Wide Studio
Class Reference

目次

A クラスリファレンス

1 WSCballoonHelp

1.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow

1.2 概要

バルーンヘルプを表示したいオブジェクトをこのバルーンヘルプに登録しておくと、マウスがオブジェクト上にマウスが動かされた時に、登録時に指定しておいた文字列でバルーンヘルプが表示されます。

1.3 機能

- バルーンヘルプを表示したい登録されたオブジェクトを記憶します。
- 登録されたオブジェクト上にマウスポインタが移動するとバルーンヘルプを表示します。

1.4 注意事項

1.5 プロパティー覧

クラス WSCballoonHelp のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNlabelString	表示文字列	
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	6
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	4
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	

unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

1.6 トリガー一覧

WSCballoonHelp クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

1.7 関数一覧

- WSCballoonHelp* WSGIappBalloonHelp() :WSCballoonHelp
- long registerClient(WSCbase*,char*) :WSCballoonHelp
- long unregisterClient(WSCbase*) :WSCballoonHelp

1.8 関数仕様

1.8.1 WSGIappBalloonHelp 関数の説明

書式 WSCballoonHelp* WSGIappBalloonHelp()

機能 デフォルトのバルーンヘルプオブジェクトを返します。

処理概要 デフォルトのバルーンヘルプオブジェクトを供給します。バルーンヘルプユーザが、自分でバルーンヘルプのインスタンスを生成する手間を省きます。

引数 なし。

復帰値 バルーンヘルプインスタンス。

注意事項 使用例：
バルーンヘルプの使用例です。初期化トリガなどで張り付けたイベントプロシージャ等で、バルーンヘルプを表示したいオブジェクトを登録します。

サンプル

```
...
#include "WSCballoonHelp.h"
...
void init_procedure(WSCbase* object){
  WSCballoonHelp* bhelp = WSGIappBalloonHelp();
  bhelp->registerClient(object,"バルーンヘルプ！");
  ...
}
```

1.8.2 registerClient 関数の説明

書式 long registerClient(WSCbase* client,char* str)

機能 バルーンヘルプを表示したいオブジェクトを指定します。

処理概要 指定したオブジェクト、文字列を記憶し、そのオブジェクト上にマウスポインターが移動した場合に、バルーンヘルプが表示される様にします。

引数

(in)WSCbase* client	バルーンヘルプを表示したいオブジェクト
(in)char* str	バルーンヘルプに表示される文字列

復帰値 WS_NO_ERR = 正常、それ以外はエラー。

注意事項 表示文字列を変更する場合は、一度登録を解除して、再び新しい表示文字列で登録してください。
なお、登録したオブジェクトをデリートする場合は、登録を解除してから行ってください。

サンプル WSGIappBalloonHelp() を参照してください。

1.8.3 unregisterClient 関数の説明

書式 long unregisterClient(WSCbase* client)

機能 登録したオブジェクトの登録を解除します。

処理概要 記憶していたオブジェクト、文字列の登録を解除します。

引数

(in)WSCbase* client	登録を解除したいオブジェクト

復帰値 WS_NO_ERR = 正常、それ以外はエラー。

注意事項 なし。

サンプル

```
...
#include "WSCballoonHelp.h"
...
void init_procedure(WSCbase* object){
  WSCballoonHelp* bhelp = WSGIappBalloonHelp();
  bhelp->unregisterClient(object);
  ...
}
```

2 WSCbase

2.1 継承元

次のオブジェクトを継承しています。

2.2 概要

ＷｉｄｅＳｔｕｄｉｏ用オブジェクトの基底となるクラスです。ＷｉｄｅＳｔｕｄｉｏアプリケーションはオブジェクトの基本クラスとして、この WSCbase 型でオブジェクトの管理を行います。

2.3 機能

- ＷｉｄｅＳｔｕｄｉｏ用オブジェクトの基本的なＡＰＩの提供
- プロパティの機構の提供
- イベントプロシージャの機構の提供
- クラス型の管理

2.4 注意事項

2.5 プロパティー覧

クラス WSCbase のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値

2.6 トリガー覧

WSCbase クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE

2.7 関数一覧

- long initialize() :WSCbase
- WSCbool getInitialized() :WSCbase
- char* getInstanceName() :WSCbase
- char* getClassName() :WSCbase
- void* cast(char*) :WSCbase
- void setInstanceName(char*) :WSCbase
- WSCbool setProperty(char*,const WSCvariant &) :WSCbase
- WSCvariant getProperty(char*) :WSCbase
- void setVisible(WSCbool) :WSCbase
- WSCbool getVisible() :WSCbase
- void setSensitive(WSCbool) :WSCbase
- WSCbool getSensitive() :WSCbase
- long draw() :WSCbase
- long redraw() :WSCbase
- long update() :WSCbase
- WSClistData & getChildren() :WSCbase
- long getAllChildren(WSClistData &) :WSCbase
- void execProcedure(long trigger) :WSCbase
- void execProcedure(char* procedure_name) :WSCbase
- long setFocus(WSCbool fl = True) :WSCbase
- WSCbool getFocus() :WSCbase
- virutal void onMouseIn(WSCpoint*) :WSCbase
- virutal void onMouseOut() :WSCbase
- virutal void onMouseMove(WSCpoint*) :WSCbase
- virutal void onMousePress(WSCpoint*) :WSCbase
- virutal void onMouseRelease(WSCpoint*) :WSCbase
- virutal void onExpose(WSCrect*) :WSCbase
- virutal void onResize(WSCrect*) :WSCbase
- virutal void onVisibleChange(WSCbool) :WSCbase
- virutal void onParentVisibleChange(WSCbool) :WSCbase
- virutal void onSensitiveChange(WSCbool) :WSCbase
- virutal void onParentSensitiveChange(WSCbool) :WSCbase
- virutal void onChildAdded(WSCbase*) :WSCbase
- virutal void setData(WSCvariant* ,long) :WSCbase
- void setVariantData(char*,WSCvariant) :WSCbase
- WSCvariant &getVariantData(char*) :WSCbase
- virutal void onFocusChange(WSCbool) :WSCbase
- virutal void onSpecialFocusChange(WSCbool) :WSCbase
- virutal void onSelectionChange(WSCbool) :WSCbase
- WSCbase* getParent() :WSCbase
- WSCbase* getParentWindow() :WSCbase
- void setInternalObject(WSCbool) :WSCbase
- WSCbool getInternalObject() :WSCbase
- virtual void setScaleOffsetPtr(double*) :WSCbase
- virtual void setXOffsetPtr(short*) :WSCbase
- virtual void setYOffsetPtr(short*) :WSCbase
- double* getScaleOffsetPtr() :WSCbase
- short* getXOffsetPtr() :WSCbase
- short* getYOffsetPtr() :WSCbase
- long isDefaultValue(char*,WSCbool*) :WSCbase
- WSCbool setDefaultValue(char*) :WSCbase
- WSCbool existProperty(char*) :WSCbase
- long setSpecialFocus(WSCbool fl = True) :WSCbase
- WSCbool getSpecialFocus() :WSCbase
- WSCbase* getChildInstance(char*) :WSCbase
- WSCbase* getFocusMoveInstance(long) :WSCbase
- void needUpdate() :WSCbase
- WSCbool isNeedUpdate() :WSCbase
- WSCbool isParent(WSCbase*) :WSCbase
- WSCbool existTrigger(long) :WSCbase
- WSCbool getMouseAddr(short*,short*) :WSCbase
- virtual long addProcedure(WSCprocedure*) :WSCbase
- long delProcedure(WSCprocedure*) :WSCbase
- WSDdev* getdev() :WSCbase
- virtual WSCbase* getPropertyInheritChild() :WSCbase

2.8 関数仕様

2.8.1 initialize 関数の説明

書式 long initialize()

機能 オブジェクトの初期化を行います。

処理概要 オブジェクトを生成した直後に、この関数を呼び出して、一度だけ オブジェクトを初期化します。

引数 なし。

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 オブジェクトの生成後、この関数を一度だけ呼び出してオブジェクトを初期化します。複数回実行してはいけません。

サンプル

```
WSCbase* create_proc(WSCbase* parent){
   WSCbase* inst = new  WSCdialog(parent,"newwin000");
   inst->initialize();
   inst->setPropertyV(WSNname,"newwin000");
   inst->setPropertyV(WSNtitleString,"title1");
   inst->setPropertyV(WSNvis,(WSCbool)1);
   return inst;
}
```

2.8.2 getInitialized 関数の説明

書式 WSCbool getInitialized()

機能 オブジェクトの初期化状態を取得します。

処理概要 オブジェクトが正しく初期化されているかどうかを、取得します。

引数 なし。

復帰値 True= 初期化済み、False=未初期化。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  if (object->getInitialized() != False){
    //object は initialize() を呼び出され、初期化されています。
  }else{
    //object is not initialized..
  }
}
```

2.8.3 getInstanceName 関数の説明

書式 char* getInstanceName()

機能 オブジェクトの名称を取得します。

処理概要

引数 なし。

復帰値 オブジェクトの名称

注意事項 返値された値を開放してはいけません。また、取得された値は、setInstanceName() 関数の呼び出しで、無効となります。

サンプル

```
void sample_proc(WSCbase* object){
  char* iname = object->getInstanceName();
  printf("instance name=%s\n",iname);

  char* iname2 = object->getInstanceName();
  object->setInstanceName("new-name");
  //間違い! iname2 は setInstanceName によって、ポインタが無効になっている。
  printf("instance name=%s\n",iname2);

  //WSCstring にインスタンス名を格納することで、
  //setInstaceName() でも値が無効になりません。
```

```
  WSCstring iname3;
  iname3 = object->getInstanceName();
  printf("instance name=%s\n",(char*)iname3);
}
```

2.8.4 setInstanceName 関数の説明

書式 void setInstanceName(char*)

機能 オブジェクトの名称を設定します。

処理概要

引数

(in)char* pname	インスタンス名称

復帰値 なし。

注意事項

サンプル getInstanceName() を参照してください。

2.8.5 getClassName 関数の説明

書式 char* getClassName()

機能 オブジェクトのクラス名称を取得します。

処理概要

引数 なし。

復帰値 クラス名称を返します。

注意事項 返値された値を開放してはいけません。

サンプル

```
void sample_proc(WSCbase* object){
  char* cname = object->getClassName();
  printf("class name=%s\n",cname);
}
```

2.8.6 cast 関数の説明

書式 void* cast(char* class_name)

機能 抽象化されてしまったポインタを具体化するための逆キャストの機能を提供します。C++ 言語では、通常、ダイナミックキャスト機能をサポートしていないコンパイラの場合、ポインタを一度、サブクラスにキャストしてしまうと、元のクラスのポインタに戻せませんが、それを可能にします。

処理概要

引数

(in)char* class_name	取得したいクラスの名称

復帰値 指定されたクラスのポインタ

注意事項 返されたポインタを次のようにキャストして使用します。なお、インスタンスと関係の無いクラス名称を与えられると NULL が返されます。

サンプル

