1　ListCtrlEx とは

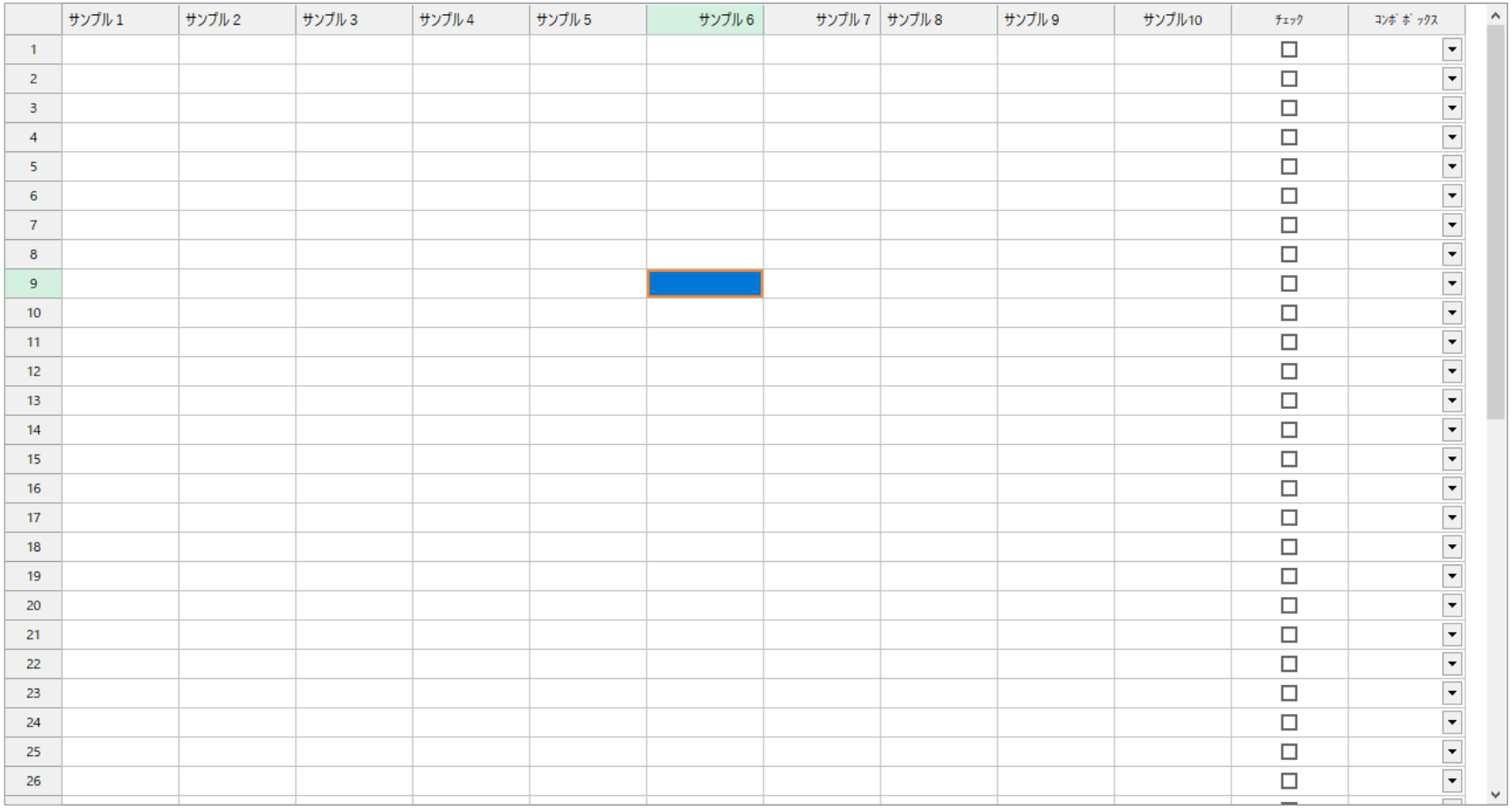
MFC【Microsoft Foundation Class】の CListCtrl を継承したグリッド感覚で編集できるようにするたのクラスライブラリです。

クラス名は「CListCtrlEx」、名前空間は「Koei」です。

※ブロック選択、列選択、セル結合、行や列の固定は対応していません。

※CListCtrlEx クラスを継承して使用することはできません。

※「2.1 開発環境」を参照してください。



目次

2　はじめに

2.1　開発環境

2.1.1 共有 DLL で MFC を使う場合

Visual Studio	Visual Studio 2015～2022（MFC バージョンが 14）
exe プロジェクト	MFC アプリケーション 文字セット：Unicode のみ プラットフォーム：x64 のみ インポートライブラリファイル：ListCtrlEx.lib（デバッグの場合は ListCtrlExd.lib） DLL ファイル：ListCtrlEx.dll（デバッグの場合は ListCtrlExd.dll） ヘッダーファイル：ListCtrlEx.h

2.1.2 スタティック ライブラリで MFC を使用する場合

Visual Studio	Visual Studio 2015～2022（MFC バージョンが 14）
exe プロジェクト	MFC アプリケーション 文字セット：Unicode のみ プラットフォーム：x64 のみ スタティックリンクライブラリファイル：ListCtrlExLib.lib（デバッグの場合は ListCtrlExLibd.lib） ヘッダーファイル：ListCtrlEx.h

2.2　ライセンスキー

ライセンスキーはメジャーバージョンごとに発行します。
購入された個人または団体が開発するすべてのアプリケーションで使用可能です。
メジャーバージョンの異なる製品を利用するには、ライセンスキーの購入が必要になります。
ライセンスキーは「4.3 インスタンス作成」にて指定します。

2.3　試用

購入前（ライセンスキー取得前）に試用することができます。
試用版として動かすには、「4.3 インスタンス作成」の第一引数（ライセンスキー）に NULL または_T(“”)を渡します。
試用版では、扱える行数が 20 行までとなります。

2.4　再配布

本製品を使用して開発されたアプリケーションに同梱して配布する場合のみ、再配布を許可します。
再配布できるファイルは、「ListCtrlEx.dll」のみです。

2.5　送金方法

ベクターレジにてお支払いください。
お支払い確認後、ライセンスキーが届きます。
1 ライセンス 16,500 円（税込み）です。

2.6　連絡先

メールでご連絡ください。
岡栄システム有限会社
software@koei-system.co.jp

3　更新履歴

3.1　バージョン　1.00

1. 新規

3.2　バージョン　1.10

1. スタティックリンクライブラリ版(ListCtrlExLib.lib)を同梱。
   アプリケーション側はスタティックリンクライブラリで MFC を使用することが可能になります。
2. 行ヘッダーの幅変更時のマウスカーソルを変更。
3. セル、列の色設定で文字色または背景色のみ変更できるようにしました。
4. Esc キー押下で Esc(0x1B)文字が入力される不具合の対応しました。

