Syuhitu.org

主筆　The text editor for Solaris Plugin Development Guide

主筆　第20版　プラグイン開発ガイド

目次

1 このマニュアルについて

1.1 概要

このマニュアルは、Sun Solaris 用テキストエディタ主筆で使用する、プラグインを開発する上での規約・API について記述しています。

1.2 対象となる読者

このマニュアルは、主筆で使用するプラグインの開発者を対象としています。

1.3 前提知識

このマニュアルでは、読者は下記の事項に関する一般的な知識を有することを前提としています。
- Unix 上の C 言語によるソフトウェア開発一般
- Unix におけるシステム管理一般
- 主筆の操作・管理一般

1.4 関連ドキュメント

必要に応じて、下記のマニュアルを参照してください。
- 主筆　ユーザーズガイド
- Syuhitu User's guide

1.5 商標などについて

このマニュアルに記載されている会社名、商品名、製品名などは、一般に各社の商標または登録商標です。

1.6 連絡先

原作者及び一次配布元は下記の通りです。

メールアドレス	nabiki_t@syuhitu.org
著作物の所在	http://www.syuhitu.org/

2 規約

2.1 形式

主筆のプラグインは、動的共有ライブラリとして作成されます。一般に拡張子は so となります。

Forte C++ もしくは Sun Studio を使用する場合は、コンパイル時に引数「-G」を指定してください。

2.2 ヘッダファイル

主筆のプラグインを作成するために必要となる宣言は「PluginFuncID.h」に記述されています。

2.3 公開する関数

主筆のプラグインは下記の形式の関数を公開しなければなりません。

```
void 関数名( PFT_GetAPIFunction pGetAPIFunction );
```

「関数名」はプラグイン設定ファイルの FunctionName により指定されます。主筆はメニューが選択されると、指定されたライブラリをロードし、指定された名前の関数を呼び出します。

この呼ばれた関数からリターンすると、プラグインの処理は終了します。

引数 pGetAPIFunction には下記の形式の関数のアドレスが渡されます。

```
void* pGetAPIFunction( int FunctionType );
```

この関数は、主筆が公開する API のアドレスを返します。取得したい関数の種別を示す定数を FunctionType に指定してください。

関数の種別を示す定数については、3.2API 関数を参照してください。

2.4 グローバル変数

グローバル変数は使用しないでください。

プラグインのライブラリは、呼び出しが終了する都度メモリ上から削除されます（正確には、削除される可能性がある状態になります）。複数回の呼び出しにまたがって共有の値を参照したい場合には、PFID_ALLOCWORKMEMORY の API 関数を用いて、作業用メモリ領域を確保してください。

3 API

プラグイン関数は、主筆から引数により渡された関数ポインタにより、主筆が公開する API の関数ポインタを取得することができます。

3.1 データ型

「PluginFuncID.h」には、各関数ポインタの型と関数の種別を示す定数以外に、下記のデータ型が定義されています。

型名	概要
PFT_BOOL	主筆プラグインで使用される真偽値型です。
PFT_ENCODEINFO	エンコードの情報を保持する構造体です。
PFT_ENCODEDLGINFO	PFID_SHOWENCODEDIALOG 関数で使用される構造体です。エンコード選択ダイアログに表示する情報、および、ユーザにより入力された情報を保持します。
PFT_FILEDLGINFO	PFID_SHOWFILEDIALOG 関数で使用される構造体です。ファイルを開く / 名前を付けて保存ダイアログに表示する情報、および、ユーザにより入力された情報を保持します。

3.1.1 PFT_BOOL

主筆プラグインで使用される真偽値型です。

```
#define PFT_BOOL unsigned char
```

概要

主筆プラグイン内で使用される真偽値型です。下記の値が使用されます。

PFT_TRUE	真を表す定数です。
PFT_FALSE	偽を表す定数です。

互換性

第 9 版以降

3.1.2 PFT_ENCODEINFO

エンコードの情報を保持する構造体です。

```
typedef struct tagPFTENCODEDLGINFO
{
          size_t Size;
          int Type;
          char *pEncodeName;
} PFT_ENCODEINFO;
```

概要

エンコード名および種別を保持する構造体です。

メンバ変数

Size

PFT_ENCODEINFO 構造体のサイズ (バイト数) を指定します。通常は sizeof (PFT_ENCODEINFO)の値を指定してください。

Type

エンコードの種別を指定します。下記の値のうちいずれか一つを指定することができます。

値	概要
PFT_ENCODE_TYPE_NON	エンコード名を明示的に指定しないことを表します。この値が指定された場合、エンコード名はリソースの enptyEncodeNameBehavior および encodeRecognizeFailedBehavior の指定により決定されます。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。
PFT_ENCODE_TYPE_SPECIFY	pEncodeName に指定したエンコード名を使用することを表します。
PFT_ENCODE_TYPE_CURRENT	カレントのロケールで使用されるエンコードを使用することを表します。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。
PFT_ENCODE_TYPE_AUTO	エンコードの自動認識を行うことを表します。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。

pEncodeName

エンコード名を指定します。なお、pEncodeName の値は、Type に PFT_ENCODE_TYPE_SPECIFY 以外の値が指定されていた場合には無視されます。

互換性

第 20 版以降

3.1.3 PFT_ENCODEDLGINFO

PFID_SHOWENCODEDIALOG の API 関数で使用される、エンコード選択ダイアログの情報を保持する構造体です。

```
typedef struct PFTENCODEDLGINFO
{
        size_t Size;
        const char *pDlgMessage;
        PFT_ENCODEINFO InitEncode;
        unsigned int Flg;
        PFT_BOOL Result;
        PFT_BOOL Padding1[3];
        PFT_ENCODEINFO Encode;
} PFT_ENCODEDLGINFO;
```

概要

PFID_SHOWENCODEDIALOG の API 関数により表示される、エンコード選択ダイアログに表示する情報、および、表示されたダイアログボックスでユーザが入力した情報を保持する構造体です。

メンバ変数

Size

PFT_ENCODEDLGINFO 構造体のサイズ (バイト数) を指定してください。通常は sizeof(PFT_ENCODEDLGINFO)の値を指定してください。

pDlgMessage

エンコード選択ダイアログの上部に表示するメッセージを指定してください。NULL もしくは長さ 0 の文字列を指定した場合は、メッセージは表示されません。

InitEncode

エンコード選択ダイアログで、初期表示状態で選択状態としておくエンコードを指定してください。指定する内容については PFT_ENCODEINFO 構造体の説明を参照してください。

Flg

下記の値を指定します。なお、これらの値はビット OR 演算により複数個指定することができます。

値	説明
PFT_ENCDLG_SHOWCURRENT	選択可能なエンコードの中に、「カレントロケールのエンコード」を表示する。
PFT_ENCDLG_SHOWAUTO	選択可能なエンコードの中に、「自動認識」を表示する。

Result

PFID_SHOWENCODEDIALOG 関数から制御が返ると、下記の値が設定されます。

値	説明
PFT_TRUE	エンコード名が入力され、ユーザが「 O K 」ボタンを押下したことを表します。
PFT_FALSE	ユーザにより「キャンセル」ボタンを押下し、エンコード選択ダイアログが閉じられたことを表します。

Padding1

使用されません。

Encode

ユーザが「OK」ボタンを押下し、エンコード選択ダイアログが閉じられ、

PFID_SHOWENCODEDIALOG 関数から制御が返ると、ユーザが入力したエンコードの情報が設定されます。設定される値については、PFT_ENCODEINFO 構造体の説明を参照してください。

なお、Result の値が PFT_FALSE の場合、Encode には有効な値は設定されません。また、Encode.pEncodName の文字列は、使用し終わった段階で PFID_SYFREE 関数により解放してください。解放処理が実施されない場合は、メモリリークが発生します。

互換性

第 20 版以降

3.1.4 PFT_FILEDLGINFO

PFID_SHOWFILEDIALOG の API 関数で使用される、ファイルを開く / 名前を付けて保存ダイアログの情報を保持する構造体です。

```
typedef struct tagPFTFILEDLGINFO
{
            size_t Size;
            unsigned int Flg;
            const char *pInitFileName;
            const char *pFilter;
            PFT_ENCODEINFO InitEncode;
            int InitCRType;
            PFT_BOOL Result;
            PFT_BOOL Padding2[3];
            PFT_ENCODEINFO Encode;
            char *pFileName;
            int CRType;
} PFT_FILEDLGINFO;
```