```
void sample_proc(WSCbase* object){
  // WSCvlabel* label = (WSCvlabel*)object; //これはコンパイルエラー
  WSCvlabel* label = (WSCvlabel*)object->cast("WSCvlabel");
  if (label == NULL){
    //object が、もともと WSCvlabel クラスとは関係の無いオブジェクトの場合、
    //NULL が返されます。
  }else{
    //キャスト成功。
    //object が、指定したクラスを継承した物である場合は、正しいポインタが
    //返されます。
  }
}
```

2.8.7 setProperty 関数の説明

書式 WSCbool setProperty(char* pname,const WSCvariant &)

機能 指定されたプロパティに値を設定します。

処理概要 与えられたプロパティ名称でプロパティを検索し、見付かったプロパティに値を設定します。

引数

(in)char* pname	プロパティ名称
(in)WSCvariant & value	プロパティの値

復帰値 プロパティが存在し、正しく設定されると True、そうでない場合、False。

注意事項 第２引数は、WSCvariant にキャストされる型ならばなんでも可です。

サンプル

```
void sample_proc(WSCbase* object){
  long  val1 = 1;
  object->setProperty(WSNlabelString,val1); //long 値による設定
  char* val2 = "char* string..";
  object->setProperty(WSNlabelString,val2); //文字列による設定
  WSCstring val3 = "WSCstring..";
  object->setProperty(WSNlabelString,val3); //WSCstring 文字列による設定
  WSCvariant val4 = 1;
  object->setProperty(WSNlabelString,val4); //WSCvariant による設定
}
```

2.8.8 getProperty 関数の説明

書式 WSCvariant getProperty(char* pname)

機能 指定されたプロパティの値を取得します。

処理概要 与えられたプロパティ名称でプロパティを検索し、見付かったプロパティの値を返値します。

引数

(in)char* pname	プロパティ名称

復帰値 WSCvariant 型で、プロパティの値が返されます。WSCvariant 型からキャストされる型ならば、なんでも可です。文字列ので値の取得の場合、char* ではなく、WSCstring 型で取得する必要があります。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  long  val1 = object->getProperty(WSNlabelString); //long 値による取得
  printf("val1=%d\n",val1);
  //文字列による取得
  WSCstring val2 = object->getProperty(WSNlabelString); //文字列による取得
  printf("val2=%s\n",(char*)val2);
//やってはいけません。無効ポインタになります。
//  char* val2 = object->getProperty(WSNlabelString);
}
```

2.8.9 setVisible 関数の説明

書式 void setVisible(WSCbool fl)

機能 表示属性を指定します。

処理概要

引数

(in)WSCbool fl	表示属性 True=表示、False=非表示

復帰値 なし。

注意事項 通常のオブジェクトにおいて、親が非表示状態で表示属性を True としても表示されません。

サンプル 表示属性を反転する例です。

```
void sample_proc(WSCbase* object){
  if ( object->getVisible() == False){
    object->setVisible(True);
  }else{
    object->setVisible(False);
  }
}
```

2.8.10 getVisible 関数の説明

書式 WSCbool getVisible()
機能 表示属性を取得します。
処理概要 親を含めて、現在表示状態にあるかどうかを取得します。
引数 なし。
復帰値 現在の表示状態を返値します。
注意事項 WSNvis プロパティとはかならずしも一致しません。
サンプル setVisible() を参照してください。

2.8.11 setSensitive 関数の説明

書式 void setSensitive(WSCbool fl)
機能 操作属性を指定します。
処理概要
引数

(in)WSCbool fl	操作属性 True=操作可、False=操作不可

復帰値 なし。
注意事項 通常のオブジェクトにおいて、親が操作不可状態で表示属性を True としても操作は可になることはありません。
サンプル 選択属性を反転する例です。

```
void sample_proc(WSCbase* object){
  if ( object->getSensitive() == False){
    object->setSensitive(True);
  }else{
    object->setSensitive(False);
  }
}
```

2.8.12 getSensitive 関数の説明

書式 WSCbool getSensitive()
機能 選択属性を取得します。
処理概要 親を含めて、現在選択可能状態にあるかどうかを取得します。
引数 なし。
復帰値 現在の選択属性の状態を返値します。
注意事項 WSNdet プロパティとはかならずしも一致しません。
サンプル setSensitive() を参照してください。

2.8.13 draw 関数の説明

書式 long draw()
機能 描画を行います。
処理概要
引数 なし。
復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項 一度描画されて、再度描画する必要が無い場合、当関数は、描画を実行しません。強制的に描画させたい場合は、メンバ関数 setAbsoluteDraw(True) を実行してください。
サンプル

```
void sample_proc(WSCbase* object){
  //描画の必要がある場合にのみ描画する場合。
  object->draw();
  //強制描画を行う場合。
  object->setAbsoluteDraw(True);
  object->draw();
  //EXPOSE イベントを発生させて、再描画を行う場合。
```

```
  object->redraw();

}
```

2.8.14 redraw 関数の説明

書式 long redraw()
機能 一度クリアして描画を行います。
処理概要
引数 なし。
復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル draw() を参照してください。

2.8.15 update 関数の説明

書式 long update()
機能 必要なら更新処理を行い、再描画を行います。
処理概要 プロパティ等が変化し、再描画が必要な場合、描画を行います。draw や、redraw と異なるところは、単に再描画だけではなく、必要であれば内部のデータ更新も伴うところです。
引数 なし。
復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル

```
void sample_proc(WSCbase* object){
  //更新処理の必要がある場合にのみ更新する場合。
  object->update();
}
```

2.8.16 getChildren 関数の説明

書式 WSClistData & getChildren()
機能 子のリストを取得します。子を管理する機能があるクラスで機能します。
処理概要
引数 なし。
復帰値 子のリスト
注意事項 子オブジェクトのアクセスの方法は、次のように行います。parent は、WSCform クラス、WSCwindow クラスなどの、子の管理機能を持つオブジェクトで getChildren() 関数が機能するものを指します。
サンプル

```
  WSClistData children = parent->getChildren();
  int i;
  int num = children.getNum();
  for(i=0; i < num; i++){
    WSCbase* child = (WSCbase*)children［i］;
    //child に対して何か処理をおこないます。
  }
```

2.8.17 execProcedure 関数の説明

書式 void execProcedure(long trigger)
機能 指定されたトリガで、イベントプロシージャを実行します。
処理概要
引数

(in)long trigger	トリガ

指定可能なトリガには、次のようなものがあります。

定義値	意味
WSEV_INITIALIZE	初期化トリガ
WSEV_DELETE	開放トリガ
WSEV_ACTIVATE	活性化トリガ
WSEV_VALUE_CH	値変化トリガ
WSEV_FOCUS_CH	フォーカス変化トリガ
WSEV_VISIBLE_CH	表示状態変化トリガ
WSEV_SENSITIVE_CH	操作属性変化トリガ
WSEV_PARENT_VISIBLE_CH	親表示状態変化トリガ
WSEV_PARENT_SENSITIVE_CH	親操作属性変化トリガ
WSEV_EXPOSE	露出トリガ
WSEV_RESIZE	サイズ変化トリガ
WSEV_MOUSE_IN	マウス内入トリガ
WSEV_MOUSE_OUT	マウス外出トリガ
WSEV_MOUSE_PRESS	マウスボタン押下トリガ
WSEV_MOUSE_RELEASE	マウスボタン開放トリガ
WSEV_MOUSE_MOVE	マウス移動トリガ
WSEV_KEY_PRESS	キー押下トリガ
WSEV_KEY_RELEASE	キー開放トリガ
WSEV_KEY_HOOK	キー搾取トリガ
WSEV_SORT	ソートトリガ
WSEV_GUI_POLICY_CH	ルックアンドフィール変化トリガ
WSEV_ITEM_SELECTED	項目選択トリガ
WSEV_STATUS_CH	状態変化トリガ
WSEV_INPUT_FIXED	入力確定トリガ
WSEV_BEGIN	先頭移動トリガ
WSEV_END	末尾移動トリガ
WSEV_INCREMENT	値増トリガ
WSEV_DECREMENT	値減トリガ
WSEV_PAGE_INCREMENT	値ページ増トリガ
WSEV_PAGE_DECREMENT	値ページ減トリガ

(注意) クラスによって、トリガの意味は多少異なる場合があります。

復帰値 なし。

注意事項 該当するイベントプロシージャが無い場合は、何も実行しません。

サンプル

```
void sample_proc(WSCbase* object){
  object->execProcedure(WSEV_ACTIVATE);
}
```

2.8.18 execProcedure 関数の説明

書式 void execProcedure(char* pname)

機能 指定された名称で、イベントプロシージャを実行します。

処理概要 指定された名称のイベントプロシージャが存在するならば、それを実行します。

引数

(in)char* pname	イベントプロシージャ名称

復帰値 なし。

注意事項 該当するイベントプロシージャが無い場合は、何も実行しません。

サンプル

```
void sample_proc(WSCbase* object){
  object->execProcedure("イベントプロシージャ名称");
}
```

2.8.19 setFocus 関数の説明

書式 long setFocus(WSCbool fl = True)

機能 フォーカスを設定します。

処理概要 True が指定されると、フォーカスを設定し、False が指定されるとフォーカスを失います。

引数

(in)WSCbool fl	True = あてる、False = 外す

復帰値 WS_NO_ERR = 正常、それ以外はエラー。

注意事項 フォーカスの状態変化により、onFocusChange() イベント関数が起動し、WSEV_FOCUS_CH が発生します。

サンプル
```
void sample_proc(WSCbase* object){
  //フォーカスをあてる場合1。
  object->setFocus();
  //フォーカスをあてる場合2。
  object->setFocus(True);
  //フォーカスをはずす場合。
  object->setFocus(False);
}
```

2.8.20 setSpecialFocus 関数の説明

書式 long setSpecialFocus(WSCbool fl = True)

機能 キーボード特別フォーカスであるスペシャルフォーカスを設定します。

処理概要 True が指定されると、スペシャルフォーカスを設定し、False が指定されるとスペシャルフォーカスを失います。

引数

(in)WSCbool fl	True = あてる、False = 外す

復帰値 WS_NO_ERR = 正常、それ以外はエラー。

注意事項 フォーカスの状態変化により、onSpecialFocusChange() イベント関数が起動し、WSEV_SPECIAL_FOCUS_CH が発生します。

サンプル
```
void sample_proc(WSCbase* object){
  //フォーカスをあてる場合1。
  object->setSpecialFocus();
  //フォーカスをあてる場合2。
  object->setSpecialFocus(True);
  //フォーカスをはずす場合。
  object->setSpecialFocus(False);
}
```

2.8.21 getFocus 関数の説明

書式 WSCbool getFocus()

機能 フォーカスの状態を取得します。

処理概要

引数 なし。

復帰値 True = フォーカスがあたっている、False = あたっていない。

注意事項

サンプル
```
void sample_proc(WSCbase* object){
  WSCbool fl = object->getFocus();
  if (fl == False){
    //フォーカスがあたっていない。
  }else{
    //フォーカスがあたっている。
  }
}
```