4　作成

4.1　リソースを使用する場合

リソース	ダイアログに「List Control」を配置します。
	配置したリストコントロールのプロパティを設定します。 ・オーナー描画の固定：True ・ビュー：レポート ・ラベルの編集：False
xxxDlg.h	「#include "ListCtrlEx.h"」を追加します。 ダイアログにリストコントロールのコントロールメンバ変数を追加します。

	`#include "ListCtrlEx.h"`
	`class CxxxDlg : public CDialogEx`
	`{`
	`　　Koei::CListCtrlEx* m_pList;`

xxxDlg.cpp	コンストラクタでコントロールメンバ変数を初期化します。

	`using namespace Koei;`
	`CxxxDlg::CxxxDlg(CWnd* pParent /*=nullptr*/)`
	`　　: CDialogEx(IDD_USER_DIALOG, pParent)`
	`{`
	`　　m_pList = NULL;`
	`}`

	コントロールメンバ変数をサブクラス化します。

	`BOOL CxxxDlg::OnInitDialog()`
	`{`
	`　　CDialogEx::OnInitDialog();`
	
	`　　m_pList = CListCtrlEx::Instance(_T("ライセンスキー"));`
	`　　m_pList->SubclassDlgItem(IDC_LIST, this);`

	デストラクタ等でメモリを開放します

	`CxxxDlg::~CxxxDlg()`
	`{`
	`　　if (m_pList) {`
	`　　　　delete m_pList;`
	`　　　　m_pList = NULL;`
	`　　}`

	}

4.2　リソースを使用しない場合

xxxView.h	「#include "ListCtrlEx.h"」を追加します。 View 等にリストコントロールのコントロールメンバ変数を追加します。
	#include "ListCtrlEx.h" class CxxxView : public Cview { 　　Koei::CListCtrlEx* m_pList;
xxxDlg.cpp	コンストラクタでコントロールメンバ変数を初期化します。

	using namespace Koei;
	CxxxView::CxxxView
	{
	　　m_pList = NULL;
	}

	コントロールを作成します。

	int CxxxView:: OnCreate(LPCREATESTRUCT lpCreateStruct)
	{
	　　if (CView::OnCreate(lpCreateStruct) == -1)
	　　　　return;
	
	　　m_pList = CListCtrlEx::Instance(_T("ライセンスキー"));
	　　CRect rect(10, 10, 100, 100);
	　　m_pList->Create(WS_VISIBLE | WS_TABSTOP, rect, this, 5000);

	デストラクタでメモリを開放します。

	CxxxDlg::~CxxxDlg()
	{
	　　if (m_pList) {
	　　　　delete m_pList;
	　　　　m_pList = NULL;
	　　}
	}

4.3　インスタンス作成

Instance

const CString&	csLincenseKey	ライセンスキー NULL または空文字を指定すると試用版として動作します。 試用版では 20 行までしか扱えません。
CListCtrlEx*	戻り値	インスタンス ライセンスキーが正しくない場合は、NULL
static 関数です。		
m_pList = CListCtrlEx::Instance(_T(“ライセンスキー”));		

4.4　コントロール作成

Create		
DWORD	dwStyle	スタイル ・通常は WS_VISIBLE、WS_TABSTOP を指定します。 ・LVS_EDITLABELS、LVS_ICON、LVS_LIST、LVS_SMALLICON、LVS_OWNERDATA は指定しても無視されます。 ・LVS_REPORT、LVS_OWNERDRAWFIXED、WS_BORDER、WS_CLIPCHILDREN、WS_CHILD は指定していなくても付加されます。
const RECT&	rect	座標
CWnd*	pParentWnd	親ウィンドウ
UINT	nID	コントロールの ID
BOOL	戻り値	TRUE：成功 FALSE：失敗

5　列

5.1　追加

InsertColumn		
int	nCol	新しい列のインデックス
LPCTSTR	lpszColumnHeading	列名 「¥n」で改行
BOOL	bReadOnly	TRUE：読取り専用列 FALSE：編集可能列
int	nFormat	表示位置 LVCFMT_LEFT：左寄せ LVCFMT_RIGHT：右寄せ LVCFMT_CENTER：中央寄せ
int	nWidth	列幅
int	nSubItem	列に関連付けられているサブ項目のインデックス
int	戻り値	追加された列のインデックス
第 3 引数以外、詳細は MSDN の CListCtrl::InsertColumn を参照してください。		

5.2　タイプ設定（エディットコントロール）

SetColumnEditType		
int	nCol	列のインデックス
void	戻り値	
エディットコントロールで編集する列にします。		

5.3　タイプ設定（チェックボックス）

SetColumnCheckType		
int	nCol	列のインデックス
BOOL	bDefaultCheck	新規行のチェック有無
void	戻り値	
チェックボックスで編集する列にします。 ・空以外のセルの値がすべて”1”に設定されチェック ON の状態になります。 ・チェックボックス ON にするには、SetItemText(行, 列, _T(“1”)) ・チェックボックス OFF にするには、SetItemText(行, 列, _T(“”))		

5.4　タイプ設定（コンボボックス）

SetColumnComboType		
int	nCol	列のインデックス
CStringList*	comboList	コンボボックスの選択リスト “値¥t 表示文字列”の形式の文字列リスト
const CString&	csDefaultValue	新規行の選択値
int	nMaxDropDownCount	ドロップダウン部分に表示される項目の最大数
void	戻り値	
コンボボックスで編集する列にします。		

5.5　削除

DeleteColumn		
int	nCol	列のインデックス

BOOL	戻り値	TRUE：削除成功 FALSE：削除失敗

5.6　色設定

SetColColor		
int	iCol	列のインデックス -1：すべての列
COLORREF	colText	文字色 設定を変更しない場合は、CLR_NONE を設定します。 設定をクリアする場合は、CLR_DEFAULT を設定します。
COLORREF	colBack	背景色 設定を変更しない場合は、CLR_NONE を設定します。 設定をクリアする場合は、CLR_DEFAULT を設定します。
void	戻り値	
表示は、「9.13 色設定」が優先されます。		