概要

PFID_SHOWFILEDIALOG の API 関数により表示される、ファイルを開く / 名前を付けて保存ダイアログに表示する情報、および、表示されたダイアログボックスでユーザが入力した情報を保持構造体です。

メンバ変数

Size

PFT_FILEDLGINFO 構造体のサイズ (バイト数) を指定してください。通常は sizeof(PFT_FILEDLGINFO)の値を指定してください。

Flg

表示するダイアログについてのオプションを指定します。下記の値を指定することができます。なお、これらの値はビット OR 演算により複数個指定することができます。

値	説明
PFT_FILEDLG_OPENDLG	「ファイルを開く」ダイアログを表示します。Flg にこの値を指定しなかった場合は「名前を付けて保存」ダイアログが表示されます。
PFT_FILEDLG_ENABLE_CRSELECT	改行コードの指定を有効にします。
PFT_FILEDLG_ENABLE_ENCODESELECT	エンコードの指定を有効にします。

pInitFileName

ダイアログ表示時に選択状態とするファイルを指定します。pInitiFileName に NULL もしくは長さ 0 の文字列が指定された場合には、ファイルが選択されていない状態でダイアログが表示されます。

pFilter

ダイアログ表示時に設定するフィルタを指定します。pFilter に NULL もしくは長さ 0 の文字列が指定された場合には、フィルタには"*"が設定されます 。すなわち、ダイアログには全てのファイルが表示されることになります。

InitEncode

ダイアログ表示時に選択状態とするエンコードを指定します。設定する値については、PFT_ENCODEINFO 構造体の説明を参照してください。

InitCRType

ダイアログ表示時に選択状態とする改行コードの種別を指定します。

改行コードの種別には、下記の値を指定することができます。

値	説明
PFT_CRTYPE_LF	改行コードとして"LF"が使用されることを表します。UNIX で標準的に使用される改行コードです。
PFT_CRTYPE_CRLF	改行コードとして"CRLF"が使用されることを表します。Windows で標準的に使用される改行コードです。
PFT_CRTYPE_CR	改行コードとして"CR"が使用されることを表します。
PFT_CRTYPE_LFCR	改行コードとして"LFCR"が使用されることを表します。

Result

PFID_SHOWFILEDIALOG 関数から制御が返ると、下記の値が設定されます。

値	説明
PFT_TRUE	ファイルが選択され、ユーザが「 O K 」ボタンを押下したことを表します。
PFT_FALSE	ユーザにより「キャンセル」ボタンを押下し、ファイルを開く / 名前を付けて保存ダイアログが閉じられたことを表します。

Padding2

使用されません。

Encode

ユーザが「OK」ボタンを押下し、ファイルを開く / 名前を付けて保存ダイアログが閉じられ、PFID_SHOWFILEDIALOG 関数から制御が返ると、ユーザが入力したエンコードの情報が設定されます。設定される値については、PFT_ENCODEINFO 構造体の説明を参照してください。

なお、Result の値が PFT_FALSE の場合、もしくは Flg に PFT_FILEDLG_ENABLE_ENCODESELECT が設定されなかった場合には、Encode には有効な値は設定されません。また、Encode.pEncodName の文字列は、使用し終わった段階で PFID_SYFREE 関数により解放してください。解放処理が実施されない場合は、メモリリークが発生します。

pFileName

ユーザが「OK」ボタンを押下し、ファイルを開く / 名前を付けて保存ダイアログが閉じられ、PFID_SHOWFILEDIALOG 関数から制御が返ると、ユーザが選択したファイルのファイル名が設定されます。

なお、 Result の値が PFT_FALSE の場合には、pFileName には有効な値は設定されません。また、pFileName の文字列は、使用し終わった段階で PFID_SYFREE 関数により解放してください。解放処理が実施されない場合は、メモリリークが発生します。

CRType

ユーザが「OK」ボタンを押下し、ファイルを開く / 名前を付けて保存ダイアログが閉じられ、PFID_SHOWFILEDIALOG 関数から制御が返ると、ユーザが選択した改行コードの種別が設定されます。設定される値は、InitCRType の説明を参照してください。

なお、 Result の値が PFT_FALSE の場合には、CRType には有効な値は設定されません。

互換性

第 20 版以降

3.2 ＡＰＩ関数

主筆は、プラグインに対し下記の API 関数を提供しています。

関数の種別	型	概要
PFID_GETCHAR	PFT_GetChar	文字を取得します。
PFID_SETCHAR	PFT_SetChar	文字を設定します。
PFID_GETSTRING	PFT_GetString	文字列を取得します。
PFID_REPLACE	PFT_Replace	文字列を置換します。
PFID_GETLINECOUNT	PFT_GetLineCount	行数を取得します。
PFID_GETCHARCOUNT	PFT_GetCharCount	指定した行に含まれる文字数を取得します。
PFID_GETCURSORPOSITION	PFT_GetCurPosition	カーソルの位置を取得します。
PFID_SETCURSORPOSITION	PFT_SetCurPosition	カーソルの位置を設定します。
PFID_GETSELECTIONRANGE	PFT_GetSelectionRange	選択範囲を取得します。
PFID_SETSELECTIONRANGE	PFT_SetSelectionRange	選択範囲を設定します。
PFID_GETCONFIGVALUE	PFT_GetConfigValue	プラグイン設定ファイルの設定値を取得します。
PFID_GETFILENAME	PFT_GetFileName	ファイル名を取得します。
PFID_SHOWINFORMATIONMSGBOX	PFT_ShowInformationMsgBox	情報メッセージボックスを表示します。
PFID_SHOWQUESTIONMSGBOX	PFT_ShowQuestionMsgBox	質問メッセージボックスを表示します。
PFID_SHOWERRORMSGBOX	PFT_ShowErrorMsgBox	エラーメッセージボックスを表示します。
PFID_GETMODIFIEDFLG	PFT_GetModifiedFlg	更新フラグを取得します。
PFID_GETVERSION	PFT_GetVersion	主筆のバージョンを取得します。
PFID_GETEXTENDINFOCOUNT	PFT_GetExtendInfoCount	利用可能な拡張情報領域の数を取得します。
PFID_GETEXTENDINFO	PTF_GetExtendInfo	拡張情報領域の値を取得します。
PFID_SETEXTENDINFO	PTF_SetExtendInfo	拡張情報領域に値を設定します。
PFID_SAVEFILE	PFT_SaveFile	ファイルを保存します。
PFID_OPENFILE	PFT_OpenFile	ファイルを開きます。
PFID_OPENFILENEWWINDOW	PFT_OpenFileNewWindow	新しいウインドウでファイルを開きます。
PFID_ALLOCWORKMEMORY	PFT_AllocWorkMemory	新規に作業用メモリ領域を確保します。
PFID_FINDWORKMEMORY	PFT_FindWorkMemory	既存の作業用メモリ領域を検索します。
PFID_FREEWORKMEMORY	PFT_FreeWorkMemory	既存の作業用メモリ領域を開放します。
PFID_ISCONSTRUCTIONMODE	PFT_IsConstructionMode	コンストラクション・モードか否かを取得します。
PFID_SETCONSTRUCTION	PFT_SetConstruction	コンストラクション・モードに設定します。
PFID_SETSTANDALONE	PFT_SetStandalone	スタンドアロン・モードに設定します。
PFID_GETENCODENAME	PFT_GetEncodeName	ファイルのエンコード名を取得します。
PFID_GETCRTYPE	PFT_GetCRType	ファイルの改行コードの種別を取得します。
PFID_SAVEFILE16	PFT_SaveFile16	ファイルを保存します。
PFID_OPENFILE16	PFT_OpenFile16	ファイルを開きます。
PFID_OPENFILE20	PFT_OpenFile20	ファイルを開きます。
PFID_SYMALLOC	PFT_SyMalloc	メモリ領域を確保します。
PFID_SYCALLOC	PFT_SyCalloc	メモリ領域を確保し、0 クリアします。
PFID_SYREALLOC	PFT_SyRealloc	確保したメモリ領域のサイズを変更します。
PFID_SYFREE	PFT_SyFree	確保したメモリ領域を開放します。