2.8.22 getSpecialFocus 関数の説明

書式 WSCbool getSpecialFocus()

機能 スペシャルフォーカスの状態を取得します。

処理概要

引数 なし。

復帰値 True = フォーカスがあたっている、False = あたっていない。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  WSCbool fl = object->getSpecialFocus();
  if (fl == False){
    //スペシャルフォーカスがあたっていない。
  }else{
    //スペシャルフォーカスがあたっている。
  }
}
```

2.8.23 getAllChildren 関数の説明

書式 long getAllChildren(WSClistData &list)

機能 全ての子どものリストを取得します。

処理概要 getChildren() は、自分の直下のみのインスタンスを対象とするのに対し、getAllChildren() は、その子のまた子まで、全てをたどります。

引数

(out)WSClistData& list	返り値を格納するリストを渡します

復帰値 WS_NO_ERR = 正常、それ以外はエラー。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  WSClistData children;
  object->getAllChildren(children);
  int i;
  long num = children.getNum();
  for(i=0; i< num; i++){
    WSCbase* child = (WSCbase*)children[i];
    // 個々の子インスタンスに処理を行います。
  }
}
```

2.8.24 getParentWindow 関数の説明

書式 WSCbase* getParentWindow()

機能 親のアプリケーションウィンドウを返します。

処理概要 親をたどっていき、一番上位のもの、すなわち、アプリケーションウィンドウを返値します。

引数 なし。

復帰値 アプリケーションウィンドウを返します。

注意事項 自分自身がアプリケーションウィンドウの場合は、自分自身が返値されます。

サンプル

```
void sample_proc(WSCbase* object){
  WSCbase* parent_window = object->getParentWindow();
}
```

2.8.25 getParent 関数の説明

書式 WSCbase* getParent()

機能 インスタンスを管理している親インスタンスを返します。

処理概要 同上。

引数 なし。

復帰値 親インスタンスを返します。

注意事項 親インスタンスがない、アプリケーションウィンドウなどの場合、NULL を返します。

サンプル

```
void sample_proc(WSCbase* object){
  WSCbase* parent = object->getParent();
}
```

2.8.26 onMouseIn 関数の説明

書式 virtual void onMouseIn(WSCpoint* pt)

機能 マウスポインタがインスタンス内に移動して来た場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_MOUSE_IN) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、マウスポインタに関するイベント処理を行うことができます。

引数

(out)WSCpoint* pt	マウスポインタのインスタンス内の座標

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onMouseIn(WSCpoint* pt){
  short x = pt->x; //X 座標
  short y = pt->y; //Y 座標
  //派生クラスでマウスポインタがインスタンス内に移動して来た場合に
  //行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onMouseIn(pt);
}
```

2.8.27 onMouseOut 関数の説明

書式 virtual void onMouseOut()

機能 マウスポインタがインスタンス外に出た場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_MOUSE_OUT) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、マウスポインタに関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onMouseOut(){
  //派生クラスでマウスポインタがインスタンス外に移動して来た場合に
  //行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onMouseOut();
}
```

2.8.28 onMouseMove 関数の説明

書式 virtual void onMouseMove(WSCpoint* pt)

機能 マウスポインタがインスタンス内で移動した場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_MOUSE_MOVE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、マウスポインタに関するイベント処理を行うことができます。

引数

(out)WSCpoint* pt	マウスポインタのインスタンス内の座標

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onMouseMove(WSCpoint* pt){
  short x = pt->x; //X 座標
  short y = pt->y; //Y 座標
  //派生クラスでマウスポインタが移動した場合に
  //行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onMouseMove(pt);
}
```

2.8.29 onMousePress 関数の説明

書式 virtual void onMousePress(WSCpoint* pt)

機能 マウスボタンがインスタンス内で押下された場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_MOUSE_PRESS) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、マウスポインタに関するイベント処理を行うことができます。

引数

(out)WSCpoint* pt	マウスポインタのインスタンス内の座標

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
#include <WSDmouse.h>
void new_class::onMousePress(WSCpoint* pt){
  short x = pt->x; //X 座標
  short y = pt->y; //Y 座標
  //派生クラスでマウスボタンが押下された場合に
  //行う処理を記述します。
  long status  = WSGIappMouse()->getMouseStatus();
  if (status \& WS_MOUSE_BTN1){
    //左ボタンが押されている。
  }
  if (status \& WS_MOUSE_BTN2){
    //中ボタンが押されている。
  }
  if (status \& WS_MOUSE_BTN3){
    //右ボタンが押されている。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onMousePress(pt);
}
```

2.8.30 onMouseRelease 関数の説明

書式 virtual void onMouseRelease(WSCpoint* pt)

機能 マウスボタンがインスタンス内で離された場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_MOUSE_RELEASE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、マウスポインタに関するイベント処理を行うことができます。

引数

(out)WSCpoint* pt	マウスポインタのインスタンス内の座標

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
#include <WSDmouse.h>
void new_class::onMouseRelease(WSCpoint* pt){
  short x = pt->x; //X 座標
  short y = pt->y; //Y 座標
  //派生クラスでマウスボタンが離された場合に
  //行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onMouseRelease(pt);
}
```

2.8.31 onExpose 関数の説明

書式 virtual void onExpose(WSCrect* rect)

機能 インスタンスが露出し、描画が必要な場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_EXPOSE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、露出に関するイベント処理を行うことができます。

引数

(out)WSCrect* rect	インスタンス内の露出した領域の座標、幅、縦幅

復帰値 なし。
注意事項 クラスを派生させる場合にオーバーライドして利用します。
サンプル
```
void new_class::onExpose(WSCrect* rect){
  //EXPOSE に必要な処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onExpose(rect);
}
```

2.8.32 onResize 関数の説明

書式 virtual void onResize(WSCrect* rect)
機能 インスタンスがサイズ変更された場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_RESIZE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、サイズ変更に関するイベント処理を行うことができます。
引数

(out)WSCrect* rect	インスタンスの座標、幅、縦幅

復帰値 なし。
注意事項 クラスを派生させる場合にオーバーライドして利用します。
サンプル
```
void new_class::onResize(){
  //RESIZE に必要な処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onResize();
}
```

2.8.33 onVisibleChange 関数の説明

書式 virtual void onVisibleChange(WSCbool vis)
機能 インスタンスの可視属性が変化した場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_VISIBLE_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、表示属性に関するイベント処理を行うことができます。
引数

(out)WSCbool vis	インスタンスの新しい可視状態

復帰値 なし。
注意事項 クラスを派生させる場合にオーバーライドして利用します。
サンプル
```
void new_class::onVisibleChange(WSCbool vis){
  //表示属性変化時に必要な処理を記述します。
  if (vis == False){
    //不可視状態である。
  }else{
    //可視状態である。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onVisibleChange(vis);
}
```

2.8.34 onParentVisibleChange 関数の説明

書式 virtual void onParentVisibleChange(WSCbool vis)
機能 親インスタンスの可視属性の変化により、可視状態が変化した場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_PARENT_VISIBLE_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、表示属性に関するイベント処理を行うことができます。
引数

(out)WSCbool vis	親インスタンスの新しい可視状態

復帰値 なし。
注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onParentVisibleChange(WSCbool vis){
  //親インスタンス表示属性変化時に必要な処理を記述します。
  if (vis == False){
    //不可視状態である。
  }else{
    //可視状態である。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onParentVisibleChange(vis);
}
```

2.8.35 onSensitiveChange 関数の説明

書式 virtual void onSensitiveChange(WSCbool det)

機能 インスタンスの選択属性が変化した場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_SENSITIVE_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、選択属性に関するイベント処理を行うことができます。

引数

(out)WSCbool det	インスタンスの新しい選択属性

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onSensitiveChange(WSCbool vis){
  //操作属性変化時に必要な処理を記述します。
  if (vis == False){
    //操作不可状態である。
  }else{
    //操作可能状態である。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onSensitiveChange(vis);
}
```

2.8.36 onParentSensitiveChange 関数の説明

書式 virtual void onParentSensitiveChange(WSCbool det)

機能 親インスタンスの選択属性の変化により、選択状態が変化した場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_PARENT_VISIBLE_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、選択属性に関するイベント処理を行うことができます。

引数

(out)WSCbool det	親インスタンスの新しい選択属性

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onParentSensitiveChange(WSCbool vis){
  //親インスタンス操作属性変化時に必要な処理を記述します。
  if (vis == False){
    //操作不可状態である。
  }else{
    //操作可能状態である。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onParentSensitiveChange(vis);
}
```

2.8.37 onChildAdded 関数の説明

書式 virtual void onChildAdded(WSCbase* child)

機能 子インスタンスが追加された場合に、呼び出されます。

処理概要 アプリケーションは、この関数をオーバーライドすることで、子インスタンス追加に関するイベント処理を行うことができます。

引数

(out)WSCbase* child	追加された子インスタンス

復帰値 なし。

注意事項 クラスを派生させる場合にオーバーライドして利用します。

サンプル

```
void new_class::onChildAdded(WSCbase* child){
  //子インスタンスが追加された場合に必要な処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onChildAdded(child);
}
```

2.8.38 setData 関数の説明

書式 virtual void setData(WSCvariant* data,long code = WS_EN_DEFAULT)

機能 データソース指定をサポートするオブジェクトクラスで、データを設定するのに利用します。

処理概要 データソースとして定義されているプロパティにバリアント型でデータを指定します。

引数

(in)WSCvariant* data	オブジェクトに設定したいデータ
(in)long code	文字列型を指定した場合のコードの指定

コードには次のような値が指定できます。省略すると WS_EN_DEFAULT を指定した場合と同じになります。

値	意味
WS_EN_DEFAULT	現在の設定を指定
WS_EN_LOCALE	現在の LANG 環境変数の設定を指定
WS_EN_NONE	設定を行わない
WS_EN_ISO8859_1	ISO8859(1) を指定
WS_EN_ISO8859_2	ISO8859(2) を指定
WS_EN_ISO8859_3	ISO8859(3) を指定
WS_EN_ISO8859_4	ISO8859(4) を指定
WS_EN_ISO8859_5	ISO8859(5) を指定
WS_EN_ISO8859_6	ISO8859(6) を指定
WS_EN_ISO8859_7	ISO8859(7) を指定
WS_EN_ISO8859_8	ISO8859(8) を指定
WS_EN_ISO8859_9	ISO8859(9) を指定
WS_EN_ISO8859_10	ISO8859(10) を指定
WS_EN_ISO8859_11	ISO8859(11) を指定
WS_EN_ISO8859_12	ISO8859(12) を指定
WS_EN_ISO8859_13	ISO8859(13) を指定
WS_EN_ISO8859_14	ISO8859(14) を指定
WS_EN_ISO8859_15	ISO8859(15) を指定
WS_EN_UTF8	UTF8 を指定
WS_EN_KOI8R	KOI8R を指定
WS_EN_EUCJP	EUCJP を指定
WS_EN_SJIS	SJIS を指定
WS_EN_EUCKR	EUCKR を指定
WS_EN_EUCCN	EUCCN を指定
WS_EN_BIG5	BIG5 を指定