5.7　色取得

GetColColor		
int	iCol	列のインデックス
COLORREF*	pcolText	文字色
COLORREF*	pcolBack	背景色
BOOL	戻り値	TRUE：列の色設定あり FALSE：列の色設定なし

5.8　色クリア

ClearColColor		
int	iCol	列のインデックス -1：すべての列
void	戻り値	

5.9　読み取り専用設定

SetColReadOnly		
int	iCol	列のインデックス -1：すべての列
BOOL	bReadOnly	TRUE：読み取り専用にする FALSE：編集可能にする
void	戻り値	
対象の列に設定されていた、セルの読取り専用設定は解除されます。		

5.10 読み取り専用設定取得

IsColReadOnly		
int	iCol	列のインデックス
BOOL	戻り値	TRUE：読み取り専用設定している FALSE：読み取り専用設定していない
列に読み取り専用設定しているかチェックします。		

6 列ヘッダー

6.1 高さ設定

SetHeaderLineCount		
int	iLineCount	行数 -1：非表示
void	戻り値	
ヘッダーの高さを行数で指定します。		

SetHeaderHeight		
int	iHeight	高さ（ピクセル） -1：非表示
void	戻り値	
ヘッダーの高さをピクセル単位で指定します。		

6.2 高さ取得

GetHeaderHeight		
int*	piHeight	高さ（ピクセル） NULL 指定可能
int	戻り値	行数 SetHeaderHeight を使用して高さを設定している場合は-1
ヘッダーの高さと行数を取得します。		

6.3　テキスト表示位置設定

SetHeaderAlign		
HDEX_ALIGN	Align	テキスト表示位置 HDEX_ALIGN:: LISTCTRLSAME：データ部分の表示位置に従う HDEX_ALIGN:: LEFTALWAYS：左寄せ HDEX_ALIGN:: CENTERALWAYS：中央寄せ HDEX_ALIGN:: RIGHTALWAYS：右寄せ
void	戻り値	

6.4　幅変更可否設定

SetColumnEnableSizing		
BOOL	bEnable	TRUE：幅変更可 FALSE：幅変更不可
void	戻り値	
マウスによる列幅変更の可否を設定します。 行ヘッダーの幅の変更可否も設定されます。		

6.5　ドラッグドロップによる列並べ替え可否設定

SetHeaderDragDrop		
BOOL	bDragDrop	TRUE：並べ替え可 FALSE：並べ替え不可
void	戻り値	
マウスによる列並び替えの可否を設定します。		

6.6　フォント設定

SetHeaderFont		
LPCTSTR	lpszFaceName	フォント名
int	nPointSize	フォントサイズ
BOOL	bBold	太字 TRUE：太字 FALSE：通常
BOOL	bItalic	斜体 TRUE：斜体 FALSE：通常
void	戻り値	
行ヘッダーのフォントも変更されます。		

SetHeaderFont		
const LOGFONT*	pLogFont	ログフォント
int	nPointSize	フォントサイズ
void	戻り値	
行ヘッダーのフォントも変更されます。		

6.7　フォント取得

GetHeaderFont		
LOGFONT*	pLogFont	ログフォント
void	戻り値	
列ヘッダーのフォントを取得します。		

7　行

7.1　追加

InsertItem		
int	iRow	挿入する行のインデックス
LPCTSTR	lpszItem	内容
BOOL	bReadOnly	TRUE：行全体を読み取り専用 FALSE：行全体を編集可能
int	戻り値	新しい行のインデックス
第 3 引数以外、詳細は MSDN の CListCtrl::InsertItem を参照してください。		

InsertItem		
UINT	nMask	行の属性マスク LVIF_TEXT、LVIF_PARAM、LVIF_STATE の組み合わせ
int	iRow	挿入する行のインデックス
LPCTSTR	lpszItem	内容
UINT	nState	行の状態
UINT	nStateMask	行の状態マスク
LPARAM	lParam	行に関連付けられている 32 ビットアプリケーション固有の値
BOOL	bReadOnly	TRUE：行全体を読み取り専用 FALSE：行全体を編集可能
int	戻り値	新しい行のインデックス
第 7 引数以外、詳細は MSDN の CListCtrl::InsertItem を参照してください。		

7.2　削除

DeleteItem		
int	iRow	削除する行のインデックス
BOOL	戻り値	TRUE：削除成功 FALSE：削除失敗
行を削除します。		

7.3　全削除

DeleteAllItems		
BOOL	戻り値	TRUE：削除成功 FALSE：削除失敗
すべての行を削除します。		

7.4　行選択

SelectLine		
int	iRow	行のインデックス -1：全行
BOOL	bSelect	TRUE：選択 FALSE：解除
void	戻り値	
行を選択（解除）します。		

7.5　クリア

ClearRow		
int	iRow	行のインデックス -1：選択行
BOOL	bForce	TRUE：読み取り専用セルでもクリアする FALSE：読み取り専用セルはクリアしない
void	戻り値	
行の内容とセルの色をクリアします。 列の色設定「5.6 色設定」はクリアされません。		

7.6　全クリア

ClearAll		
BOOL	bForce	TRUE：読み取り専用セルでもクリアする FALSE：読み取り専用セルはクリアしない
void	戻り値	
すべての行の内容とセルの色をクリアします。 列の色設定「5.6 色設定」はクリアされません。		

7.7　色設定

「9.13 色設定」の第 2 引数を-1 にして設定します。

7.8　色クリア

「9.16 色クリア」の第 2 引数を-1 にしてクリアします。

7.9　読み取り専用設定

SetRowReadOnly		
int	iRow	行のインデックス -1：すべての行
BOOL	bReadOnly	TRUE：読み取り専用にする FALSE：編集可能にする
void	戻り値	
行に設定されていた、セルの読取り専用設定は解除されます。		

7.10 読み取り専用設定取得

IsRowReadOnly		
int	iRow	行のインデックス
BOOL	戻り値	TRUE：読み取り専用設定している FALSE：読み取り専用設定していない
行に読み取り専用設定しているかチェックします。		

7.11 高さ設定

SetRowHeight		
int	iHeight	高さ（ピクセル） -1：自動
void	戻り値	
行の高さを設定します。		

7.12 高さ取得

GetRowHeight		
int	戻り値	高さ
行の高さを取得します。		

7.13 状態取得

GetLineStatus		
int	iRow	行のインデックス
LINESTATUS	戻り値	LINESTATUS:: NORMAL：下記以外 LINESTATUS::EDITING：編集中（エディットコントロール） LINESTATUS::SELECTING：編集中（コンボボックス） LINESTATUS::NEW：新規入力用
行の状態を取得します。		