関数の種別	型	概要
PFID_SHOWENCODEDIALOG	PFT_ShowEncodeDialog	エンコードの選択ダイアログを表示し、ユーザにエンコード名の入力を促します。
PFID_SHOWFILEDIALOG	PFT_ShowFileDialog	ファイルを開く / 名前を付けて保存ダイアログを表示し、ユーザにファイル名の入力を促します。

3.2.1 PFID_GETCHAR

指定した位置の文字を取得します。

```
PFT_BOOL (*PFT_GetChar)(
        unsigned long line,
        unsigned long cpos,
        wchar_t* pChar
);
```

型

PFT_GetChar

概要

line 行目 cpos 文字目の文字を*pChar に取得します。

引数

line

行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

cpos

line 行内の何番目の文字を取得するか、を指定します。指定可能な値は 0 から line 行目に存在する文字数-1 までです。

pChar

文字を取得するための変数のアドレスを指定します。

戻り値

文字の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.2 PFID_SETCHAR

指定した位置に文字を設定します。

```
PFT_BOOL (*PFT_SetChar)(
        unsigned long line,
        unsigned long cpos,
        wchar_t c
);
```

型

PFT_SetChar

概要

line 行目 cpos 文字目に文字 c を設定します。

引数

line

行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

cpos

line 行内の何番目の文字を取得するか、を指定します。指定可能な値は 0 から line 行目に存在する文字数-1 までです。

c

設定する文字を指定します。

戻り値

文字の設定に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.3 PFID_GETSTRING

指定した範囲の文字列を取得します。

```
PFT_BOOL (*PFT_GetString)(
        unsigned long SLP,
        unsigned long SCP,
        unsigned long ELP,
        unsigned long ECP,
        wchar_t* pBuf,
        unsigned long BufLength
);
```

型

PFT_GetString

概要

SLP 行目 SCP 文字目から ELP 行目 ECP-1 文字目までの文字列を pBuf が示すアドレスに取得します。

文字列の末尾には'\0'が設定されます。

もし、取得される文字列が BufLength より長かった場合には、バッファに格納することができた位置までの文字列が返されます。

引数

SLP

文字列の取得を開始する行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

SCP

SLP 行内の何番目の文字から取得するか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数-1 までです。

ELP

文字列の取得を終了する行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

ECP

ELP 行内の何番目の文字まで取得するか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数までです。 なお、。取得される文字列には ELP 行目 ECP 文字目の文字は含まれません。

pBuf

文字列を取得するためのメモリ領域のアドレスを指定します。

BufLength

pBuf に用意されたバッファの長さ (文字数) を指定します。

戻り値

文字列の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.4 PFID_REPLACE

指定した範囲の文字列を置換します。

```
PFT_BOOL (*PFT_Replace)(
        unsigned long SLP,
        unsigned long SCP,
        unsigned long ELP,
        unsigned long ECP,
        const wchar_t* pBuf
);
```

型

PFT_Replace

概要

SLP 行目 SCP 文字目から ELP 行目 ECP-1 文字目までの文字列を pBuf に指定された文字列と置換します。

pBuf に指定された、置換後の文字列の末尾には'\0'が設定されている必要があります。

引数

SLP

置換範囲の開始位置の行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

SCP

置換範囲が SLP 行内の何番目の文字から開始されるか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数-1 までです。

ELP

置換範囲の終了位置の行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

ECP

置換範囲が ELP 行内の何番目の文字で終了されるか、を指定します。指定可能な値は 0 から SLP 行目に存在する文字数までです。 なお、置換される文字列には ELP 行目 ECP 文字目の文字は含まれません。

pBuf

文字列を指定します。

戻り値

文字列の置換に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.5 PFID_GETLINECOUNT

行数を取得します。

```
PFT_BOOL (*PFT_GetLineCount)(
        unsigned long* pCnt
);
```

型

PFT_GetLineCount

概要

pCnt に指定した変数に行数を取得します。

引数

pCnt

行数を取得するための変数のアドレスを指定します。

戻り値

行数の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.6 PFID_GETCHARCOUNT

指定した行の文字数を取得します。

```
PFT_BOOL (*PFT_GetCharCount)(
        unsigned long LP,
        unsigned long* pCnt
);
```

型

PFT_GetCharCount

概要

LP 行目に含まれる文字の文字数を pCnt に指定した変数に取得します。

取得される文字数には、行の末尾に存在する改行コードも含みます。ただし、一番最後の行には改行コードは存在しません。

引数

LP

文字数を取得する行の行番号を指定します。指定可能な行番号は 0 から総行数-1 までです。

pCnt

文字数を取得するための変数のアドレスを指定します。

戻り値

文字数の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.7 PFID_GETCURSORPOSITION

現在のカーソルの位置を取得します。

```
PFT_BOOL (*PFT_GetCursorPosition)(
        unsigned long* pLP,
        unsigned long* pCP
);
```

型

PFT_GetCursorPosition

概要

現在のカーソルの位置を取得します。

引数

pLP

カーソルが存在する位置の行番号を取得するための、変数のアドレスを指定します。

pCP

カーソルが存在する位置の文字位置を取得するための、変数のアドレスを指定します。

戻り値

カーソル位置の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.8 PFID_SETCURSORPOSITION

カーソルの位置を設定します。

```
PFT_BOOL (*PFT_SetCursorPosition)(
        unsigned long LP,
        unsigned long CP
);
```

型

PFT_SetCursorPosition

概要

カーソルの位置を設定します。

引数

LP

カーソルを設定する位置の行番号を指定します。指定可能な値は 0 から総行数-1 までです。

CP

カーソルを設定する位置の文字位置を指定します。指定可能な値は 0 から LP 行目に含まれる文字数-1 までです。

戻り値

カーソル位置の設定に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.9 PFID_GETSELECTIONRANGE

現在の選択範囲を取得します。

```
PFT_BOOL (*PFT_GetSelectionRange)(
        unsigned long* pSLP,
        unsigned long* pSCP,
        unsigned long* pELP,
        unsigned long* pECP
);
```

型

PFT_GetSelectionRange

概要

現在の選択範囲を取得します。

引数

pSLP

選択範囲の開始位置の行番号を取得するための、変数のアドレスを指定します。

pSCP

選択範囲の開始位置の文字位置を取得するための、変数のアドレスを指定します。

pELP

選択範囲の終了位置の行番号を取得するための、変数のアドレスを指定します。

pECP

選択範囲の終了位置の文字位置を取得するための、変数のアドレスを指定します。 なお、(*pELP)行目(*pECP) 文字目の文字は選択範囲に含まれません。

戻り値

選択範囲の取得に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第9版以降

3.2.10 PFID_SETSELECTIONRANGE

選択範囲を設定します。

```
PFT_BOOL (*PFT_SetSelectionRange)(
        unsigned long SLP,
        unsigned long SCP,
        unsigned long ELP,
        unsigned long ECP
);
```

型

PFT_SetSelectionRange

概要

選択範囲を設定します。

引数

SLP

選択範囲の開始位置の行番号を指定します。

SCP

選択範囲の開始位置の文字位置を指定します。

ELP

選択範囲の終了位置の行番号を指定します。

ECP

選択範囲の終了位置の文字位置を指定します。 なお、ELP 行目 ECP 文字目の文字は選択範囲に含まれません。

戻り値

選択範囲の設定に成功した場合は真が返されます。そうでない場合は偽が返されます。

互換性

第 9 版以降

3.2.11 PFID_GETCONFIGVALUE

プラグイン設定ファイルに記述された設定情報を取得します。

```
const wchar_t* (*PFT_GetConfigValue)(
        const wchar_t* pKey
);
```

型

PFT_GetConfigValue

概要

プラグイン設定ファイルに記述された設定情報を参照します。参照可能な値は、呼び出されたプラグインについて記述しているセクションに属する値のみです。

たとえば、プラグイン設定ファイルに下記のように記述されていて、

```
[MyPlugin1]
PluginName = MyPlugin1
LibraryName = libmyplugin.so
FunctionName = foo
MenuLabel = プラグイン 1
ConfigValue1 = あいうえお

[MyPlugin2]
PluginName = MyPlugin2
LibraryName = libmyplugin.so
FunctionName = foo
MenuLabel = プラグイン 2
ConfigValue2 = かきくけこ
```