復帰値 なし。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  WSCvariant value;
  value = ... // 設定したい値を格納します。

  //エンコーディングを省略した場合
  object->setData(value);
  //エンコーディングを指定した場合
  object->setData(value,WS_EN_SJIS);
}
```

2.8.39 setVariantData 関数の説明

書式 void setVariantData(char* vname,WSCvariant)

機能 オブジェクトにいろいろな値を名称を付けて保持させます。

処理概要 いろいろな名称を指定することで、個別に値を記憶させることができます。記憶させた値は、getVariantData 関数により指定した名称で取り出します。状態を記憶させたり、イベントプロシージャ間でデータの受渡しをするのに便利です。

引数

(in)char* vname	記憶させたい値につける名称
(in)WSCvariant	記憶させたい値

復帰値 なし。

注意事項 WSCvariant 型なので、いろいろな型を指定することが出来ますが、char* 以外のポインタは、領域は確保されません。

サンプル

```
void sample_proc(WSCbase* object){
  //long 型の値を data1 の名称で記憶させます。
  long  val =1;
  object->setVariantData("data1",val);
  //char* 型の値を data2 の名称で記憶させます。
  char* str = "test";
  object->setVariantData("data2",str);
}
void sample_proc2(WSCbase* object){
  //data1 の名称で記憶させた値を取得します。
  long  val = object->getVariantData("data1");

  //data2 の名称で記憶させた char* は、WSCstring で受け取ります。
  WSCstring str;
  str = object->getVariantData("data2");

}
```

2.8.40 getVariantData 関数の説明

書式 WSCvariant &setVariantData(char* vname)

機能 名称を指定して保持されてある値を取得します。

処理概要 setVariantData 関数により、いろいろな名称を指定することで、個別に値を記憶させることができます。記憶させた値は、この getVariantData 関数により指定した名称で取り出します。状態を記憶させたり、イベントプロシージャ間でデータの受渡しをするのに便利です。

引数

(in)char* vname	取得したい値の名称

復帰値 WSCvariant&

注意事項 取得される値は WSCvariant 型なので、いろいろな型に代入することが出来ますが、char* 以外は、領域は確保されません。また、char* ポインタで値を取得したい場合は、一旦 WSCstring クラス、または、WSCvariant 型に代入してから、(char*) にキャストしてください。

サンプル setVariantData() を参照してください。

2.8.41 onFocusChange 関数の説明

書式 void onFocusChange(WSCbool fl)

機能 インスタンスのフォーカスの状態が変化した場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_FOCUS_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、フォーカスに関するイベント処理を行うことができます。

引数

(in)WSCbool fl	新たなフォーカスの状態

復帰値 なし。

注意事項 オーバーライドして使用してください。

サンプル

```
void new_class::onFocusChange(WSCbool fl){
  //フォーカスの状態が変化した場合に必要な処理を記述します。
  if (fl == False){
```

```
    //フォーカスを失った。
  }else{
    //フォーカスを獲得した。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onFocusChange(fl);
}
```

2.8.42 onSpecialFocusChange 関数の説明

書式 void onSpecialFocusChange(WSCbool fl)

機能 オブジェクトのスペシャルフォーカスの状態が変化した場合に実行されます。

処理概要 アプリケーションは、この関数をオーバーライドすることで、スペシャルフォーカスに関するイベント処理を行うことができます。

引数

(in)WSCbool fl	新たなフォーカスの状態

復帰値 なし。

注意事項 オーバーライドして使用してください。スペシャルフォーカスは、WSCvbtn、WSCvifield などのクラスで使用される、通常のフォーカスとは異なるリターンキー特別フォーカスです。

サンプル

```
void new_class::onSpecialFocusChange(WSCbool fl){
  //スペシャルフォーカスの状態が変化した場合に必要な処理を記述します。
  if (fl == False){
    //フォーカスを失った。
  }else{
    //フォーカスを獲得した。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onSpecialFocusChange(fl);
}
```

2.8.43 onSelectionChange 関数の説明

書式 void onSelectionChange(WSCbool fl)

機能 セレクション (選択されているデータ) の状態が変化した場合に実行されます。

処理概要 アプリケーションは、この関数をオーバーライドすることで、セレクションに関するイベント処理を行うことができます。セレクションとは、マウスで選択された文字列などのデータでカットアンドペーストされるデータのことを指します。False が渡される場合は、セレクションを失い、True が渡される場合は、セレクションを獲得したことを意味します。

引数

(in)WSCbool fl	セレクションの獲得状態

復帰値 なし。

注意事項 オーバーライドして使用してください。

サンプル

```
void new_class::onSelectionChagen(WSCbool fl){
  //セレクションの状態が変化した場合に必要な処理を記述します。
  if (fl == False){
    //セレクションを失った。
  }else{
    //セレクションを獲得した。
  }
  //処理を派生元クラスに引き継ぎます。
  old_class::onSelectionChange(fl);
}
```

2.8.44 setInternalObject 関数の説明

書式 void setInternalObject(WSCbool fl)

機能 内部インスタンス属性フラグを設定します。

処理概要 内部インスタンス属性を True に設定すると、win ファイルへの出力や、アプリケーションビルダでの編集が行えなくなります。

引数

(in)WSCbool fl	内部インスタンス指定

復帰値 なし。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
   WSCbase* inst = new  WSCdialog(object,"newwin000");
   inst->initialize();
   inst->setInternalObject(True);
   inst->setPropertyV(WSNname,"newwin000");
   inst->setPropertyV(WSNtitleString,"title1");
   inst->setPropertyV(WSNvis,(WSCbool)1);
   return inst;
}
```

2.8.45 getInternalObject 関数の説明

書式 WSCbool getInternalObject()

機能 内部インスタンス属性を取得します。

処理概要 内部インスタンス属性フラグの状態を返します。

引数 なし。

復帰値 内部インスタンス属性

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  WSClistData children = object->getChildren();
  long i;
  long num = children.getNum();
  for(i=0; i< num; i++){
    WSCbase* child = (WSCbase*)children［i］;
    WSCbool fl = inst->getInternalObject();
    if (fl != False){
      //内部インスタンスである。
    }else{
      //通常のインスタンスである。
    }
  }
}
```

2.8.46 setScaleOffsetPtr 関数の説明

書式 void setScaleOffsetPtr(double* ptr)

機能 表示倍率を格納した変数へのポインタを指定します。

処理概要 表示倍率を格納した変数へのポインタが設定されると、その倍率で描画します。複数のインスタンスに同じポインタを設定することで、一度に表示倍率を変更することができます。

引数

(in)double* ptr	倍率を格納した変数へのポインタ

復帰値 なし。

注意事項 ポインタの設定を解除したい場合は、NULL を指定します。

サンプル

```
double scale;
void sample_proc(WSCbase* object){
  //初期化プロシージャ等で、オフセットを設定します。
  object->setScaleOffsetPtr(\&scale);
}
```

2.8.47 setXOffsetPtr 関数の説明

書式 void setXOffset(short* ptr)

機能 表示位置Xのオフセットを格納した変数へのポインタを指定します。

処理概要 表示位置Xのオフセットを格納した変数へのポインタが設定されると、オフセット値を加算した座標で描画します。複数のインスタンスに同じポインタを設定することで、一度に表示位置を変更することができます。

引数

(in)short* ptr	オフセット値を格納した変数へのポインタ

復帰値 なし。

注意事項 ポインタの設定を解除したい場合は、NULL を指定します。

サンプル

```
short offsetx;
void sample_proc(WSCbase* object){
  //初期化プロシージャ等で、オフセットを設定します。
  object->setXOffsetPtr(\&offsetx);
}
```

2.8.48 setYOffsetPtr 関数の説明

書式 void setYOffset(short* ptr)

機能 表示位置Yのオフセットを格納した変数へのポインタを指定します。

処理概要 表示位置Yのオフセットを格納した変数へのポインタが設定されると、オフセット値を加算した座標で描画します。複数のインスタンスに同じポインタを設定することで、一度に表示位置を変更することができます。

引数

(in)short* ptr	オフセット値を格納した変数へのポインタ

復帰値 なし。

注意事項 ポインタの設定を解除したい場合は、NULL を指定します。

サンプル

```
short offsety;
void sample_proc(WSCbase* object){
  //初期化プロシージャ等で、オフセットを設定します。
  object->setXOffsetPtr(\&offsety);
}
```

2.8.49 getScaleOffsetPtr 関数の説明

書式 double* getScaleOffset()

機能 表示倍率を格納した変数へのポインタを取得します。

処理概要 表示倍率を格納した変数へのポインタを取得します。設定されていない場合、NULL が返されます。

引数 なし。

復帰値 表示倍率を格納した変数へのポインタ

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //初期化プロシージャ等で、オフセットを取得します。
  double* ptr = object->getScaleOffsetPtr();
}
```

2.8.50 getXOffsetPtr 関数の説明

書式 short* getXOffset()

機能 表示位置Xのオフセットを格納した変数へのポインタを取得します。

処理概要 表示位置Xのオフセットを格納した変数へのポインタを取得します。設定されていない場合、NULL が返されます。

引数 なし。

復帰値 表示位置Xのオフセットを格納した変数へのポインタ

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //初期化プロシージャ等で、オフセットを取得します。
  short* ptr = object->getXOffsetPtr();
}
```

2.8.51 getYOffsetPtr 関数の説明

書式 short* getYOffset()

機能 表示位置Yのオフセットを格納した変数へのポインタを取得します。

処理概要 表示位置Yのオフセットを格納した変数へのポインタを取得します。設定されていない場合、NULL が返されます。

引数 なし。

復帰値 表示位置Yのオフセットを格納した変数へのポインタ

注意事項 なし。

サンプル
```
void sample_proc(WSCbase* object){
  //初期化プロシージャ等で、オフセットを取得します。
  short* ptr = object->getYOffsetPtr();
}
```

2.8.52 isDefaultValue 関数の説明

書式 long isDefaultValue(char* pname,WSCbool* fl)

機能 プロパティがデフォルト値であるか否かを調べます。

処理概要 指定されたプロパティがデフォルト値であるか否かをパラメータ fl に格納して返します。

引数

(in)char* pname	プロパティ名
(out)WSCbool* fl	結果を取得する変数へのポインタ

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 なし。

サンプル
```
void sample_proc(WSCbase* object){
  //WSNlabelString プロパティがデフォルト値かどうかしらべます。
  WSCbool fl;
  long ret = object->isDefaultValue(WSNlabelString,&fl);
}
```

2.8.53 setDefaultValue 関数の説明

書式 long setDefaultValue(char* pname)

機能 指定されたプロパティにデフォルト値を設定します。

処理概要 指定されたプロパティにデフォルト値を設定します。

引数

(in)char* pname	プロパティ名

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 なし。

サンプル
```
void sample_proc(WSCbase* object){
  //WSNlabelString プロパティをデフォルト値にします。
  object->setDefaultValue(WSNlabelString);
}
```

2.8.54 existProperty 関数の説明

書式 WSCbool existProperty(char* pname)

機能 指定されたプロパティの存在を調べます。

処理概要 指定されたプロパティが存在するか否かを返します。

引数

(in)char* pname	プロパティ名

復帰値 True= 存在、False=存在せず。

注意事項 なし。

サンプル
```
void sample_proc(WSCbase* object){
  //WSNlabelString プロパティが存在するかどうか調べます。
  WSCbool exist = object->existProperty(WSNlabelString);
  if (exist != False){
```