7.14 ソート

SortItems
使用方法は MSDN の CListCtrl:: SortItems を参照してください。

SortImtemsEx
使用方法は MSDN の CListCtrl:: SortItemsEx を参照してください。

8　行ヘッダー

8.1　幅設定

SetRowHeaderWidth		
int	iWidth	行ヘッダー幅 -1：行ヘッダー無し（初期値） 最小幅は 3
void	戻り値	
行ヘッダーの幅を設定します。		

8.2　幅取得

GetRowHeaderWidth		
int	戻り値	行ヘッダーの幅 -1：非表示

8.3　テキスト表示位置設定

SetRowHeaderAlign		
UINT	uAlign	テキスト表示位置 DT_LEFT：左寄せ DT_CENTER：中央寄せ DT_RIGHT：右寄せ
void	戻り値	

8.4　テキストモード設定

SetRowHeaderContentsMode		
CONTENTSMODE	contentsMode	テキスト内容モード CONTENTSMODE:: AUTONUMBER：自動連番 CONTENTSMODE::CUSTOM：任意設定（「8.5 テキスト設定」で設定します）
void	戻り値	

8.5　テキスト設定

SetRowHeaderContents		
int	iRwo	行のインデックス
LPCTSTR	lpszContents	内容
BOOL	戻り値	TRUE：設定成功 FALSE：設定失敗

8.6　フォント設定

「6.6 フォント設定」を参照

8.7　再描画

RedrawRowHeader		
void	戻り値	

9　セル

9.1　アクティブセル設定

SetActiveCell		
int	iRow	行のインデックス
int	iCol	列のインデックス
BOOL	bVisible	指定セルが表示されるようにスクロールする。
SELECTSTATUS	selectStatus	選択状態 SELECTSTATUS:: CELL：セル選択 SELECTSTATUS::LINE：行選択
BOOL	戻り値	TRUE：設定成功 FALSE：設定失敗
アクティブセル、またはアクティブ行を設定します。		

9.2　アクティブセル取得

GetActiveCell		
Int*	piRow	行のインデックス
Int*	piCol	列のインデックス
BOOL	bSelected	アクティブセルの判定方法 TRUE：フォーカス＋選択セルをアクティブセルとする FALSE：フォーカスセルをアクティブセルとする
BOOL	戻り値	アクティブセルの有無 TRUE：アクティブセルあり FALSE：アクティブセルなし

9.3　アクティブセルクリア

ClearActiveCell		
Void	戻り値	
セルのフォーカス枠、選択を解除します。 1 行 1 列目にスクロールします。		

9.4　セル状態取得

GetCellState		
int*	iRow	行のインデックス
int*	iCol	列のインデックス
UINT	戻り値	セル状態（以下のいずれか 1 または組み合わせ） CELLST_NONE：選択もフォーカスも無い CELLST_SELECTED：選択 CELLST_FOCUSED：フォーカス

9.5　編集開始

StartEdit		
int	iRow	行のインデックス
int	iCol	列のインデックス
BOOL	戻り値	TRUE：編集開始成功 FALSE：編集開始失敗
指定セルの編集を開始します。 第 1 引数または第 2 引数が-1 の場合、フォーカスがあり選択されているセルの編集を開始します。 セルのダブルクリックまたは F2 キーで編集開始します。		

9.6　編集確定終了

EndEdit		
void	戻り値	
入力中の内容を確定して編集を終了します。		

9.7　編集キャンセル終了

CancelEdit		
void	戻り値	
入力中の内容を破棄して編集を終了します。		

9.8　編集中確認

IsEditing		
BOOL	戻り値	TRUE：編集中 FALSE：編集中ではない

9.9　クリア

ClearCell		
int	iRow	行のインデックス -1：選択行
int	iCol	列のインデックス -1：選択列

BOOL	bForce	TRUE：読み取り専用セルでもクリアする FALSE：読み取り専用セルはクリアしない
void	戻り値	
セルの内容とセルの色をクリアします。		

9.10 リセット

Reset		
void	戻り値	
すべての内容、色設定、読み取り専用設定をクリアします。 行は削除しません。		

9.11 読み取り専用設定

SetCellReadOnly		
int	iRow	行のインデックス
int	iCol	列のインデックス
BOOL	bReadOnly	TRUE：読み取り専用にする FALSE：編集可能にする
void	戻り値	

9.12 読み取り専用設定取得

IsCellReadOnly		
int	iRow	行のインデックス
int	iCol	列のインデックス

BOOL	戻り値	TRUE：読み取り専用 FALSE：編集可能
セルが読み取り専用かチェックします。		

9.13 色設定

SetCellColor		
int	iRow	行のインデックス
int	iCol	列のインデックス -1：すべての列
COLORREF	colText	文字色 設定を変更しない場合は、CLR_NONE を設定します。 設定をクリアする場合は、CLR_DEFAULT を設定します。
COLORREF	colBack	背景色 設定を変更しない場合は、CLR_NONE を設定します。 設定をクリアする場合は、CLR_DEFAULT を設定します。
void	戻り値	

9.14 色取得

GetCellColor		
int	iRow	行のインデックス
int	iCol	列のインデックス
COLORREF*	pcolText	文字色
COLORREF*	pcolBack	背景色
BOOL	戻り値	TRUE：セルの色設定あり FALSE：セルの色設定なし

実際に画面に表示されている色を取得する場合は、「9.15 表示色取得」を使用します。

9.15 表示色取得

GetCellDisplayColor		
int	iRow	行のインデックス
int	iCol	列のインデックス
COLORREF*	pcolText	文字色
COLORREF*	pcolBack	背景色
BOOL	戻り値	TRUE：正常取得 FALSE：取得失敗
画面に表示されている色を取得します。		

9.16 色クリア

ClearCellColor		
int	iRow	行のインデックス
int	iCol	列のインデックス -1：すべての列
void	戻り値	

9.17 フォント設定

SetCellFont		
LPCTSTR	lpszFaceName	フォント名
int	nPointSize	フォントサイズ

BOOL	bBold	太字 TRUE：太字 FALSE：通常
BOOL	bItalic	斜体 TRUE：斜体 FALSE：通常
void	戻り値	
セルのフォントを取得します。		