ユーザが「プラグイン 1 」を選択した時に、 キーとして”ConfigValue1”を指定してこの関数を呼び出した場合は、「あいうえお」の値が返ってきます。 また、キーに”ConfigValue2”を指定した場合には NULL が返ってきます。 この例のように、呼び出しているライブラリと関数が同じであっても、別のセクションの値を参照することはできません。

引数

pKey

値を検索するためのキーを指定します。

戻り値

値の取得に成功した場合は、そのアドレスが返されます。そうでない場合は NULL が返されます。 なお、返される値は内部で保持している値のアドレスなので、プラグイン側では解放しないでください。

互換性

第 9 版以降

3.2.12 PFID_GETFILENAME

ファイル名を取得します。

```
const char* (*PFT_GetFileName)();
```

型

PFT_GetFileName

概要

現在開いているファイルのファイル名を取得します。 もしファイル名が未定ならば、長さ 0 の文字列が返されます。 NULL が返されることはありません。

戻り値

ファイル名の文字列のアドレスが返されます。この文字列は内部で保持している値なので、プラグイン側で解放しないでください。

互換性

第 9 版以降

3.2.13 PFID_SHOWINFORMATIONMSGBOX

情報メッセージボックスを表示します。

```
void (*PFT_ShowInformationMsgBox)(
        const wchar_t* pMsg
);
```

型

PFT_ShowInformationMsgBox

概要

モーダルな情報メッセージボックスを表示させます。 ユーザが「OK」ボタンを押下するまで制御が戻りません。

引数

pMsg

表示するメッセージを指定します。

互換性

第 10 版以降

3.2.14 PFID_SHOWQUESTIONMSGBOX

質問メッセージボックスを表示します。

```
int (*PFT_ShowQuestionMsgBox)(
        const wchar_t* pMsg,
        PFT_BOOL ShowCancel
);
```

型

PFT_ShowQuestionMsgBox

概要

モーダルな質問メッセージボックスを表示させます。ユーザが「はい」「いいえ」「キャンセル」のいずれかのボタンを押下するまで制御が戻りません。

引数

pMsg

表示するメッセージを指定します。

ShowCancel

真を設定した場合には「キャンセル」ボタンが表示されます。

戻り値

ユーザが押下したボタンに応じて、下記の値が返されます。

ボタン	値
はい	1
いいえ	2
キャンセル	3

互換性

第10版以降

3.2.15 PFID_SHOWERRORMSGBOX

エラーメッセージボックスを表示します。

```
void (*PFT_ShowErrorMsgBox)(
        const wchar_t* pMsg
);
```

型

PFT_ShowErrorMsgBox

概要

モーダルなエラーメッセージボックスを表示させます。 ユーザが「OK」ボタンを押下するまで制御が戻りません。

引数

pMsg

表示するメッセージを指定します。

互換性

第 10 版以降

3.2.16 PFID_GETMODIFIEDFLG

更新フラグを取得します。

```
PFT_BOOL (*PFT_GetModifiedFlg)();
```

型

PFT_GetModifiedFlg

概要

更新フラグを取得します。最後に更新されてから変更が加えられていた場合には真を返します。

戻り値

最後にファイルが保存されてから、何らかの変更が行われていた場合には真、最後にファイルが保存されてから変更が行われていなければ偽が返されます。

互換性

第 12 版以降

3.2.17 PFID_GETVERSION

主筆のバージョンを取得します。

```
int (*PFT_GetVersion)();
```

型

PFT_GetVersion

概要

プラグインを呼び出した主筆のバージョン番号を返します。例えば、主筆のバージョンが第 20 版であれば 20 という値が返されます。

戻り値

バージョン番号が返されます。

互換性

第 12 版以降

3.2.18 PFID_GETEXTENDINFOCOUNT

拡張情報領域の数を取得します。

```
int (*PFT_GetExtendInfoCount)();
```

型

PFT_GetExtendInfoCount

概要

現在利用可能な拡張情報領域の数を取得します。

拡張情報領域の数はリソースファイルの extendInfoColumnCount によって設定されます。

戻り値

利用可能な拡張情報領域の数が返されます。

互換性

第 12 版以降

3.2.19 PFID_GETEXTENDINFO

拡張情報領域の値を取得します。

```
PFT_BOOL (*PTF_GetExtendInfo)(
          unsigned long LP,
          unsigned long idx,
          PFT_BOOL* pVal
);
```

型

PFT_GetExtendInfo

概要

拡張情報領域に設定されている値を取得します。

拡張情報領域は、各行ごとに複数個の真偽値を保持することが可能な領域です。また、拡張情報領域に真が設定されると、画面の左端にマークが表示され、ユーザは拡張情報領域のステータスを確認することができます。

引数

LP

値を取得する行の行番号を指定してください。

idx

何番目の拡張情報領域の値を取得するのかを指定してください。指定可能な値は 0 から「PFID_GETEXTENDINFOCOUNT で取得される値-1」までです。

pVal

値を取得するための変数のアドレスを指定してください。

戻り値

処理に成功したら真が返されます。

互換性

第 12 版以降

3.2.20 PFID_SGETEXTENDINFO

拡張情報領域に値を設定します。

```
PFT_BOOL (*PTF_SetExtendInfo)(
          unsigned long LP,
          unsigned long idx,
          PFT_BOOL Val
);
```

型

PFT_SetExtendInfo

概要

拡張情報領域に値を設定します。

拡張情報領域は、各行ごとに複数個の真偽値を保持することが可能な領域です。また、拡張情報領域に真が設定されると、画面の左端にマークが表示され、ユーザは拡張情報領域のステータスを確認することができます。

引数

LP

値を設定する行の行番号を指定してください。

idx

何番目の拡張情報領域に値を設定するのか、を指定してください。指定可能な値は 0 から「PFID_GETEXTENDINFOCOUNT で取得される値-1」までです。

pVal

設定する値を指定してください。

戻り値

処理に成功したら真が返されます。

互換性

第 12 版以降

3.2.21 PFID_SAVEFILE

ファイルを保存します。

```
PFT_BOOL (*PFT_SaveFile)();
```

型

PFT_SaveFile

概要

ファイルを保存します。

現在開いているファイルに名前がなかった場合には、「名前を付けて保存」ダイアログボックスが表示されます。

保存時に使用されるエンコードや改行コードは、当該のファイルで使用されているエンコード名・改行コードとなります。「名前を付けて保存」ダイアログによりファイル名が指定された場合には、そのときにユーザにより指定されたエンコード名・改行コードが使用されます。

エンコードや改行コードの種別を指定したい場合には、PFT_SaveFile16 を使用してください。

本 API 関数を呼び出すと、呼び出し前後に行われた編集を一回の Undo 操作で元に戻すことができなくなります。

戻り値

ファイルの保存に成功した場合には真が返されます。

「名前を付けて保存」ダイアログでユーザがキャンセルした、ファイル保存時に実行するコマンドによりファイルの保存がキャンセルされた、もしくは何らかの理由でファイルの保存に失敗した場合には偽が返されます。

互換性

第 12 版以降

3.2.22 PFID_OPENFILE

ファイルを開きます。

```
PFT_BOOL (*PFT_OpenFile)(
            const char* pFileName
);
```

型

PFT_OpenFile

概要

指定された名前のファイルを開きます。

pFileName に NULL もしくは長さ 0 の文字列が指定された場合には、「ファイルを開く」ダイアログボックスを表示します。

ファイルを開くときに使用されるエンコードは、カレントのロケールのエンコードとなります。たとえば、主筆が ja のロケールで起動されている場合には、eucJP が使用されます。