```
    //存在する。
  }else{
    //存在しない。
  }
}
```

2.8.55 getChildInstance 関数の説明

書式 WSCbase* getChildInstance(char* iname)

機能 指定された名称の子インスタンスを取得します。

処理概要

引数

(in)char* iname	子インスタンス名

復帰値 子インスタンス

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //newvlab_001 の名称を持つ子インスタンスを取得
  WSCbase* child = object->getChildInstance("newvlab_001");
  if (child != NULL){
    //取得された。
  }
}
```

2.8.56 getFocusMoveInstance 関数の説明

書式 WSCbase* getFocusMoveInstance(long direction)

機能 指定された方向へのフォーカス移動対象インスタンスを取得します。

処理概要

引数

(in)long direction	方向

方向には次のような値が指定できます。

値	意味
WS_UP	上方向
WS_DOWN	下方向
WS_RIGHT	右方向
WS_LEFT	左方向
WS_RET	リターンキーによる移動方向

復帰値 フォーカス移動先インスタンス

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //フォーカス移動可能な右側に存在するインスタンスを取得
  WSCbase* inst = object->getFocusMoveInstance(WS_RIGHT);
  if (inst != NULL){
    //取得された。
  }
}
```

2.8.57 needUpdate 関数の説明

書式 void needUpdate()

機能 更新を行うことを指定します。

処理概要 更新フラグたてて、イベントプロシージャ終了時に表示更新されるようにします。実際の表示更新は、update() 関数で行われます。WSGIappObjectList()-¿execUpdate() によっても表示更新されます。

引数 なし。

復帰値 なし。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //更新フラグをたて、プロシージャ終了後に update() が実行されるようにします。
  object->needUpdate();
}
```

2.8.58 isNeedUpdate 関数の説明

書式 WSCbool isNeedUpdate()

機能 更新フラグの状態を取得します。

処理概要 プロパティが変更されたり、needUpdate() が実行されると、更新フラグが True になります。これから表示更新されることを示します。

引数 なし。

復帰値 True = これから更新される、False = 更新されない。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //更新フラグがたっているか否かを調べます。
  WSCbool* fl = object->isNeedUpdate();
  if (fl == False){
    //更新フラグはたっていない。
  }else{
    //更新フラグはたっていない。
  }
}
```

2.8.59 isParent 関数の説明

書式 WSCbool isParent(WSCbase* instance)

機能 指定されたインスタンスが親インスタンスであるか調べます。

処理概要

引数

(in)WSCbase* instance	インスタンス

復帰値 True = 親インスタンスである、False = 親インスタンスでない。

注意事項 なし。

サンプル

```
extern WSCmainWindow* newwin000;
void sample_proc(WSCbase* object){
  //ウィンドウ newwin000 が親インスタンスかどうか調べます。
  WSCbool fl = object->isParent(newwin000);
  if (fl == False){
    //親インスタンスでない。
  }else{
    //親インスタンスである。
  }
}
```

2.8.60 existTrigger 関数の説明

書式 WSCbool existTrigger(long trigger)

機能 指定されたトリガーが使用可能か否かを取得します。

処理概要

引数

(in)long trigger	トリガー

復帰値 True = 使用可能である、False = 使用可能ではない。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //トリガが使用可能かどうか調べます。
  WSCbool fl = object->existTrigger(WSEV_ACTIVATE);
  if (fl != False){
```

```
    //使用可能。
  }
}
```

2.8.61 getMouseAddr 関数の説明

書式 WSCbool getMouseAddr(short* x,short* y)

機能 インスタンスの座標系でのマウスポインタの座標を取得します。

処理概要

引数

(out)short* x	X座標を格納する変数へのポインタ
(out)short* y	Y座標を格納する変数へのポインタ

復帰値 True = 取得可能、False = 取得可能ではない。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  short px,py;
  object->getMouseAddr(\&x,\&y);
}
```

2.8.62 addProcedure 関数の説明

書式 long addProcedure(WSCprocedure* ep)

機能 イベントプロシージャを追加します。

処理概要

引数

(in)WSCprocedure* ep	イベントプロシージャインスタンス

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 なし。

サンプル

```
//新たに張り付けるイベントプロシージャ用関数
void procedure1(WSCbase*){
  //処理
}
//イベントプロシージャ本体
void sample_proc(WSCbase* object){
  //procedure1 関数をプロシージャ名 procedure1、
  //トリガ WSEV_ACTIVATE で張り付けます。
  //WSEV_ACTIVATE が発生すると、procedure1 が起動されるようになります。
  WSCprocedure* ep = new WSCprocedure("procedure1",WSEV_ACTIVATE);
  ep->setFunction(procedure1,"procedure1");
  object->addProcedure(ep);
}
```

2.8.63 delProcedure 関数の説明

書式 long delProcedure(WSCprocedure* ep)

機能 イベントプロシージャの登録を解除します。

処理概要 addProcedure() 関数で登録されたイベントプロシージャの登録を解除します。

引数

(in)WSCprocedure* ep	イベントプロシージャインスタンス

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  //張り付けられているイベントプロシージャを削除します。
  WSClistData procedures = object->getProcedures();
  long i;
  long num = procedures.getNum();
  for(i=0; i<num; i++){
    WSCprocedure* ep = (WSCprocedure*)procedures [i] ;
```

```
    object->delProcedures(ep);
  }
}
```

2.8.64 getdev 関数の説明

書式 WSDdev* getdev()

機能 インスタンスで使用されているデバイスクラスのインスタンスを取得します。

処理概要 デバイスクラスのインスタンスは、オブジェクトの描画やイベントのハンドリングに使用されます。そのデバイスクラスのインスタンスを返します。

引数 なし。

復帰値 デバイスクラスのインスタンス

注意事項 インスタンス生成後、一度も表示されていない場合、デバイスクラスのインスタンスが作成されていない場合があります。作成されていない場合、NULL が返されます。

サンプル WSDdev クラスのサンプルを参照してください。

2.8.65 getPropertyInheritChild 関数の説明

書式 WSCbase* getPropertyInheritChild()

機能 この関数は、子インスタンスのプロパティを親インスタンスにマージする場合にオーバーライドして使用します。

処理概要 クラス派生において、この関数をオーバーライドし、プロパティをマージしたいメンバであるメンバインスタンスを返すようにすることで、クラス本体のインスタンスにそのメンバインスタンスのプロパティをクラス本体に追加することができます。

引数 なし。

復帰値 メンバインスタンス

注意事項

サンプル

```
WSCbase* new_class::getPropertyInheritChild(){
  //メンバインスタンスのプロパティをクラス本体のプロパティとして
  //公開したい場合、この関数をオーバーライドし、
  //そのインスタンスを返すようにします。
  return member1;
}
```

3 WSCbaseDialog

3.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow

3.2 概要

ポップアップダイアログの基本的な機能を提供します。

3.3 機能

- デフォルトで OK、NO、キャンセルボタンを提供します。
- ポップアップ制御関数を提供します。

3.4 注意事項

3.5 プロパティー覧

クラス WSCbaseDialog のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
WSCbool	WSNok	OK ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNno	NO ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	
char*	WSNokString	OK ボタン表示文字列	OK
char*	WSNnoString	NO ボタン表示文字列	NO
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	100
unsigned char	WSNshadowThickness	影幅	0
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	

	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

3.6 トリガー一覧

WSCbaseDialog クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

3.7 関数一覧

- void onActivate() :WSCbaseDialog
- long getStatus() :WSCbaseDialog
- long popup() :WSCbaseDialog
- WSCbase* getFormObject() :WSCbaseDialog
- WSCbase* getCmdFormObject() :WSCbaseDialog

3.8 関数仕様

3.8.1 onActivate 関数の説明

書式 void onActivate()

機能 ダイアログが提供するボタンが押下された場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、ボタン押下に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項 なし。

サンプル

```
void new_class::onActivate(){
  //ダイアログが提供するボタンが押下された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

3.8.2 getStatus() 関数の説明

書式 long getStatus()

機能 ダイアログが提供するボタンのどれが押下された状態であるかを取得します。

処理概要

引数 なし。

復帰値 WS_DIALOG_OK = OK ボタン、WS_DIALOG_NO = NO ボタン、WS_DIALOG_CANCEL = キャンセルボタン。

注意事項 なし。

サンプル

```
void sample_proc(WSCbase* object){
  WSCbaseDialog* dialog = (WSCbaseDialog*)object->cast("WSCbaseDialog");
  if (dialog == NULL){
    return;
  }
  long status = dialog->getStatus();
  if (status == WS_DIALOG_OK){
    //OK ボタンが押された
  }else
  if (status == WS_DIALOG_NO){
    //NO ボタンが押された
  }else
  if (status == WS_DIALOG_CANCEL){
    //CANCEL ボタンが押された
  }
}
```

3.8.3 popup() 関数の説明

書式 long popup()

機能 ダイアログを表示状態にし、ボタン押下があるまで待機します。ボタンが押下されると、どのボタンが押下されたかを返値します。

処理概要

引数 なし。

復帰値 WS_DIALOG_OK = OK ボタン、WS_DIALOG_NO = NO ボタン、WS_DIALOG_CANCEL = キャンセルボタン。

注意事項 なし。

サンプル