SetCellFont		
const LOGFONT*	pLogFont	ログフォント
void	戻り値	
セルのフォントを取得します。		

9.18 フォント取得

GetCellFont		
LOGFONT*	pLogFont	ログフォント
void	戻り値	
セルのフォントを取得します。		

10 各種情報の設定/取得

10.1 キー動作設定

SetKeyMotion		
UINT	uTabKey	タブキーの動作 ・現在設定を変更しない KEY_NOCHANGE ・移動方向（いずれか 1 つ） KEY_DIR_LEFT：左移動 KEY_DIR_RIGHT：右移動（初期値） KEY_DIR_DOWN：下移動 KEY_DIR_UP：上移動 ・行（列）端に来たときの行（列）移動（いずれか 1 つ） KEY_NEXT_TRUE：行（列）移動する KEY_NEXT_FALSE：行（列）移動しない（初期値） ・ローテーション移動（いずれか 1 つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値） KEY_ROT_FOUT：次のコントロールに移動する
UINT	uEnterKey	Enter キー動作 ・現在設定を変更しない KEY_NOCHANGE ・移動方向（いずれか 1 つ） KEY_DIR_LEFT：左移動 KEY_DIR_RIGHT：右移動 KEY_DIR_DOWN：下移動（初期値） KEY_DIR_UP：上移動

		・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する KEY_NEXT_FALSE：行（列）移動しない（初期値） ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT	uUpKey	上キー動作 ・現在設定を変更しない KEY_NOCHANGE ・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT	uDownKey	下キー動作 ・現在設定を変更しない KEY_NOCHANGE ・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT	uLeftKey	左キー動作 ・現在設定を変更しない KEY_NOCHANGE ・行（列）端に来たときの行（列）移動（いずれか１つ）

		KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT	uRightKey	右キー動作 ・現在設定を変更しない KEY_NOCHANGE ・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
void	戻り値	

```
m_pList->SetKeyMotion(    KEY_DIR_RIGHT | KEY_NEXT_FALSE | KEY_ROT_FALSE,
                          KEY_DIR_DOWN | KEY_NEXT_FALSE | KEY_ROT_FALSE,
                          KEY_NOCHANGE,
                          KEY_NEXT_TRUE | KEY_ROT_FALSE,
                          KEY_NEXT_TRUE | KEY_ROT_FALSE,
                          KEY_NEXT_TRUE | KEY_ROT_FALSE);
```

10.2 キー動作取得

GetKeyMotion		
UINT*	puTabKey	タブキーの動作 ・移動方向（いずれか１つ） KEY_DIR_LEFT：左移動 KEY_DIR_RIGHT：右移動（初期値） KEY_DIR_DOWN：下移動

		KEY_DIR_UP：上移動 ・行（列）端に来たときの行（列）移動（いずれか 1 つ） KEY_NEXT_TRUE：行（列）移動する KEY_NEXT_FALSE：行（列）移動しない（初期値） ・ローテーション移動（いずれか 1 つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値） KEY_ROT_FOUT：次のコントロールに移動する
UINT*	puEnterKey	Enter キー動作 ・移動方向（いずれか 1 つ） KEY_DIR_LEFT：左移動 KEY_DIR_RIGHT：右移動 KEY_DIR_DOWN：下移動（初期値） KEY_DIR_UP：上移動 ・行（列）端に来たときの行（列）移動（いずれか 1 つ） KEY_NEXT_TRUE：行（列）移動する KEY_NEXT_FALSE：行（列）移動しない（初期値） ・ローテーション移動（いずれか 1 つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT*	puUpKey	上キー動作 ・行（列）端に来たときの行（列）移動（いずれか 1 つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか 1 つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT*	puDownKey	下キー動作

		・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT*	puLeftKey	左キー動作 ・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
UINT*	puRightKey	右キー動作 ・行（列）端に来たときの行（列）移動（いずれか１つ） KEY_NEXT_TRUE：行（列）移動する（初期値） KEY_NEXT_FALSE：行（列）移動しない ・ローテーション移動（いずれか１つ） KEY_ROT_TRUE：ローテーションする KEY_ROT_FALSE：ローテーションしない（初期値）
void	戻り値	
UINT uTabKey, uEnterKey, uUpKey, uDownKey, uRightKey; m_pList->GetKeyMotion(&uTabKey, &uEnterKey, &uUpKey, &uDownKey, NULL, &uRightKey);		

10.3 編集モード設定

SetEditMode

EDITMODE	eEditMode	編集モード EDITMODE:: EDIT：変更モード（ユーザー操作による行追加不可）（初期値） EDITMODE::ADD：追加モード
void	戻り値	
m_pList->SetEditMode(EDITMODE::ADD);		

10.4 編集モード取得

GetEditMode		
EDITMODE	戻り値	EDITMODE:: EDIT：変更モード（ユーザー操作による行追加不可） EDITMODE::ADD：追加モード
EDITMODE editMode = m_pList->GetEditMode();		

10.5 選択状態取得

GetSelectStatus		
SELECTSTATUS	戻り値	選択状態 SELECTSTATUS:: CELL：セル選択 SELECTSTATUS::LINE：行選択
現在の選択状態を取得します。		
SELECTSTATUS selectStatus = m_pList->GetSelectStatus();		

10.6 行選択可否設定

EnableLineSelect		
BOOL	bEnable	TRUE：行選択可能（初期値） FALSE：行選択不可
void	戻り値	

マウス、キー操作による行選択の可否を設定します。

10.7 変更状態取得

IsModify		
BOOL	戻り値	TRUE：変更あり FALSE：変更なし コードによる内容の変更・行追加・行削除では、変更状態は更新されません。 編集コントロールによる編集が確定した場合のみ、変更状態が更新されます。

10.8 変更状態設定

SetModify		
BOOL	bModify	変更状態 TRUE：変更あり FALSE：変更なし
void	戻り値	

10.9 編集開始時カーソル状態設定

SetEditCursor		
EDITCURSOR	eEditCursor	カーソル状態 EDITCURSOR:: LEFT：左端にカーソルを置く（初期値） EDITCURSOR:: RIGHT：右端にカーソルを置く EDITCURSOR::SELECT：全選択
void	戻り値	
m_pList->SetEditCursor(EDITCURSOR::LEFT);		

10.10　編集開始時カーソル状態取得

GetEditCursor		
EDITCURSOR	戻り値	カーソル状態 EDITCURSOR:: LEFT：左端にカーソルを置く EDITCURSOR:: RIGHT：右端にカーソルを置く EDITCURSOR::SELECT：全選択
EDITCURSOR editCursor = m_pList->GetEditCursor(); 		