エンコードを指定して開きたい場合には、PFT_OpenFile16 もしくは PFT_OpenFile20 を使用してください。

本 API 関数を呼び出すと、呼び出し前後に行われた編集を一回の Undo 操作で元に戻すことができなくなります。

引数

pFileName

ファイル名を指定してください。

相対パスが指定された場合には、プロセスのカレントディレクトリからの相対パスが使用されます。

長さ 0 の文字列が指定された場合には、「ファイルを開く」ダイアログボックスが表示されます。

戻り値

正常にファイルを開くことができた場合には真が返されます。

「ファイを開く」ダイアログでユーザがキャンセルした、ファイルオープン時に実行するコマンドによりファイルのオープンがキャンセルされた、もしくは何らかの理由でファイルのオープンに失敗した場合には偽が返されます。

互換性

第 12 版以降

3.2.23 PFID_OPENFILENEWWINDOW

ファイルを新しいウインドウで開きます。

```
PFT_BOOL (*PFT_OpenFileNewWindow)(
            const char* pFileName
);
```

型

PFT_OpenFileNewWindow

概要

指定された名前のファイルを新しいウインドウで開きます。

もし pFileName に長さ 0 の文字列が指定された場合には、新しいウインドウが開かれ、新規のファイルが作成されます。

主筆は新しいウインドウを開くために新規にプロセスを立ち上げます。

ファイルを開くときに使用されるエンコードは、カレントのロケールのエンコードとなります。たとえば、主筆が ja のロケールで起動されている場合には、eucJP が使用されます。

エンコードを指定して開きたい場合には、PFT_OpenFile16 もしくは PFT_OpenFile20 を使用してください。

引数

pFileName

ファイル名を指定してください。

相対パスが指定された場合には、プロセスのカレントディレクトリからの相対パスが使用されます。

NULL もしくは長さ 0 の文字列が指定された場合には、新しいウインドウが開かれ、新規のファイルが作成されます。

戻り値

新しいウインドウを開くことに成功した場合には真が返ります。

ファイルのオープンそのものに失敗した場合でも、新規のウインドウを開くことに成功してさえいれば真が返されます。新規のウインドウを開くことができなかった場合には偽が返されます。

互換性

第 12 版以降

3.2.24 PFID_ALLOCWORKMEMORY

作業用メモリを確保します。

```
void* (*PFT_AllocWorkMemory)(
          unsigned long size
          const wchar_t* pName,
          PFT_BOOL IsGlobal
);
```

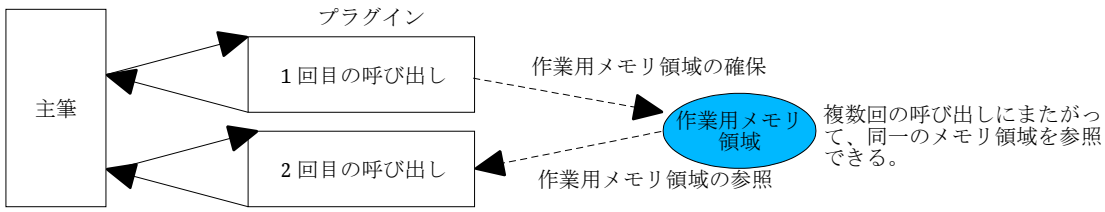
型

PFT_AllocWorkMemory

概要

作業用のメモリ領域を確保します。

この API で確保された作業用メモリ領域は、PFID_FINDWORKMEMORY により名前を指定することで、メモリ領域のアドレス値を参照することができます。また、確保されたメモリ領域は、主筆のプロセスが終了されるまで継続して使用することができます。そのため、プラグインの複数回の呼び出しにわたって継続して値を保持したい場合に使用することができます。



IsGlobal に真を設定した場合は、すべてのプラグインで同一のメモリ領域を参照することができます。そのため、複数種の異なるプラグイン間での通信に利用することができます。IsGlobal に偽を設定した場合は、メモリ領域を参照できるのはメモリ領域の確保を行ったプラグインのみとなります。

名前によるメモリ領域の検索にはある程度の時間がかかります。そのため、一回の呼び出しの中でメモリ領域の確保と解放を行うような場合には、通常の malloc を用いた方が高速です。

この API 関数ではメモリ領域の初期化は行われません。確保された領域内の値は不定なので、必要に応じて初期化を行ってください。

本 API 関数により確保されたメモリ領域は、直接 free で解放せず、必ず PFID_FREEWORKMEMORY を用いて解放するようにしてください。

引数

size

確保するメモリ領域長をバイト単位で指定してください。

pName

作業用メモリ領域の名前を指定してください。

IsGlobal

全プラグイン間で共通して参照可能にする場合は真を指定してください。

戻り値

確保されたメモリ領域のアドレスが返されます。

失敗した場合は NULL が返されます。

互換性

第 12 版以降

3.2.25 PFID_FINDWORKMEMORY

既存の作業用メモリ領域を参照します。

```
void* (*PFT_FindWorkMemory)(
          const wchar_t* pName,
          PFT_BOOL IsGlobal
);
```

型

PFT_FindWorkMemory

概要

PFID_ALLOCWORKMEMORY で確保された作業用メモリ領域を pName で検索し、アドレスを返します。

引数

pName

作業用メモリ領域の名前を指定してください。

戻り値

検索された作業用メモリ領域のアドレスが返されます。

指定された名前の作業用メモリ領域が存在しない場合には NULL が返されます。

互換性

第 12 版以降

3.2.26 PFID_FREEWORKMEMORY

既存の作業用メモリ領域を解放します。

```
void* (*PFT_FreeWorkMemory)(
          const wchar_t* pName,
          PFT_BOOL IsGlobal
);
```

型

PFT_FreeWorkMemory

概要

PFID_ALLOCWORKMEMORY で確保された作業用メモリ領域を pName で検索し、その作業用メモリ領域を開放します。

不要になった作業用メモリ領域はこの API 関数で解放してください。

指定された名前の作業用メモリ領域が存在しない場合には、この関数は何も処理を行いません。

引数

pName

作業用メモリ領域の名前を指定してください。

互換性

第 12 版以降

3.2.27 PFID_ISCONSTRUCTIONMODE

コンストラクション・モードか否かを取得します。

```
PFT_BOOL (*PFT_IsConstructionMode)();
```

型

PFT_IsConstructionMode

概要

プラグインを呼び出した主筆が、コンストラクション・モードか否かを取得します。

主筆サーバが起動していて、かつ、プラグインを呼び出した主筆がサーバの管理下におかれている場合には真を返します。

互換性

第 15 版以降

3.2.28 PFID_SETCONSTRUCTION

コンストラクション・モードに設定します。

```
PFT_BOOL (*PFT_SetConstruction)(
          PFT_BOOL ShowMessage
);
```

型

PFT_SetConstruction

概要

プラグインを呼び出した主筆を、コンストラクション・モードに設定します。

コンストラクション・モードへの移行に成功した場合には真を返します。

なお、以下の場合にはコンストラクション・モードへの移行は失敗します。

- 主筆サーバが起動していない場合
- プラグインを呼び出した主筆が開いているファイルが、他のコンストラクション・モードの主筆によって開かれている場合。

引数

ShowMessage

この引数に真を指定し、かつ、コンストラクション・モードへの移行に失敗した場合は、主筆側でエラーメッセージを表示します。

この引数に偽を指定した場合は、失敗した場合にもメッセージは表示しません。

互換性

第 15 版以降

3.2.29 PFID_SETSTANDALONE

コンストラクション・モードに設定します。

```
PFT_BOOL (*PFT_SetStandalone)(
            PFT_BOOL ShowMessage
);
```

型

PFT_SetStandalone

概要

プラグインを呼び出した主筆を、スタンドアロン・モードに設定します。

スタンドアロン・モードへの移行に成功した場合には真を返します。

引数

ShowMessage

この引数に真を指定し、かつ、スタンドアロン・モードへの移行に失敗した場合は、主筆側でエラーメッセージを表示します。

この引数に偽を指定した場合は、失敗した場合にもメッセージは表示しません。

互換性

第15版以降

3.2.30 PFID_GETENCODENAME

エンコード名を取得します。

```
unsigned long (*PFT_GetEncodeName)(
          char *pBuf,
          unsigned long BufLength
);
```