```
extern WSCbaseDialog* newdial_000;
void sample_proc(WSCbase* object){
  //ダイアログを表示。
  //ダイアログ終了まで、復帰待ち。
  long result = newdial_000->popup();

  if (result == WS_DIALOG_OK){
    //OK ボタンが押された
  }else
  if (result == WS_DIALOG_NO){
    //NO ボタンが押された
  }else
  if (result == WS_DIALOG_CANCEL){
    //CANCEL ボタンが押された
  }
}
```

3.8.4 getFormObject() 関数の説明

書式 WSCbase* getFormObject()

機能 ダイアログ上のフォームインスタンスを取得します。

処理概要
引数 なし。
復帰値 ダイアログ上のフォームインスタンス
注意事項 なし。
サンプル
```
void sample_proc(WSCbase* object){
  WSCbaseDialog* dialog = (WSCbaseDialog*)object->cast("WSCbaseDialog");
  if (dialog == NULL){
    return;
  }
  WSCbase* form = dialog->getFormObject();
}
```

3.8.5 getCmdFormObject() 関数の説明

書式 WSCbase* getCmdFormObject()
機能 ダイアログ上のコマンド領域のフォームインスタンスを取得します。
処理概要
引数 なし。
復帰値 ダイアログ上のコマンド領域のフォームインスタンス
注意事項 なし。
サンプル
```
void sample_proc(WSCbase* object){
  WSCbaseDialog* dialog = (WSCbaseDialog*)object->cast("WSCbaseDialog");
  if (dialog == NULL){
    return;
  }
  WSCbase* form = dialog->getCmdFormObject();
}
```

4 WSCbaseList

4.1 継承元

次のオブジェクトを継承しています。

4.2 概要

GUI クラスのインスタンスの管理をします。インスタンスの検索等を行うのに利用します。

4.3 機能

- インスタンスの検索

4.4 注意事項

4.5 関数一覧

- WSDbaseList* WSGIappObjectList(); :WSDbaseList
- WSCbase* getInstance(char* class_name,char* iname) :WSDbaseList
- WSCRbase* getRemoteInstance(char* iname) :WSDbaseList
- WSCbool existInstance(WSCbase*); :WSDbaseList
- void execUpdate(); :WSDbaseList
- void addEvent(WSCbase*,long fl = WSEV_NONE); :WSDbaseList
- void execEvent(long); :WSDbaseList
- void execEvent(char*); :WSDbaseList
- void delEvent(WSCbase*,long fl = WSEV_NONE); :WSDbaseList

4.6 関数仕様

4.6.1 WSGIappObjectList 関数の説明

書式 WSDbaseList* WSGIappObjectList()

機能 アプリケーションにひとつ存在するインスタンス管理クラスのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 インスタンス管理クラスインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル

```
//更新待ちのインスタンス達を更新します。
WSGIappObjectList()->execUpdate();
```

4.6.2 getInstance 関数の説明

書式 WSCbase* getInstance(char* cname,char* iname)

機能 GUI クラスのインスタンスの検索を行います。

処理概要 クラス名に WSCbase を指定すると、全検索を行います。その他は、その指定されたクラスのみを検索します。

引数

(in)char* cname	クラス名称
(in)char* iname	インスタンス名称

復帰値 GUI インスタンス

注意事項 指定するクラス名称で WSCbase 以外を指定した場合、そのクラス名称で生成された物しか検索をしません。したがって、例えば、WSCvlabel を継承する WSCvbtn を、WSCvlabel で検索したとしてもヒットしません。

サンプル

```
//WSCvlabel クラスの名称 newvlab_000 なるインスタンスを取得します。
WSCbase* inst = WSGIappObjectList()->getInstance("WSCvlabel","newvlab_000");
```

4.6.3 getRemoteInstance 関数の説明

書式 WSCRbase* getRemoteInstance(char* iname)

機能 GUI クラスのリモートインスタンスの取得を行います。

処理概要 リモートサーバに存在するリモートインスタンスのうち、公開されているリモートインスタンスへの参照 (仮想リモートインスタンス) を取得します。

引数

(in)char* iname	インスタンス名称

復帰値 仮想リモートインスタンス

注意事項 エージェント、リモートインスタンスサーバーが起動されている必要があります。

サンプル

```
// newvlab_000 なるリモートインスタンスへの参照を取得します。
WSCRbase* remote_inst = WSGIappObjectList()->getRemoteInstance("newvlab_000");
// 取得されたリモートインスタンスへアクセスします。
if (remote_inst != NULL){
  remote_inst->setProperty(WSNlabelString,"Hello!");
}
```

4.6.4 existInstance 関数の説明

書式 WSCbool existInstance(WSCbase* instance)

機能 インスタンスが妥当なものかどうか、開放されてしまったかどうかを調べます。

処理概要

引数

(in)WSbase* instance	GUI クラスのインスタンス

復帰値 True = 正しいインスタンス、False = 妥当でないインスタンス

注意事項

サンプル

```
void sample(){
  WSCbase* inst = ...//正しいインスタンスかどうか調べたいインスタンス。
  WSCbool exit = WSGIappObjectList()->existInstace(inst);
  if (exit == False){
    //無効インスタンス、または無効ポインタである。
  }else{
    //正しいインスタンスである。
  }
}
```

4.6.5 execUpdate 関数の説明

書式 void execUpdate()

機能 未更新のインスタンスを更新します。

処理概要 未更新のインスタンスの udpate() 関数を呼び出し、表示更新を行います。

引数 なし。

復帰値 なし。

注意事項

サンプル WSGIappObjectList() を参照してください。

4.6.6 addEvent 関数の説明

書式 void addEvent(WSCbase*,long)

機能 イベントを通知したいインスタンスと、トリガーを指定します。

処理概要 登録されたインスタンスに対して、execEvent() 関数によって、イベントを一斉に通知し、イベントプロシージャを実行することができます。省略すると WSEV_NONE が指定されます。省略した場合は、トリガ指定による execEvent() は行えません。

引数

(in)WSbase* instance	登録したいインスタンス
(in)long trigger	登録したいトリガー

復帰値 なし。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  // WSCbaseList::execEvent(char*) でイベント通知を行います。
  WSGIappObjectList()->addEvent(object);
}
void sample_proc2(WSCbase* object){
  // WSCbaseList::addEvent() で登録されたインスタンスの
  // 指定されたプロシージャ名のイベントプロシージャを実行します。
  WSGIappObjectList()->execEvent("procedure1");
}
```

4.6.7 execEvent 関数の説明

書式 void execEvent(long)

機能 指定されたイベント通知登録したインスタンスに対して、イベントを通知します。

処理概要 指定されたトリガで登録されたインスタンスに対して、イベントを一斉に通知し、イベントプロシージャを実行することができます。

引数

(in)long trigger	通知したいトリガー

復帰値 なし。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  // WSCbaseList::execEvent(char*) でイベント通知を行います。
  WSGIappObjectList()->addEvent(object,WSEV_ACTIVATE);
```

```
}
void sample_proc2(WSCbase* object){
  // WSCbaseList::addEvent() で登録されたインスタンスの
  // 指定されたトリガのイベントプロシージャを実行します。
  WSGIappObjectList()->execEvent(WSEV_ACTIVATE);
}
```

4.6.8 execEvent 関数の説明

書式 void execEvent(char* evname)

機能 イベント通知登録したインスタンスに対して、イベントプロシージャ名称によってイベントを通知します。

処理概要 登録されたインスタンスに対して、この execEvent() 関数によって、イベントを一斉に通知し、イベントプロシージャを実行することができます。

引数

(in)char* evname	実行させたいイベントプロシージャ名称

復帰値 なし。

注意事項

サンプル addEvent() を参照してください。

4.6.9 delEvent 関数の説明

書式 void delEvent(WSCbase*,long)

機能 登録したインスタンス登録解除します。

処理概要 addEvent() 関数で登録されたインスタンスの登録を解除します。

引数

(in)WSbase* instance	登録したインスタンス
(in)long trigger	登録したトリガー

復帰値 なし。

注意事項

サンプル

```
void sample_proc(WSCbase* object){
  // WSCbaseList::addEvent() で登録されたインスタンスの登録を解除します。
  WSGIappObjectList()->delEvent(object,WSEV_ACTIVATE);
}
```

5 WSCcheckGroup

5.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform WSCradioGroup

5.2 概要

チェックボタングループを持つフォームです。チェックボタングループは、WSCvradio クラスのグループで、複数選択属性になっています。択一選択できる WSCradioGroup クラスもあります。

5.3 機能

- チェックボタングループの提供

5.4 注意事項

5.5 プロパティー覧

クラス WSCcheckGroup のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned long	WSNvalue	値	0
char*	WSNtitleString	タイトル文字列	
char*	WSNmenuItems	メニュー項目	item1,item2,item3
unsigned short	WSNmenuItemHeight	メニュー項目縦幅	20
unsigned char	WSNfont	フォント番号	0
unsigned char	WSNmargin	マージン	4
WSCbool	WSNindicatorOn	インジケータ使用	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNindicatorType	インジケータ TYPE	0
	IN-OUT	WS_IN_OUT (0)	
	OUT	WS_OUT (1)	
	IN	WS_IN (2)	
	NONE	WS_NONE (3)	
unsigned char	WSNindicatorSize	インジケータ SIZE	16
unsigned char	WSNindicatorShadow	インジケータ影幅	2
short	WSNindicatorColor	インジケータ色	DEF11
short	WSNindicatorPixmap	インジケータ画	
short	WSNselectColor	選択色	DEF7
short	WSNselectPixmap	インジケータ選択画	
short	WSNlabelPixmap	表示画	
unsigned char	WSNorientation	表示方向	1
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	

	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

5.6 トリガー一覧

WSCcheckGroup クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

5.7 関数一覧

5.8 関数仕様

6 WSCcolorSet

6.1 継承元

次のオブジェクトを継承しています。

6.2 概要

WSDcolor 色クラスのインスタンスを管理します。色に関する情報を取得したい場合にこのクラスを利用します。

6.3 機能

- WSDcolor 色クラスのインスタンスの管理

6.4 注意事項

6.5 関数一覧

- WSCcolorSet* WSGIappColorSet(); :WSCcolorSet
- WSDcolor* getColor(short no); :WSCcolorSet
- WSDcolor* getColor(char* name); :WSCcolorSet
- short getColorNo(char*); :WSCcolorSet
- short getDefaultColorNo(long kind); :WSCcolorSet
- WSDcolor* getDefaultColor(long kind); :WSCcolorSet
- char* getColorName(short); :WSCcolorSet
- WSCbool existColor(short cid); :WSCcolorSet

6.6 関数仕様

6.6.1 WSGIappColorSet 関数の説明

書式 WSCcolorSet* WSGIappColorSet()

機能 アプリケーションにひとつ存在する色管理クラスのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 色管理クラスインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル

```
//色名称を指定して、色番号を取得します。
short cno = WSGIappColorSet()->getColorNo("#ffffff");
```

6.6.2 getColor 関数の説明

書式 WSDcolor* getColor(short cno)

機能 色番号から色インスタンスを取得します。

処理概要

引数

(in)short cno	色番号

復帰値 色インスタンス

注意事項 返された色インスタンスを開放してはなりません。

サンプル

```
//色番号を指定して、色インスタンスを取得します。
WSDcolor* color = WSGIappColorSet()->getColor(cno);
```

6.6.3 getColor 関数の説明

書式 WSDcolor* getColor(char* cname)

機能 色名称から色インスタンスを取得します。

処理概要

引数

(in)char* cname	色名称

色名称は、次のフォーマットで指定します。
#RRGGBB
RR: ００～ｆｆの範囲で１６進数で赤の輝度を指定します。
GG: ００～ｆｆの範囲で１６進数で緑の輝度を指定します。
BB: ００～ｆｆの範囲で１６進数で青の輝度を指定します。

色	色名称
白	#ffffff
黒	#000000
赤	#ff0000
緑	#00ff00
青	#0000ff
灰色	#888888
暗赤	#880000
暗緑	#008800
暗青	#000088

復帰値 色インスタンス

注意事項 返された色インスタンスを開放してはなりません。

サンプル

