESOURCE	-31	関数戻り値
	N_ERR_DOCNOTSTARTED	-71	関数戻り値
	N_ERR_ALREADYSTARTDOC	-72	関数戻り値
	N_ERR_ARGUMENT_01	-91	関数戻り値
	N_ERR_ARGUMENT_02	-92	関数戻り値
	N_ERR_ARGUMENT_03	-93	関数戻り値
	N_ERR_UDPTHREADSTOPPED	-112	関数戻り値
	N_ERR_PRTINFO_NOTFOUND	-139	関数戻り値
	N_ERR_DEVICE_NOTSUPPORT	-150	関数戻り値
NPrintEpW NPrintEp NPrintEpW_DX NPrintEp_DX	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_PRINTER_ID	-322	Printer IDエラー
	N_ERR_HANDLE	-1	関数戻り値
	N_ERR_FILEOPEN	-10	関数戻り値
	N_ERR_PRTOUTPUT	-13	関数戻り値
	N_ERR_NONE_PRTLST	-21	関数戻り値
	N_ERR_NOHANDLE	-22	関数戻り値
	N_ERR_LACKRESOURCE	-31	関数戻り値
	N_ERR_LOADFROMFILE	-50	関数戻り値
	N_ERR_IMAGESIZE	-51	関数戻り値
	N_ERR_ARGUMENT	-90	関数戻り値
	N_ERR_ARGUMENT_01	-91	関数戻り値
	N_ERR_ARGUMENT_02	-92	関数戻り値
	N_ERR_ARGUMENT_03	-93	関数戻り値

関数別エラーコード一覧(2/5)

関数名	定義名	定義値	備考
NPrinterInitEpW NPrinterInitEp	N_SUCCESS	0	正常
	N_WRN_STATUS	300	ステータス監視実行中
	N_ERR_STATUS	-304	ステータスエラー
	N_ERR_READ_PRINTER_INFO	-324	プリンタ情報取得エラー
	N_ERR_STARTTHREAD	-351	スレッド作成失敗
	N_ERR_STOPTHREAD	-352	スレッド停止失敗
	N_ERR_NOT_GET_STATUS	-364	ステータス取得対象外
	N_WRN_PRTALREADYOPEN	1	関数戻り値
	N_WRN_ALREADY_PRTOPEN	10	関数戻り値
	N_ERR_HANDLE	-1	関数戻り値
	N_ERR_PRTOPEN	-2	関数戻り値
	N_ERR_SEND_ERROR	-3	関数戻り値
	N_ERR_OFFLINE	-5	関数戻り値
	N_ERR_PRTCLOSE	-6	関数戻り値
	N_ERR_STATUSTIMEOUT	-7	関数戻り値
	N_ERR_FILEOPEN	-10	関数戻り値
	N_ERR_NOT_MAPPING	-11	関数戻り値
	N_ERR_NOT_OPEN_MAPFILE	-12	関数戻り値
	N_ERR_PRTOUTPUT	-13	関数戻り値
	N_ERR_PRTILLEGAL	-16	関数戻り値
	N_ERR_NONE_PRTLST	-21	関数戻り値
	N_ERR_NOHANDLE	-22	関数戻り値
	N_ERR_LACKRESOURCE	-31	関数戻り値
	N_ERR_LOADFROMFILE	-50	関数戻り値
	N_ERR_IMAGESIZE	-51	関数戻り値
	N_ERR_ARGUMENT	-90	関数戻り値
	N_ERR_ARGUMENT_01	-91	関数戻り値
	N_ERR_ARGUMENT_02	-92	関数戻り値
	N_ERR_ARGUMENT_03	-93	関数戻り値
	N_ERR_UDPTHREADSTOPPED	-112	関数戻り値
	N_ERR_PRTINFO_CREATE	-130	関数戻り値
	N_ERR_PRTINFO_READ	-131	関数戻り値
	N_ERR_PRTINFO_WRITE	-132	関数戻り値
	N_ERR_PRTNAME_ALLOC	-133	関数戻り値
	N_ERR_PRTINFO_GET	-136	関数戻り値
	N_ERR_PRTINFO_ILLEGAL	-137	関数戻り値
	N_ERR_PRTINFO_NOTFOUND	-139	関数戻り値
	N_ERR_DEVICE_NOTSUPPORT	-150	関数戻り値