型

PFT_GetEncodeName

概要

現在主筆で開かれているファイルで使用されているエンコード名を取得します。

エンコード名を返すためのバッファが不足する場合には、エンコード名は設定されません。

引数

pBuf

エンコード名を取得するためのバッファのアドレスを指定します。

BufLength

pBuf のバッファ長を指定します。

戻り値

エンコード名の文字数を返します。

互換性

第 16 版以降

3.2.31 PFID_GETCRTYPE

改行コードの種別を取得します。

```
unsigned long (*PFT_GetCRType)(
          char *pBuf
);
```

型

PFT_GetCRType

概要

現在主筆で開かれているファイルで使用されている改行コードの種別を取得します。

種別に応じて、下記の値が返されます。

種別	値
CR	CR
LF	LF
CR+LF	CRLF
LF+CR	LFCR

引数には必ず 5 バイト以上の空き領域があるバッファを指定してください。バッファ長が不足する場合の動作は未定義です。

引数

pBuf

改行コードの種別を取得するためのバッファのアドレスを指定します。必ず 5 バイト以上あるバッファを指定してください。

戻り値

成功した場合には真を返します。

互換性

第 16 版以降

3.2.32 PFID_SAVEFILE16

ファイルの保存を行います。

```
PFT_BOOL (*PFT_SaveFile16)(
          const char *pFileName,
          const char *pEncodeName,
          const char *pCRTypeName,
          PFT_BOOL QueryFlg
);
```

型

PFT_SaveFile16

概要

ファイルを保存します。pFileName に NULL もしくは長さ 0 の文字列を指定した場合には、上書き保存が行われます。

上書き保存を行う場合で、かつ開いているファイルにファイル名がなかった場合、QueryFlg に真が設定されていれば「名前を付けて保存」ダイアログが表示され、ユーザに保存先ファイル名を問い合わせます。QueryFlg に偽が設定されていた場合には、保存処理は行われません。

pEncodeName と pCRTypeName に、NULL もしくは長さ 0 の文字列を指定した場合には、現在開いているファイルのエンコード名と改行コードの種別が使用されます。また、上書き保存が行われる場合には、エンコードと改行コードの指定は無視されます。

本 API 関数を呼び出すと、呼び出し前後に行われた編集を一回の Undo 操作で元に戻すことができなくなります。

引数

pFileName

ファイル名を指定してください。NULL または長さ 0 の文字列が指定された場合には上書き保存が行われます。

pEncodeName

エンコード名を指定してください。

NULL または長さ 0 の文字列を指定した場合は、現在開いているファイルのエンコードが使用されます。名前のないファイルの場合は、カレントのロケールで使用されるエンコードが使用されます。

上書き保存が行われる場合には、エンコード名の指定は無視されます。

pCRTypeName

改行コードの種別を指定してください。指定可能な種別は"CR"・"LF"・"CRLF"・"LFCR"です。

NULL もしくは長さ 0 の文字列を指定した場合は、現在開いているファイルで使用されている改行コードの種別が使用されます。名前のないファイルの場合は、"LF"が使用されます。

上書き保存が行われる場合は、改行コードの種別の指定は無視されます。

QueryFlg

名前のないファイルに対して上書き保存を行う際、「名前を付けて保存」ダイアログを表示するか否かを指定します。真を指定した場合には、ダイアログを表示してユーザに保存先ファイル名を問い合わせます。偽を指定した場合は、ファイルの保存処理を行わずに偽を返して終了します。

pFileName にファイル名が指定された場合や、名前のあるファイルを上書き保存する場合には、QuertFlg の値は無視されます。

戻り値

ファイルの保存に成功した場合は真を返します。保存されなかった場合には偽が返

されます。

互換性

第 16 版以降

3.2.33 PFID_OPENFILE16

ファイルを開きます。

```
PFT_BOOL (*PFT_OpenFile16)(
          const char *pFileName,
          const char *pEncodeName,
          const char *pLangType、
          PFT_BOOL NewWindow
);
```

型

PFT_OpenFile16

概要

ファイル名・エンコード名・構文種別を指定して、ファイルを開きます。

NewWindow に真を設定した場合には、新しいウインドウでファイルを開きます。

なお、この関数を呼び出すと、呼び出し前後に行われた編集を一回の Undo 操作で元に戻すことができなくなります。

引数

pFileName

ファイル名を指定してください。

NULL もしくは長さ 0 の文字列を指定した場合、かつ、NewWindow に偽が設定されていた場合は、「ファイルを開く」ダイアログを表示してユーザにファイル名を問い合わせます。NewWindow に真が設定された場合には、新しいウインドウを開きますが、ファイルは開かれません。

pEncodeName

エンコード名を指定してください。

NULL または長さ 0 の文字列を指定した場合は、カレントのロケールで使用されるエンコードが使用されます。

pfileName に NULL もしくは長さ 0 の文字列が指定された場合には、pencodeName に指定された値は無視されます。

pLangType

構文種別を指定してください。

NULL または長さ 0 の文字列が指定された場合には、ファイル名から自動的に決定します。

pFileName に NULL もしくは長さ 0 の文字列が指定された場合には、pLangType に指定された値は無視されます。

NewWindow

ファイルを新しいウインドウで開く場合には真を指定してください。偽を指定すると、現在のウインドウで開きます。

戻り値

ファイルのオープンに成功した場合は真を返します。失敗した場合は偽が返されます。

互換性

第 16 版以降

3.2.34 PFID_OPENFILE20

ファイルを開きます。

```
PFT_BOOL (*PFT_OpenFile20)(
          const char *pFileName,
          int EncodeType,
          const char *pEncodeName,
          const char *pLangType、
          PFT_BOOL NewWindow
);
```

型

PFT_OpenFile20

概要

ファイル名・エンコード種別・エンコード名・構文種別を指定して、ファイルを開きます。

NewWindow に真を設定した場合には、新しいウインドウでファイルを開きます。

なお、この関数を呼び出すと、呼び出し前後に行われた編集を一回の Undo 操作で元に戻すことができなくなります。

引数

pFileName

ファイル名を指定してください。

NULL もしくは長さ 0 の文字列を指定した場合、かつ、NewWindow に偽が設定されていた場合は、「ファイルを開く」ダイアログを表示してユーザにファイル名を問い合わせます。NewWindow に真が設定された場合には、新しいウインドウを開きますが、ファイルは開かれません。

EncodeType

エンコードの種別を指定してください。

下記の値が使用できます。

値	説明
0	エンコード名を指定しません。ファイルのエンコードは、リソースの enptyEncodeNameBehavior および encodeRecognizeFailedBehavior の指定により決定されます。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。
1	エンコードの自動認識を行います。自動認識に失敗した場合は、リソースの encodeRecognizeFailedBehavior の指定に従いエンコード名を決定します。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。
2	ファイルのエンコードが、カレントのロケールで使用されるエンコードであると見なします。たとえば、現在のロケールが ja であれば、ファイルの文字コードは eucJP であると判断します。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。
3	画面にエンコード名の選択を促すダイアログボックスを表示し、ユーザにエンコード名を問い合わせます。なお、この値を指定した場合は、pEncodeName に指定された値は無視されます。
4	pEncodeName に指定したエンコード名を使用します。なお、EncodeType に 4 を指定し、かつ、pEncodeName に NULL や長さ 0 の文字列を指定した場合は、指定なしと見なします

なお、pFileName に NULL もしくは長さ 0 の文字列が指定された場合には、EncodeType に指定された値は無視されます。

pEncodeName

エンコード名を指定してください。

NULL または長さ 0 の文字列を指定した場合は、エンコード名が指定されなかったものと見なし、リソースの enptyEncodeNameBehavior および encodeRecognizeFailedBehavior の指定に従いエンコード名を決定します。

なお、EncodeType に 4 以外の値が指定された場合、もしくは、pFileName に NULL もしくは長さ 0 の文字列が指定された場合には、pEncodeName に指定された値は無視されます。

pLangType

構文種別を指定してください。

NULL または長さ 0 の文字列が指定された場合には、ファイル名から自動的に決定します。

pFileName に NULL もしくは長さ 0 の文字列が指定された場合には、pLangType の指定は無視されます。

NewWindow

ファイルを新しいウインドウで開く場合には真を指定してください。偽を指定すると、現在のウインドウで開きます。

戻り値

ファイルのオープンに成功した場合は真を返します。失敗した場合は偽が返されます。

互換性

第 20 版以降

3.2.35 PFID_SYMALLOC

メモリ領域を確保します。

```
void* (*PFT_SyMalloc)(
          unsigned long size
);
```

型

PFT_SyMalloc

概要

ヒープからメモリ領域を確保します。C 言語の標準ライブラリで提供される malloc 関数と同等です。

なお、将来の互換性、および、主筆本体とプラグインとでコンパイル時の指定が異なる場合に生じうる問題を防ぐために、本関数により確保されたメモリ領域は、必ず SyFree 関数を用いて解放してください。