```
//色名称を指定して、色インスタンスを取得します。
WSDcolor* color = WSGIappColorSet()->getColor("#ff0000");
```

6.6.4 getColorNo 関数の説明

書式 short getColorNo(char* cname)

機能 色名称から色番号を取得します。

処理概要

引数

(in)char*	色名称

色名称に関しては、getColor(char*) 関数を参照ください。

復帰値 色番号

注意事項

サンプル WSGIappColorSet() を参照してください。

6.6.5 getDefaultColorNo 関数の説明

書式 short getDefaultColorNo(long kind)

機能 指定されたデフォルト色の色番号を取得します。

処理概要

引数

(in)long	デフォルト色種別子

指定可能なデフォルト色識別子には、次のようなものがあります。

デフォルト色種別子	意味
WS_DF_FORECOLOR	デフォルト表示色
WS_DF_BACKCOLOR	デフォルト背景色
WS_DF_TOPSHADOWCOLOR	デフォルト上影色
WS_DF_BOTTOMSHADOWCOLOR	デフォルト下影色
WS_DF_MENUFORECOLOR	デフォルトメニュー表示色
WS_DF_MENUBACKCOLOR	デフォルトメニュー背景色
WS_DF_MENUSELECTFORECOLOR	デフォルトメニュー選択表示色
WS_DF_MENUSELECTCOLOR	デフォルトメニュー選択背景色
WS_DF_MENUTOPSHADOWCOLOR	デフォルトメニュー上影色
WS_DF_MENUBOTTOMSHADOWCOLOR	デフォルトメニュー下影色
WS_DF_DARKBACKCOLOR	デフォルト暗背景色
WS_DF_WORKBACKCOLOR	デフォルト作業領域背景色
WS_DF_BARSHADOWCOLOR	デフォルトスクロールバー影色
WS_DF_NWFORECOLOR	デフォルトコマンド表示色
WS_DF_NWBACKCOLOR	デフォルトコマンド背景色
WS_DF_NWTOPSHADOWCOLOR	デフォルトコマンド上影色
WS_DF_NWBOTTOMSHADOWCOLOR	デフォルトコマンド下影色
WS_DF_DARKBOTTOMSHADOWCOLOR	デフォルト暗下影色

復帰値 色番号

注意事項

サンプル

```
//デフォルト色識別子を指定して、色番号を取得します。
short cno = WSGIappColorSet()->getDefaultColorNo(WS_DF_FORECOLOR);
```

6.6.6 getDefaultColor 関数の説明

書式 WSDcolor* getDefaultColor(long kind)
機能 指定されたデフォルト色の色インスタンスを取得します。
処理概要
引数

(in)long	デフォルト色種別

デフォルト色種別に関しては、getDefaultColorNo(long) 関数を参照ください。
復帰値 色インスタンス
注意事項 返された色インスタンスを解放してはなりません。
サンプル

```
//デフォルト色識別子を指定して、色インスタンスを取得します。
WSCcolor* color = WSGIappColorSet()->getDefaultColor(WS_DF_FORECOLOR);
```

6.6.7 getColorName 関数の説明

書式 char* getColorName(short cno)
機能 指定された色番号の色名称を取得します。
処理概要
引数

(in)short cno	色番号

復帰値 色名称
注意事項 返された色名称を解放してはなりません。
サンプル

```
//色番号を指定して、色名称を取得します。
char* cname = WSGIappColorSet()->getColorName(cno);
```

6.6.8 existColor 関数の説明

書式 WSCbool existColor(short cno)
機能 指定された色番号が存在するか否かを取得します。
処理概要
引数

(in)short cno	色番号

復帰値 True = 存在、False = 存在しない。
注意事項
サンプル

```
//指定された色番号 cno が存在するか否かを取得します。
WSCbool exist = WSGIappColorSet()->existColor(cno);
if (exist == False){
  //存在せず。
}else{
  //存在。
}
```

7 WSCcomboBox

7.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

7.2 概要

テキスト入力と選択メニューを持った、入力を提供します。選択メニューの設定に関しては、注意事項を御参照ください。

7.3 機能

- キーボードからのテキスト入力機能
- マウスによる、選択一覧メニューからの入力機能

7.4 注意事項

コンボボックスオブジェクトの選択メニューに一覧表示したいものは、プロパティ WSNmenuItems で指定します。

例えば、挿絵の様な選択表示を行わせる場合は、

```
select1,select2,select3
```

の様に設定します。

7.5 プロパティ一覧

クラス WSCcomboBox のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNmenuItems	メニュー項目	item1,item2,item3
unsigned short	WSNmaxHeight	最大縦幅	0
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
short	WSNworkBackColor	作業領域背景色	DEF11
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0
	不可	False (0)	
	可	True (1)	
unsigned char	WSNusePixmap	ちらつき防止	0
	無	(0)	
	有	(1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned short	WSNcolumns	カラム数	20
short	WSNcursorPos	カーソル位置	0
WSCbool	WSNkanjiIn	漢字入力	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNinterCur	挿入モード	1
	不可	False (0)	
	可	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
char*	WSNreturn	リターンキー移動先	

WSCbool	WSNenableFocusMove	フォーカス移動	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNfillSpace	フィルスペース	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNcursorAdjust	カーソル補正	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNifieldSkipMode	フィールド間連係	0
	不可	False (0)	
	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	1
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0

不可	False (0)
可	True (1)

7.6 トリガー一覧

WSCcomboBox クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

7.7 関数一覧

- void onActivate() :WSCcomboBox

7.8 関数仕様

7.8.1 onActivate 関数の説明

書式 void onActivate()

機能 値が選択された場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、値選択に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項 なし。

サンプル

```
void new_class::onActivate(){
  //値が選択された場合に
  //行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

8 WSCconductor

8.1 継承元

次のオブジェクトを継承しています。

8.2 概要

ストア機能を実現します。指定されたインスタンスをシリアライズしてファイル出力したり、ファイルからインスタンスを読み込んだりすることができます。簡単にストア機能を利用できる WSGFloadWindow()、WSGFsaveWindow() グローバル関数も利用できます。

8.3 機能

- アプリケーションウィンドウのファイル等へのシリアライズ
- シリアライズされたインスタンスの読み込み

8.4 注意事項

8.5 関数一覧

- WSCconductor* WSGIconductor(); :WSCconductor
- long save(WSDserialize*, char* type, char* nm, void* ptr); :WSCconductor
- long load(WSDserialize*, char* type, char* nm, void* ptr); :WSCconductor
- long saveGUI(WSDserialize*,char* name,WSCbase*); :WSCconductor
- long loadGUI(WSDserialize*,WSCbase**,WSCbase*); :WSCconductor
- WSDserialize* beginTransaction(char* serialize_name,char* field); :WSCconductor
- long endTransaction(WSDserialize*); :WSCconductor

8.6 関数仕様

8.6.1 WSGIconductor 関数の説明

書式 WSCconductor* WSGIconductor()

機能 アプリケーションにひとつ存在するストア管理クラスのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 ストア管理クラスインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル

```
#include <WSCconductor.h>
void sample_proc(){
  WSCbase* window   = NULL; //返り値(ロードしたアプリケーションウィンドウ)を格納
  char*    stype    = "FILE";           //FILE 指定
  char*    fname    = "newpic001.oof";  //FILE 名称
  char*    path     = "/usr1/win/data"; //DIR  指定
  char*    parent   = NULL; //ウィンドウをロードするので親インスタンスは指定しない。
  //読み先ディレクトリを指定。
  WSGIconductor()->setSerializePath(path);

  WSDserialize* db = WSGIconductor()->beginTransaction(stype,fname);
  if (db == NULL){
    return WS_ERR;
  }
  long ret = WSGIconductor()->loadGUI(db,\&window,parent);
  long ret2 = WSGIconductor()->endTransaction(db);
}
```

8.6.2 saveGUI 関数の説明

書式 long saveGUI(WSDserialize* dest,char* name,WSCbase* inst)

機能 指定されたインスタンスを指定された名称のシリアライズ先に出力します。

処理概要

引数

(in)WSDserialize* dest	シリアライズ先インスタンス
(in)char* name	シリアライズ名
(in)inst	ストアしたいインスタンス名

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
#include <WSCconductor.h>
void sample_proc(WSCbase* object){
  char*    stype    = "FILE";           //FILE 指定
  char*    fname    = "newpic001.oof";  //FILE 名称
  char*    path     = "/usr1/win/data"; //DIR  指定

  //読み先ディレクトリを指定。
  WSGIconductor()->setSerializePath(path);
  WSDserialize* db = WSGIconductor()->beginTransaction(stype,fname);
  if (db == NULL){
    return WS_ERR;
  }
  WSGIconductor()->saveGUI(db,object->getInstanceName(),object);
  return WSGIconductor()->endTransaction(db);

}
```

8.6.3 loadGUI 関数の説明

書式 long loadGUI(WSDserialize* src,WSCbase** inst,WSCbase* parent)

機能 指定されたシリアライズ元からインスタンスを読み込みます。

処理概要

引数

(in)WSDserialize* src	シリアライズ元インスタンス
(out)WSCbase** inst	読み込んだインスタンスを格納するポインタ
(in)WSCbase* parent	読み込むインスタンスの親インスタンス

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル WSGIappConductor() を参照してください。

8.6.4 save 関数の説明

書式 long save(WSDserialize* dest, char* type, char* name, void* ptr)

機能 指定されたデータを指定された名称のシリアライズ先に出力します。GUI インスタンスだけでない通常のデータ型のシリアライズする場合に用います。

処理概要

引数

(in)WSDserialize* dest	シリアライズ先インスタンス
(in)char* type	データのクラス名
(in)char* name	シリアライズ名
(in)void* data	出力データ

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
#include <WSCconductor.h>
void sample_proc(WSCbase* object){
  char*    stype    = "FILE";           //FILE 指定
  char*    fname    = "newpic001.oof";  //FILE 名称
  char*    path     = "/usr1/win/data"; //DIR  指定

  //読み先ディレクトリを指定。
  WSGIconductor()->setSerializePath(path);
  WSDserialize* db = WSGIconductor()->beginTransaction(stype,fname);
  if (db == NULL){
    return WS_ERR;
  }
  WSGIconductor()->save(db,object->getClassName(),object->getInstanceName(),object);
  return WSGIconductor()->endTransaction(db);

}
```

8.6.5 load 関数の説明

書式 long load(WSDserialize* dest, char* type, char* name, void* ptr)
機能 指定された名称のシリアライズ元からしていされた名称のデータを読み込みます。
処理概要
引数

(in)WSDserialize* dest	シリアライズ元インスタンス
(in)char* type	データのクラス名
(in)char* name	シリアライズ名
(in)void* data	読み込んだデータを格納するポインタ