10.11　基本色取得

GetBaseColor		
COLOR_KEY	ColorKey	取得する色の種類 COLOR_KEY:: BACK：セルの背景色 COLOR_KEY:: BACK_READONLY：読み取り専用セルの背景色 COLOR_KEY:: BACK_SELECT：選択セルの背景色（アクティブ時） COLOR_KEY:: BACK_SELECT_LOST：選択セルの背景色（非アクティブ時） COLOR_KEY:: BORDER：罫線色 COLOR_KEY::TEXT：文字色 COLOR_KEY::TEXT_READONLY：読み取り専用セルの文字色 COLOR_KEY::TEXT_SELECT：選択セルの文字色（アクティブ時） COLOR_KEY::TEXT_SELECT_LOST：選択セルの文字色（非アクティブ時） COLOR_KEY::EDIT_BORDER：編集コントロール境界線色
COLORREF	戻り値	色
COLORREF backColor = m_pList->GetBaseColor(COLOR_KEY::BACK); 		

10.12　基本色設定

SetBaseColor		
COLOR_KEY	ColorKey	設定する色の種類 「10.11 基本色取得」を参照
COLORREF	color	設定する色 CLR_DEFAULT を設定するとデフォルトの色になります。
void	戻り値	
m_pList->SetBaseColor(COLOR_KEY::BACK, RGB(255, 255, 0));		

10.13　コントロール矩形取得

GetWindowRectWithRowHeader		
LPRECT	lpRect	スクリーン座標
void	戻り値	
行ヘッダーを含むコントロール全体のスクリーン座標を取得します。		
CRect rect; m_pList->GetWindowRectWithRowHeader(&rect);		

10.14　コントロール移動

MoveWindowWithRowHeader		
LPRECT	lpRect	移動先のクライアント座標
BOOL	bRepaint	TRUE：再描画する FALSE：再描画しない。
void	戻り値	
行ヘッダーを含むコントロール全体を移動する。		

```
CRect rect(10, 10, 500, 400);
m_pList->MoveWindowWithRowHeader(&rect);
```

10.15 コントロールサイズ変更

Fit

int	iWidth	幅
int	iHeight	高さ
void	戻り値	

行ヘッダーを含むコントロール全体を指定したサイズに変更します。

```
CRect rect;
GetClientRect(&rect);
m_pList->Fit(rect.Width() - 24, rect.Height() - 90);
```

FitRBMargin

int	iRightMargin	右マージン
int	iBottomMargin	下マージン
void	戻り値	

行ヘッダーを含むコントロール全体を親ウィンドウに対して、指定したマージンになるようにサイズを変更します。

```
m_pList->FitRBMargin(12, 90);
```

10.16 ヒットテスト

HitTestEx

CPoint	pt	テストするスクリーン座標

LPLVNEXHITTEST	info	結果 ・セル上の場合 info.iRow：行インデックス（0 以上） info.iCol：列インデックス（0 以上） info.item：HITTESTEXITEM::CELL ・列ヘッダ上ーの場合 info.iRow：-1 info.iCol：列インデックス（0 以上） info.item：HITTESTEXITEM:: COLHEADER ・行ヘッダー上の場合 info.iRow：行インデックス（-1 以上） info.iCol：-1 info.item：HITTESTEXITEM:: ROWHEADER
BOOL	戻り値	TRUE：結果あり FALSE：結果なし
指定した座標の情報を取得します。		

10.17 キャンセルボタン設定

SetCancelWnd		
CWnd*	pWnd	編集キャンセルするウィンドウ
BOOL	bAdd	TRUE：追加する FALSE：削除する
void	戻り値	
セル編集中に閉じるボタンなど押された場合に、編集をキャンセルするため。		
m_pList->SetCancelWnd(GetDlgItem(IDCANCEL));		

10.18 キャンセルボタン設定クリア

ClearCancelWnd		
void	戻り値	
キャンセルボタン設定をクリアします。		

10.19 終了設定

Finish		
void	戻り値	
セル編集中にウィンドウの閉じるボタンやフォーカスを受け取らないコントロールのクリックやキー操作で、ウィンドウを閉じようとしたときに、編集をキャンセルする場合に呼び出します。		
m_pList->Finish();		

10.20 バージョン番号取得

GetVersion		
CString	戻り値	製品バージョン番号
static 関数です。		
CString csVersion; csVersion = CListCtrlEx::GetVersion();		

11 コールバック関数

11.1 編集開始時

SetBeforeEditFunc		
PFBEFOREEDIT	pfBeforeEdit	編集開始時に呼ぶコールバック関数
LPVOID	pParam	コールバック関数に渡されるポインタ
void	戻り値	
編集開始時に呼ばれるコールバック関数を登録します。 編集開始時に編集コントロールに初期設定する値を編集したい場合に使用します。 セルの値が「1,234」のとき、編集コントロールに「1234」を設定したい場合など。 エディットコントロール列のみ呼ばれます。		
`m_pList->SetBeforeEditFunc(OnBeforeEdit, this);`		

PFBEFOREEDIT（コールバック関数の形式）		
CListCtrlEx*	pList	リストコントロール
int	iRow	行のインデックス
int	iCol	列のインデックス
const CString&	csSrc	編集前の値
CString&	csDst	エディットコントロールに設定する値
LPVOID	pParam	SetBeforeEditFunc で設定されたポインタ
BOOL	戻り値	TRUE：csDst に設定した値で編集を開始する FALSE：編集前の値で編集を開始する
クラスの静的メンバーであるか、任意のクラスのメンバーではないスタンドアロン関数である必要があります。		

11.2 編集終了時

SetAfterEditFunc		
PFAFTEREDIT	pfAfterEdit	編集終了時に呼ぶコールバック関数
LPVOID	pParam	コールバック関数に渡されるポインタ
void	戻り値	
編集終了時に呼ばれるコールバック関数を登録します。 編集終了時にチェック/編集等を行いたい場合に使用します。 編集しないで編集終了したときは呼ばれません。 チェックボックス列は呼ばれません。		
`m_pList->SetAfterEditFunc(OnAfterEdit, this);`		

PFAFTEREDIT（コールバック関数の形式）		
CListCtrlEx*	pList	リストコントロール
int	iRow	行のインデックス
int	iCol	列のインデックス
const CString&	csSrc	入力された値
CString&	csDst	セルに設定する値
LPVOID	pParam	SetAfterEditFunc で設定されたポインタ
BOOL	戻り値	TRUE：csDst に設定した値で編集を確定する FALSE：編集をキャンセルする
クラスの静的メンバーであるか、任意のクラスのメンバーではないスタンドアロン関数である必要があります。		