関数別エラーコード一覧(3/5)

関数名	定義名	定義値	備考
NGetStEpW	N_SUCCESS	0	正常
NGetStEp	N_ERR_ARGUMENT	-301	NULL無効
NGetStEpW_DX	N_ERR_PRINTER_ID	-322	Printer IDエラー
NGetStEp_DX	N_ERR_NOT_GET_STATUS	-364	ステータス取得対象外
	N_ERR_SEND_ERROR	-3	関数戻り値
	N_ERR_OFFLINE	-5	関数戻り値
	N_ERR_NOT_MAPPING	-11	関数戻り値
	N_ERR_NOT_OPEN_MAPFILE	-12	関数戻り値
	N_ERR_PRTOUTPUT	-13	関数戻り値
	N_ERR_LACKRESOURCE	-31	関数戻り値
	N_ERR_ARGUMENT_01	-91	関数戻り値
	N_ERR_ARGUMENT_02	-92	関数戻り値
NGetInfoEpW	N_SUCCESS	0	正常
NGetInfoEp	N_ERR_ARGUMENT	-301	NULL無効
NGetInfoEpW_DX	N_ERR_PRINTER_ID	-322	Printer IDエラー
NGetInfoEp_DX	N_ERR_HANDLE	-1	関数戻り値
	N_ERR_FILEOPEN	-10	関数戻り値
	N_ERR_NOT_MAPPING	-11	関数戻り値
	N_ERR_NOT_OPEN_MAPFILE	-12	関数戻り値
	N_ERR_PRTOUTPUT	-13	関数戻り値
	N_ERR_NONE_PRTLIST	-21	関数戻り値
	N_ERR_NOHANDLE	-22	関数戻り値
	N_ERR_LACKRESOURCE	-31	関数戻り値
	N_ERR_LOADFROMFILE	-50	関数戻り値
	N_ERR_IMAGESIZE	-51	関数戻り値
	N_ERR_ARGUMENT	-90	関数戻り値
	N_ERR_ARGUMENT_01	-91	関数戻り値
	N_ERR_ARGUMENT_02	-92	関数戻り値
	N_ERR_ARGUMENT_03	-93	関数戻り値
NStartOnChangeStW	N_SUCCESS	0	正常
NStartOnChangeSt	N_WRN_STATUS	300	ステータス監視実行中
NStartOnChangeStW_DX	N_ERR_ARGUMENT	-301	NULL無効
NStartOnChangeSt_DX	N_ERR_PRINTER_ID	-322	Printer IDエラー
	N_ERR_STARTTHREAD	-351	スレッド作成失敗
	N_ERR_STOPTHREAD	-352	スレッド停止失敗
NEndOnChangeSt	N_SUCCESS	0	正常
	N_ERR_STOPTHREAD	-352	スレッド停止失敗

関数別エラーコード一覧(4/5)

関数名	定義名	定義値	備考
NSSetTextEpW NSSetTextEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_CONVERT	-355	データ変換エラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー
NSSetBarcodeEpW NSSetBarcodeEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_CONVERT	-355	データ変換エラー
	N_ERR_SIZE	-356	サイズエラー
NSSetImageEpW NSSetImageEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_SIZE	-356	サイズエラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー
NSSetCellEpW NSSetCellEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_CONVERT	-355	データ変換エラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー

関数別エラーコード一覧(5/5)

関数名	定義名	定義値	備考
NGetTextEpW NGetTextEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_CONVERT	-355	データ変換エラー
	N_ERR_DATA	-357	データ不良エラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー
NGetBarcodeEpW NGetBarcodeEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_CONVERT	-355	データ変換エラー
	N_ERR_DATA	-357	データ不良エラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー
NGetImageEpW NGetImageEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー
NGetCellEpW NGetCellEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_NOTFINDNAME	-353	コンポーネント名不明エラー
	N_ERR_NOTKIND	-354	コンポーネント種別エラー
	N_ERR_CONVERT	-355	データ変換エラー
	N_ERR_DATA	-357	データ不良エラー
	N_ERR_NOT_FIND_FORMAT	-365	フォーマット存在なしエラー
NSaveDataEpW NSaveDataEp	N_SUCCESS	0	正常
	N_ERR_ARGUMENT	-301	NULL無効
	N_ERR_BUFFERSIZE	-358	コマンドバッファサイズエラー