引数

size

エンコード名を取得するためのバッファのアドレスを指定します。

戻り値

確保されたメモリ領域のアドレスが返されます。失敗した場合は NULL が返されます。

詳細は malloc 関数の仕様を参照してください。

互換性

第 20 版以降

3.2.36 PFID_SYCALLOC

ゼロクリアしたメモリ領域を確保します。

```
void* (*PFT_SyCalloc)(
          unsigned long n,
          unsigned long s
);
```

型

PFT_SyCalloc

概要

ヒープからメモリ領域を確保し、ゼロクリアします。C 言語の標準ライブラリで提供される calloc 関数と同等です。

なお、将来の互換性、および、主筆本体とプラグインとでコンパイル時の指定が異なる場合に生じうる問題を防ぐために、本関数により確保されたメモリ領域は、必ず SyFree 関数を用いて解放してください。

引数

n

メモリ上に確保する要素の要素数を指定します。

s

メモリ上に確保する要素の個数を指定します。

戻り値

確保されたメモリ領域のアドレスが返されます。失敗した場合は NULL が返されます。

詳細は calloc 関数の仕様を参照してください。

互換性

第 20 版以降

3.2.37 PFID_SYREALLOC

SyMalloc や SyCalloc で確保されたメモリ領域の、領域長を変更します。

```
void* (*PFT_SyRealloc)(
          void *pBuf,
          unsigned long size
);
```

型

PFT_SyRealloc

概要

SyMalloc や SyCalloc により割り当てられたメモリ領域の、領域長を変更します。C 言語の標準ライブラリで提供される realloc 関数と同等です。

なお、将来の互換性、および、主筆本体とプラグインとでコンパイル時の指定が異なる場合に生じうる問題を防ぐために、本関数の pBuf に指定するメモリ領域は、SyMalloc と SyCalloc によるメモリ領域のみとしてください。同様に、本関数により割り当てられたメモリ領域を開放する場合には、SyFree 関数を用いるようにしてください。

引数

pbuf

SyMalloc や SyCalloc に割り当てられたメモリ領域のアドレスを指定してください。

s

サイズ変更後のバッファ長を指定してください。

戻り値

サイズ変更後のメモリ領域のアドレスが返されます。失敗した場合は NULL が返されます。

詳細は realloc 関数の仕様を参照してください。

互換性

第 20 版以降

3.2.38 PFID_SYFREE

SyMalloc や SyCalloc・SyRealloc で確保されたメモリ領域のを解放します。

```
void* (*PFT_SyFree)(
          void *pBuf
);
```

型

PFT_SyFree

概要

SyMalloc や SyCalloc により割り当てられたメモリ領域を解放し、その他の用途で利用可能な状態とします。C 言語の標準ライブラリで提供される free 関数と同等です。

なお、将来の互換性、および、主筆本体とプラグインとでコンパイル時の指定が異なる場合に生じうる問題を防ぐために、本関数の pBuf に指定するアドレスは、SyMalloc と SyCalloc によって割り当てられたメモリ領域のアドレス値のみとしてください。また、同様の理由から、主筆本体側で確保されたメモリ領域を開放する場合には、必ず本関数を用いるようにしてください。

引数

pbuf

SyMalloc や SyCalloc・SyRealloc に割り当てられたメモリ領域のアドレスを指定してください。

戻り値

戻り値はありません。

詳細は free 関数の仕様を参照してください。

互換性

第 20 版以降

3.2.39 PFID_SHOWENCODEDIALOG

エンコード選択ダイアログを表示し、ユーザにエンコード名の選択を求めます。

```
PFT_BOOL (*PFT_ShowEncodeDialog)(
            PFT_ENCODEDLGINFO *pInfo
);
```

型

PFT_ShowEncodeDialog

概要

エンコード選択ダイアログを表示し、ユーザにエンコード名の選択を求めます。なお、本関数が提供する機能は、ユーザが入力したエンコード名を取得するだけであり、現在開いているファイル等のエンコードの指定は変更されません。

引数

pInfo

PFT_ENCODEDLGINFO 構造体のアドレスを指定してください。
PFT_ENCODEDLGINFO 構造体に指定する値については、PFT_ENCODEDLGINFO 構造体の説明を参照してください。

戻り値

関数の処理が成功した場合は PFT_TRUE が返されます。失敗した場合には PFT_FALSE が返されます。

なお、ユーザがキャンセルボタンを押下してダイアログを閉じた場合、その他にエラーが発生していなければ本関数の戻り値は PFT_TRUE となります。ユーザが OK ボタンを押下したのか否かを取得する場合は、pInfo->Result の値を参照してください。

互換性

第 20 版以降

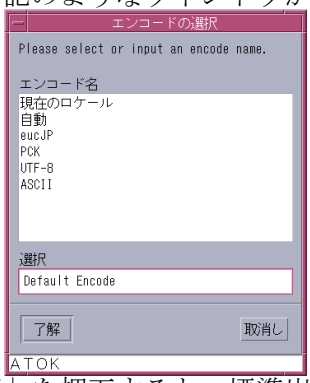
使用例

プログラム例

```
void foo( PFT_GetAPIFunction GetAPIFunction )
{
  // API 関数の関数ポインタを取得する。
  PFT_SyFree SyFree = (PFT_SyFree)GetAPIFunction( PFID_SYFREE );
  PFT_ShowEncodeDialog ShowEncodeDialog =
    (PFT_ShowEncodeDialog)GetAPIFunction( PFID_SHOWENCODEDIALOG );

  // エンコード選択ダイアログの情報を保持する構造体を初期化する。
  PFT_ENCODEDLGINFO d;
  memset( &d, 0, sizeof( PFT_ENCODEDLGINFO ) );

  d.Size = sizeof( PFT_ENCODEDLGINFO );
  d.pDlgMessage = "Please select or input an encode name.";
  d.InitEncode.Size = sizeof( PFT_ENCODEINFO );
  d.InitEncode.Type = PFT_ENCODE_TYPE_SPECIFY;
  d.InitEncode.pEncodeName = "Default Encode";
  d.Flg = PFT_ENCDLG_SHOWCURRENT | PFT_ENCDLG_SHOWAUTO;
  d.Result = PFT_FALSE;
  d.Encode.Size = sizeof( PFT_ENCODEINFO );
  d.Encode.Type = PFT_ENCODE_TYPE_NON;
  d.Encode.pEncodeName = NULL;

  // エンコード選択ダイアログを表示する。
  if ( ShowEncodeDialog( &d ) ) {
    // OK ボタンが押下されたのか否か。
    printf( "Result = %s\n", d.Result ? "TRUE" : "FALSE" );
    if ( d.Result ) {
      printf( "Encode.Type = %d\n", d.Encode.Type );
      printf( "Encode.pEncodeName = %s\n", d.Encode.pEncodeName ? d.Encode.pEncodeName : "" );
      // エンコード名を保持する文字列を解放する。
      //  (下記の処理を行わないとメモリリークが発生する)
      SyFree( d.Encode.pEncodeName );
    }
  }
}
```

実行結果

関数が呼び出されると、下記のようなウインドウが表示されます。



上記のウインドウで「了解」を押下すると、標準出力に下記のようなメッセージが出力されます。

```
Result = TRUE
Encode.Type = 1
Encode.pEncodeName = Default Encode
```

3.2.40 PFID_SHOWFILEDIALOG

ファイルを開く / 名前を付けて保存ダイアログを表示し、ユーザにファイルの選択を求めます。

```
PFT_BOOL (*PFT_ShowFileDialog)(
            PFT_FILEDLGINFO *pInfo
);
```