11.3 Enter,Tab キーによる編集終了後

SetKeyEndEditFunc

PFKEYENDEDIT	pfKeyEndEdit	Enter,Tab キーによる編集終了後に呼ぶコールバック関数
LPVOID	pParam	コールバック関数に渡されるポインタ
void	戻り値	
Enter,Tab キーによる編集終了後に呼ばれるコールバック関数を登録します。 Enter,Tab キーによる編集終了後のセル移動などを行いたい場合に使用します。(10.1 キー動作以外の移動をしたいとき)		
`m_pList->SetKeyEndEditFunc(OnKeyEndEdit, this);`		

PFKEYENDEDIT（コールバック関数の形式）		
CListCtrlEx*	pList	リストコントロール
int	iRow	行のインデックス
int	iCol	列のインデックス
int	iKeyCode	VK_RETURN：Enter キーによる編集終了 VK_TAB：Tab キーによる編集終了
BOOL	bShiftKey	TRUE：シフトキーが押されている FALSE：シフトキーが押されていない
LPVIOD	pParam	SetKeyEndEditFunc で設定されたポインタ
BOOL	戻り値	TRUE：10.1 キー動作で指定したセル移動を行う FALSE：セル移動を行わない（コールバック関数内でセル移動を行いたい場合）
クラスの静的メンバーであるか、任意のクラスのメンバーではないスタンドアロン関数である必要があります。		

11.4 クリックによる編集終了後

SetClickCellEndEditFunc		
PFCLICKCELLENDEDIT	pfClickCellEndEdit	クリック（左右）による編集終了後に呼ぶコールバック関数
LPVOID	pParam	コールバック関数に渡されるポインタ
void	戻り値	

セル上、ヘッダー上のクリックによる編集終了後に呼ばれるコールバック関数を登録します。 クリックによる編集終了後のセル移動などを行いたい場合に使用します。
m_pList->SetClickCellEndEditFunc(OnClickCellEndEdit, this);

PFCLICKCELLENDEDIT（コールバック関数の形式）		
CListCtrlEx*	pList	リストコントロール
int	iRow	行のインデックス
int	iCol	列のインデックス
int	iClickRow	クリックされた行インデックス 列ヘッダーをクリックされた場合は-1
int	iClickCol	クリックされた列インデックス 行ヘッダーをクリックされた場合は-1
BOOL	bLeft	TRUE：左クリック FALSE：右クリック
LPVIOD	pParam	SetClickCellEndEditFunc で設定されたポインタ
BOOL	戻り値	TRUE：クリックしたセルに移動を行う FALSE：セル移動を行わない（コールバック関数内でセル移動を行いたい場合）
クラスの静的メンバーであるか、任意のクラスのメンバーではないスタンドアロン関数である必要があります。		

11.5 文字入力時

SetPressCharFunc		
PFPRESSCHAR	pfPressChar	文字入力時に呼ぶコールバック関数
LPVOID	pParam	コールバック関数に渡されるポインタ
void	戻り値	

文字入力時に呼ばれるコールバック関数を登録します。 文字制限や文字変換を行いたい場合に使用します。 編集コントロールにテキストを貼り付けた場合にも、文字数分コールバック関数が呼ばれます。
`m_pList-> SetPressCharFunc(OnPressChar, this);`

PFPRESSCHAR（コールバック関数の形式）		
CListCtrlEx*	pList	リストコントロール
int	iRow	行のインデックス
int	iCol	列のインデックス
const WCHAR	wSrc	入力された文字
WCHAR&	wDst	編集コントロールに設定する文字 初期値：wSrc
LPVIOD	pParam	SetPressCharFunc で設定されたポインタ
BOOL	戻り値	TRUE：wDst に設定された文字を編集コントロールに設定する FALSE：入力された文字をキャンセルする。
クラスの静的メンバーであるか、任意のクラスのメンバーではないスタンドアロン関数である必要があります。		

11.6 文字入力後

SetAfterPressCharFunc		
PFAFTERPRESSCHAR	pfAfterPressChar	文字入力後に呼ぶコールバック関数
LPVOID	pParam	コールバック関数に渡されるポインタ
void	戻り値	
文字入力後（編集コントロールへの反映前）に呼ばれるコールバック関数を登録します。 文字数制限や入力形式のチェックを行いたい場合に使用します。 文字が入力される度に呼ばれ、編集コントロールに反映後に想定される文字列が、コールバック関数に渡されます。(csCheckText) 編集コントロールにテキストを貼り付けた場合にも、文字数分コールバック関数が呼ばれます。		

```
m_pList-> SetEndPressCharFunc(OnEndPressChar, this);
```

PFAFTERPRESSCHAR（コールバック関数の形式）		
CListCtrlEx*	pList	リストコントロール
int	iRow	行のインデックス
int	iCol	列のインデックス
const CString&	csCheckText	チェックする文字列
LPVIOD	pParam	SetAfterPressCharFunc で設定されたポインタ
BOOL	戻り値	TRUE：csCheckText を編集コントロールに反映する FALSE：csCheckText を編集コントロールに反映しない
クラスの静的メンバーであるか、任意のクラスのメンバーではないスタンドアロン関数である必要があります。		

12 通知

12.1 行追加編集通知

セル編集、行の追加時に通知します。

通知コード		LVNEX_CHANGEROW
通知構造体	LPLVNEXNOTIFY	
NMHDR	hdr	
int	iRow	行のインデックス
int	iCol	列のインデックス
UINT	uAction	NTACTION_EDIT：セルの編集 NTACTION_ADD：行の追加
DWORD	dwTextLength	未使用
WCHAR	cText[1]	未使用

編集終了後に通知されます。

```
afx_msg void OnChangeRow(NMHDR* pNotifyStruct, LRESULT* result);

ON_NOTIFY(LVNEX_CHANGEROW, IDC_LIST, &CxxxDlg::OnChangeRow)

void CxxxDlg::OnChangeRow(NMHDR* pNotifyStruct, LRESULT* result)
{
	LPLVNEXNOTIFY phdr = (LPLVNEXNOTIFY)pNotifyStruct;

	// TODO

	*result = 0;
}
```

12.2 Delete キー押下通知

Delete キー押下時に通知します。

通知コード		LVNEX_DELETEKEY
通知構造体	NMHDR	
NMHDR	hdr	
int	iRow	未使用
int	iCol	未使用
UINT	uAction	NTACTION_DEL
size_t	TextLength	未使用
TCHAR	Text[1]	未使用

```
afx_msg void OnDeleteKey(NMHDR* pNotifyStruct, LRESULT* result);

ON_NOTIFY(LVNEX_DELETEKEY, IDC_LIST, &CxxxDlg::OnDeleteKey)

void CxxxDlg::OnDeleteKey(NMHDR* pNotifyStruct, LRESULT* result)
{
	LPLVNEXNOTIFY phdr = (LPLVNEXNOTIFY)pNotifyStruct;

	// TODO

	*result = 0;
}
```

12.3 Ctrl+C キー押下時通知

Ctrl+C キー押下時に通知します。

通知コード		LVNEX_COPYKEY
通知構造体	LPLVNEXNOTIFY	
NMHDR	hdr	
int	iRow	未使用
int	iCol	未使用
UINT	uAction	NTACTION_COPY
size_t	TextLength	未使用

TCHAR	Text[1]	未使用

```
afx_msg void OnCopyKey(NMHDR* pNotifyStruct, LRESULT* result);

ON_NOTIFY(LVNEX_COPYKEY, IDC_LIST, &CxxxDlg::OnCopyKey)

void CxxxDlg::OnCopyKey(NMHDR* pNotifyStruct, LRESULT* result)
{
	LPLVNEXNOTIFY phdr = (LPLVNEXNOTIFY)pNotifyStruct;

	// TODO

	*result = 0;
}
```

12.4 Ctrl+V キー押下時通知

Ctrl+V キー押下時に通知します。
クリップボードにテキストデータが格納されていない場合は、通知されません。

通知コード		LVNEX_PASTEKEY
通知構造体	LPLVNEXNOTIFY	
NMHDR	hdr	
int	iRow	未使用
int	iCol	未使用
UINT	uAction	NTACTION_PASTE
size_t	TextLength	格納文字数
TCHAR	Text[1]	クリップボードに格納されている文字列 最後は NULL

```
afx_msg void OnPasteKey(NMHDR* pNotifyStruct, LRESULT* result);

ON_NOTIFY(LVNEX_PASTEKEY, IDC_LIST, &CxxxDlg::OnPasteKey)

void CxxxDlg::OnPasteKey(NMHDR* pNotifyStruct, LRESULT* result)
{
	LPLVNEXNOTIFY phdr = (LPLVNEXNOTIFY)pNotifyStruct;
```

```
        // TODO

        *result = 0;
}
```

12.5 アクティブセル変更通知

マウス・キー操作によるアクティブセル変更時に通知します。

通知コード		LVNEX_ACTIVECELL
通知構造体	NMHDR	
NMHDR	hdr	
int	iRow	行のインデックス
int	iCol	列インデックス
UINT	uAction	NTACTION_ACTIVECELL
size_t	TextLength	未使用
TCHAR	Text[1]	未使用

```
afx_msg void OnActiveCell(NMHDR* pNotifyStruct, LRESULT* result);

ON_NOTIFY(LVNEX_ACTIVECELL, IDC_LIST, &CxxxDlg::OnActiveCell)

void CxxxDlg::OnPasteKey(NMHDR* pNotifyStruct, LRESULT* result)
{
        LPLVNEXNOTIFY phdr = (LPLVNEXNOTIFY)pNotifyStruct;

        // TODO

        *result = 0;
}
```

12.6 右クリック通知

右クリック時に通知します。

通知コード		LVNEX_RCLICK

通知構造体	NMHDR	
NMHDR	hdr	
int	iRow	行のインデックス
int	iCol	列インデックス
UINT	uAction	NTACTION_CELL：セルの右クリック NTACTION_COLHEADER：列ヘッダーの右クリック NTACTION_ROWHEADER：行ヘッダーの右クリック
size_t	TextLength	未使用
TCHAR	Text[1]	未使用

```
afx_msg void OnRClick(NMHDR* pNotifyStruct, LRESULT* result);

ON_NOTIFY(LVNEX_RCLICK, IDC_LIST, &CxxxDlg::OnRClick)

void CxxxDlg::OnRClick(NMHDR* pNotifyStruct, LRESULT* result)
{
	LPLVNEXNOTIFY phdr = (LPLVNEXNOTIFY)pNotifyStruct;

	// TODO

	*result = 0;
}
```

13 その他

13.1 列ヘッダーのクリックイベント取得

```
afx_msg void OnItemclickList(NMHDR* pNMHDR, LRESULT* pResult);

BEGIN_MESSAGE_MAP(CxxxDlg, CDialogEx)
	ON_NOTIFY(HDN_ITEMCLICK, 0, &CxxxDlg::OnItemclickList)
END_MESSAGE_MAP()

void CxxxDlg::OnItemclickList(NMHDR* pNMHDR, LRESULT* pResult)
{
	LPNMHEADER phdr = reinterpret_cast<LPNMHEADER>(pNMHDR);

	if (phdr->hdr.hwndFrom == m_pList->GetHeaderCtrl()->GetSafeHwnd()) {
		CString cs;
		cs.Format(_T("列:%d"), phdr->iItem);
		AfxMessageBox(cs);
	}

	*pResult = 0;
}
```

13.2 選択セル/選択行の取得

```
int iCol;
int iRow;

if (m_pList->GetSelectStatus() == SELECTSTATUS::CELL) {
	// セル選択状態
	if(m_pList->GetActiveCell(&iRow, &iCol, TRUE)){
		// iRow行,iCol列が選択されている
	}
}
else {
	// 行選択状態
	iRow = m_pList->GetNextItem(-1, LVNI_SELECTED);
	while (iRow != -1) {
		// iRow行が選択されている
		iRow = m_pList->GetNextItem(iRow, LVNI_SELECTED);
	}
```

```
}
```

13.3 動的にライブラリを読み込む

```
HINSTANCE m_hInstance;
Koei::CListCtrlEx* m_pList;

using namespace Koei;
m_hInstance = AfxLoadLibrary("ListCtrlEx.dll");
PFINSTANCE pfInstance = (PFINSTANCE)GetProcAddress(m_hInstance, "Instance");
m_pList = (*pfInstance)(_T("ライセンスキー"));
CRect rect(10, 10, 100, 100);
m_pList->Create(WS_VISIBLE | WS_TABSTOP, rect, this, 5000);
・・・

// 終了時
delete m_pList;
AfxFreeLibrary(m_hInstance);
```

以上。