型

PFT_ShowFileDialog

概要

ファイルを開くもしくは名前を付けて保存ダイアログを表示し、ユーザにファイルの選択を求めます。なお、本関数が提供する機能は、ユーザが入力したファイル名を取得するだけであり、現在開いているファイルのファイル名などは変更されません。

引数

pInfo

PFT_FILEDLGINFO 構造体のアドレスを指定してください。PFT_FILEDLGINFO 構造体に指定する値については、PFT_FILEDLGINFO 構造体の説明を参照してください。

戻り値

関数の処理が成功した場合は PFT_TRUE が返されます。失敗した場合には PFT_FALSE が返されます。

なお、ユーザがキャンセルボタンを押下してダイアログを閉じた場合、その他にエラーが発生していなければ本関数の戻り値は PFT_TRUE となります。ユーザが OK ボタンを押下したのか否かを取得する場合は、pInfo->Result の値を参照してください。

互換性

第 20 版以降

使用例

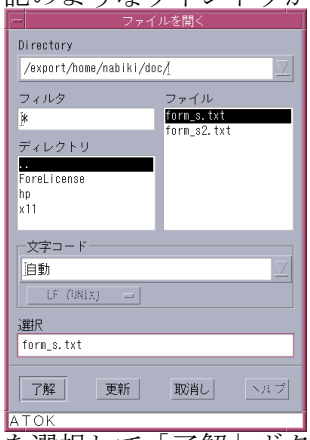
プログラム例

```
void foo( PFT_GetAPIFunction GetAPIFunction )
{
  // API 関数の関数ポインタを取得する。
  PFT_SyFree SyFree = (PFT_SyFree)GetAPIFunction( PFID_SYFREE );
  PFT_ShowFileDialog ShowFileDialog =
    (PFT_ShowFileDialog)GetAPIFunction( PFID_SHOWFILEDIALOG );

  // ファイルを開く / 名前を付けて保存ダイアログの情報を保持する構造体を初期化する。
  PFT_FILEDLGINFO d;
  memset( &d, 0, sizeof( PFT_FILEDLGINFO ) );

  d.Size = sizeof( PFT_FILEDLGINFO );
  d.Flg =  PFT_FILEDLG_OPENDLG | PFT_FILEDLG_ENABLE_CRSELECT | PFT_FILEDLG_ENABLE_ENCODESELECT;
  d.pInitFileName = "/export/home/a.txt";
  d.pFilter = "*.txt";
  d.InitEncode.Size = sizeof( PFT_ENCODEINFO );
  d.InitEncode.Type = PFT_ENCODE_TYPE_SPECIFY;
  d.InitEncode.pEncodeName = "euc-jp";
  d.InitCRType = PFT_CRTYPE_LFCR;
  d.Result = PFT_FALSE;
  d.Encode.Size = sizeof( PFT_ENCODEINFO );
  d.Encode.Type = PFT_ENCODE_TYPE_NON;
  d.Encode.pEncodeName = NULL;
  d.pFileName = NULL;
  d.CRType = PFT_CRTYPE_LFCR;

  // ファイルを開くダイアログを表示する。
  if ( ShowFileDialog( &d ) ) {
    // OK ボタンが押下されたのか否か。
    printf( "Result = %s\n", d.Result ? "TRUE" : "FALSE" );
    if ( d.Result ) {
      printf( "Encode.Type = %d\n", d.Encode.Type );
```

```
        printf( "Encode.pEncodeName = %s\n", d.Encode.pEncodeName ? d.Encode.pEncodeName : "" );
        printf( "pFileName = %s\n", pFileName ? pFileName : "" );
        printf( "CRType = %d\n", CRType  );
        // エンコード名・ファイル名を保持する文字列を解放する。
        //  (下記の処理を行わないとメモリリークが発生する)
        SyFree( d.Encode.pEncodeName );
        SyFree( d.pFileName );
    }
  }
}
```

実行結果

関数が呼び出されると、下記のようなウインドウが表示されます。



上記画面で、適宜ファイルを選択して「了解」ボタンを押下すると、標準出力に下記のようなメッセージが出力されます。

```
Result = TRUE
Encode.Type = 1
Encode.pEncodeName = euc-jp
pFileName = /export/home/nabiki/doc/form_s.txt
CRType = 3
```

4 例

下記に、簡単なプラグインの例を示します。

4.1 プログラム

プ呼び出されたときに「Welcome to Syuhitu plugin hell world」というメッセージ表示するプラグインを作成します。

下記にコーディング例を示します。

msg.c

```
#include "../TaEdit/PluginFuncID.h"
#include <wchar.h>

/* メニューが選択されたときに、この関数が実行されます。 */
void ShowMessage( PFT_GetAPIFunction GetAPIFunction )
{
  /* メッセージボックスを表示するための、API 関数を取得します */
  PFT_ShowInformationMsgBox ShowInformationMsgBox =
    PFT_ShowInformationMsgBox)GetAPIFunction( PFID_SHOWINFORMATIONMSGBOX );

  /* メッセージを表示します */
  ShowInformationMsgBox( L"Welcome to Syuhitu plugin hell world" );
}
```

4.2 コンパイル

上記「msg.c」ファイルを、下記のコマンドでコンパイルします。

```
cc -xarch=v9 -G -o msg.so msg.c
```

なお、主筆は標準では 64 ビットでコンパイルされているため、対象アーキテクチャで v9 を指定して 64 ビットでコンパイルされるようにして下さい。そうでないと、実行時にライブラリのロードに失敗します。

4.3 組み込み

コンパイルして生成された「msg.so」ファイルを、環境変数 LD_LIBRARY_PATH が設定されているフォルダに移動して下さい。なお、主筆の標準のインストールでは、実行時にインストール先ディレクトリ内の「plugin」ディレクトリを、LD_LIBRARY_PATH に追加します。

下記のようにプラグイン設定ファイルを記述して下さい。

```
[00000]
PluginName = MyPlugin
LibraryName = msg.so
FunctionName = ShowMessage
```

プラグイン設定ファイルは、デフォルトでは/opt/NBKTtaed/plugin/Plugin.ini に格納されています。また、主筆が参照するプラグイン設定ファイルのパス名は、リソースファイルの TaEdit.pluginConfigFileName により指定します。

セクション名と PluginName は任意です。

LibraryName は、コンパイルした結果生成された「msg.so」が参照できるように記述して下さい。フルパスで記述しても問題ありません。

FunctionName は、プラグインを実行するときに呼び出す関数の名前を指定して下さい。ここでは上記ソースコードより「ShowMessage」となります。

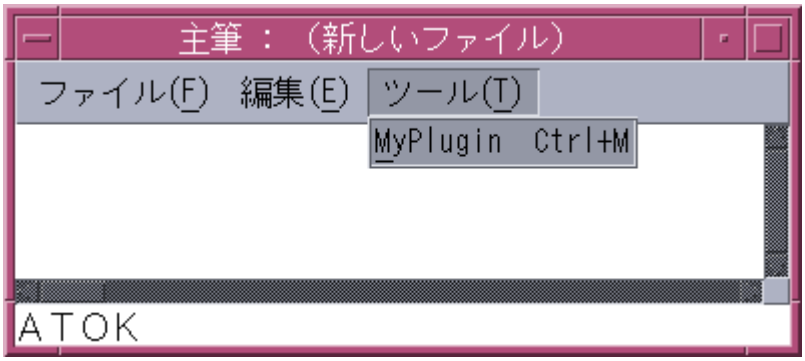
下記の文をリソースファイルに追加して下さい。

```
TaEdit*TEPI_MyPlugin.labelString : MyPlugin
TaEdit*TEPI_MyPlugin.mnemonic: M
TaEdit*TEPI_MyPlugin.accelerator: Ctrl<Key>M
TaEdit*TEPI_MyPlugin.acceleratorText: Ctrl+M
```

主筆で使用するリソースを記述したリソースファイルは、デフォルトでは/opt/NBKTtaed/resource/ロケール名/syuhitu.res に格納されています。このファイルを直接編集するか、あるいは、その他のリソースファイル (たとえば ~/.Xdefaults) を編集してください。

4.4 実行

以上を設定を終えたら主筆を起動して下さい。ツールメニューに下記のようなメニュー項目が表示されます。



「MyPlugin」を選択すると、情報メッセージボックスが表示されます。