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル

```
#include <WSCconductor.h>
void sample_proc(WSCbase* object){
  WSCbase* window   = NULL; //返り値(ロードしたアプリケーションウィンドウ)を格納
  char*    stype    = "FILE";           //FILE 指定
  char*    fname    = "newpic001.oof";  //FILE 名称
  char*    path     = "/usr1/win/data"; //DIR  指定
  char*    parent   = NULL; //ウィンドウをロードするので親インスタンスは指定しない。
  //読み先ディレクトリを指定。
  WSGIconductor()->setSerializePath(path);
  WSDserialize* db = WSGIconductor()->beginTransaction(stype,fname);
  if (db == NULL){
    return WS_ERR;
  }
  WSGIconductor()->save(db,"WSCbase","*",\&window,parent);
  return WSGIconductor()->endTransaction(db);
}
```

8.6.6 beginTransaction 関数の説明

書式 WSDserialize* beginTransaction(char* stype,char* sname)
機能 シリアライズ名を指定してストア出力を開始します。
処理概要 シリアライズ名を指定してストア出力用シリアライズインスタンスを取得します。
引数

(in)char* stype	出力タイプ名
(in)char* sname	シリアライズ出力名

復帰値 シリアライズインスタンス
注意事項
サンプル WSGIappConductor() を参照してください。

8.6.7 endTransaction 関数の説明

書式 WSDserialize* endTransaction(WSDserialize* sr)
機能 シリアライズインスタンスを指定して一連のストア出力を終了します。
処理概要
引数

(in)WSDserialize* sr	シリアライズインスタンス

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル WSGIappConductor() を参照してください。

9 WSCdialog

9.1 継承元

次のオブジェクトを継承しています。

WSCbase WSCwindow WSCbaseDialog

9.2 概要

ポップアップダイアログの基本的な機能を提供します。WSCbaseDialog に対して、クライアント領域となるフォームが追加されています。

9.3 機能

- デフォルトで OK、NO、キャンセルボタンを提供します。
- ポップアップ制御関数を提供します。

9.4 注意事項

9.5 プロパティー覧

クラス WSCdialog のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNworkBackColor	作業領域背景色	DEF10
WSCbool	WSNok	OK ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNno	NO ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	
char*	WSNokString	OK ボタン表示文字列	OK
char*	WSNnoString	NO ボタン表示文字列	NO
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	100
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	

	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

9.6 トリガー一覧

WSCdialog クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

9.7 関数一覧

9.8 関数仕様

10 WSCdirTree

10.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform WSCscrForm WSClist WSCtreeList

10.2 概要

WSCtreeList クラスを基にしたディレクトリをツリー表示するツリー型リストです。選択されたディレクトリが、プロパティWSNdirName で取得することができます。

10.3 機能

- ディレクトリのツリー表示機能
- 選択されたディレクトリのプロパティによる取得

10.4 注意事項

10.5 プロパティ一覧

クラス WSCdirTree のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNdirName	ディレクトリ名	
unsigned char	WSNitemHeight	項目縦幅	20
short	WSNiconPixmap	アイコン画	$(WSDIR)/sys/pixmaps/bi16.xpm
WSCbool	WSNuseIcon	アイコン使用	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNfont	フォント番号	0
short	WSNselectColor	選択色	DEF7
short	WSNselectForeColor	選択表示色	DEF18
WSCbool	WSNmultiSelect	複数選択	0
WSCbool	WSNreverseSelect	選択時反転	1
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNtitleHeight	タイトル縦幅	0
char*	WSNtitleString	タイトル文字列	title
char*	WSNbarValue	バー位置	20
char*	WSNdata	データ	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNenableInput	入力可否	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNseparator	セパレータ	,
unsigned short	WSNbarThickness	バー幅	16
unsigned short	WSNworkWidth	仮想領域横幅	1000
unsigned short	WSNworkHeight	仮想領域縦幅	20
unsigned short	WSNhbarValue	横スクロール位置	0
unsigned short	WSNvbarValue	縦スクロール位置	0
unsigned short	WSNincrement	ボタンスクロール単位	10
unsigned short	WSNpageIncrement	ページスクロール単位	100
WSCbool	WSNhbarVisible	横スクロールバー	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNvbarVisible	縦スクロールバー	1
	不可視	False (0)	
	可視	True (1)	
short	WSNworkBackColor	作業領域背景色	DEF11
short	WSNbarShadowColor	バー背景色	DEF12
unsigned short	WSNscrollWidth	横スクロール単位	30
unsigned char	WSNmargin	マージン	4
WSCbool	WSNvirtualScroll	仮想スクロール	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNframe	フレーム	1
	不可	False (0)	

	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

10.6 トリガー一覧

WSCdirTree クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
SORT	序列変更となった場合	WSEV_SORT
ITEM-SELECTED	項目が選択された場合	WSEV_ITEM_SELECTED
INPUT-FIXED	入力が確定状態になった場合	WSEV_INPUT_FIXED
SCR-MOUSE-PRESS	スクロールエリア内でマウスのボタンが押された場合	WSEV_SCR_MOUSE_PRESS
SCR-MOUSE-RELEASE	スクロールエリア内でマウスのボタンがはなされた場合	WSEV_SCR_MOUSE_RELEASE
SCR-MOUSE-MOVE	スクロールエリア内でマウスポインタが動いた場合	WSEV_SCR_MOUSE_MOVE

10.7 関数一覧

- long setSelectedDirName(char*,WSCbool opened = False) :WSCdirTree
- char* getSelectedDirName() :WSCdirTree

10.8 関数仕様

10.8.1 setSelectedDirName 関数の説明

書式 long setSelectedDirName(char* dirname,WSCbool opened = True)

機能 選択状態にあるディレクトリを指定します。

処理概要 opened を True にすると、指定されたディレクトリが下位のディレクトリで、表示状態にない場合、自動的に開かれ、選択表示状態になります。

引数

(in)char* dirname	ディレクトリ名
(in)WSCbool opened	表示状態

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
#include <WSCdirTree.h>
void sample_proc(WSCbase* object){
  WSCdirTree* dirtree = (WSCdirTree*)object->cast("WSCdirTree");
  if (dirtree == NULL){
    return;
  }
  dirtree->setSelectedDirName("/home/users");
}
```

10.8.2 getSelectedDirName 関数の説明

書式 char* getSelectedDirName()

機能 選択状態にあるディレクトリを取得します。

処理概要 現在選択状態にあるディレクトリを取得します。

引数 なし。

復帰値 ディレクトリ名

注意事項 返されたディレクトリ名を開放してはいけません。

サンプル

```
#include <WSCdirTree.h>
void sample_proc(WSCbase* object){
  WSCdirTree* dirtree = (WSCdirTree*)object->cast("WSCdirTree");
```

```
    if (dirtree == NULL){
      return;
    }
    char* selected_dir = dirtree->getSelectedDirName();
  }
```

11 WSCfform

11.1 継承元

次のオブジェクトを継承しています。
WSCbase

11.2 概要

タブを掴んでウィンドウとして取り外せるフォームです。ウィンドウ状態で再びタブを掴んで元の位置に移動すると、元のフォーム状態に戻すことができます。

11.3 機能

- ウィンドウとして取り外すことのできるフォーム
- マウスで掴んで移動するためのタブの表示

11.4 注意事項

11.5 プロパティー覧

クラス WSCfform のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
char*	WSNtitleString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	

	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNorientation	表示方向	1
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

11.6 トリガー一覧

WSCfform クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
STATUS-CH	ステータスに変化があった場合	WSEV_STATUS_CH

11.7 関数一覧

- void setFloated(WSCbool fl) :WSCffrom
- WSCbool getFloated() :WSCffrom

11.8 関数仕様

11.8.1 setFloated 関数の説明

書式 void setFloated(WSCbool fl)

機能 フォームの状態を指定します。

処理概要 True を設定すると、ウィンドウ状態になり、False を設定すると、フォーム状態となります。

引数

(in)WSCbool fl	フォームの状態

復帰値 なし。

注意事項

サンプル

```
#include <WSCfform.h>
void sample_proc(WSCbase* object){
  WSCfform* fform = (WSCfform*)object->cast("WSCfform");
  if (fform == NULL){
    return;
  }
  fform->setFloated(True);
}
```

11.8.2 getFloated 関数の説明

書式 WSCbool getFloated()

機能 フォームの状態を取得します。

処理概要 ウィンドウ状態の場合 True 、フォーム状態の場合、False を返します。

引数 なし。

復帰値 フォームの状態

注意事項

サンプル

```
#include <WSCfform.h>
void sample_proc(WSCbase* object){
  WSCfform* fform = (WSCfform*)object->cast("WSCfform");
  if (fform == NULL){
    return;
  }
  WSCbool fl = fform->getFloated();
  if (fl == False){
    //フォーム状態
  }else{
    //ウィンドウ状態
  }
}
```

12 WSCfileSelect

12.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow WSCbaseDialog

12.2 概要

ファイルを選択するためのダイアログです。ディレクトリの移動、拡張子によるファイルのマスキング、ファイルの選択ができます。

12.3 機能

- マウスによるディレクトリの移動とファイルの選択

12.4 注意事項

12.5 プロパティー覧

クラス WSCfileSelect のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNdirName	ディレクトリ名	/mnt3/wsdoc2/dumptool/wsautodump/object
char*	WSNfileName	ファイル名	
char*	WSNmaskFileName	ファイル拡張子名	
WSCbool	WSNselectDir	ディレクトリ選択	0
	オフ	False (0)	
	オン	True (1)	
short	WSNworkBackColor	作業領域背景色	DEF10
WSCbool	WSNok	OK ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	$(WSDIR)/sys/pixmaps/bi13.xpm
char*	WSNokString	OK ボタン表示文字列	OK
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	FileSelection
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	250
unsigned short	WSNheight	縦幅	300
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0

	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

12.6 トリガー一覧

WSCfileSelect クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

12.7 関数一覧

- char* getFileName() :WSCfileSelect

12.8 関数仕様

12.8.1 getFileName 関数の説明

書式 char* getFileName()

機能 ユーザによって選択されたファイル名を取得します。

処理概要

引数 なし。

復帰値 ファイル名称

注意事項 取得されるファイル名はフルパスで取得されます。返された領域を開放してはいけません。また、ファイル名称とディレクトリ名称を個別に取得したい場合は、プロパティWSNfileName と、WSNdirName で取得してください。

サンプル

```
#include <WSCfileSelect.h>
extern WSCfileSelect* newfsel_000;
void sample_proc(WSCbase* object){
  //ファイル選択ダイアログの表示
  long ret = newfsel_000->popup();
  //選択された場合
```

```
  if (ret == WS_DIALOG_OK){
    //選択されたファイル名
    char* fname = fselect->getFileName();
  }
}
```

13 WSCfontSet

13.1 継承元

次のオブジェクトを継承しています。

13.2 概要

WSDfont フォントクラスのインスタンスを管理します。フォントに関する情報を取得したい場合にこのクラスを利用します。

13.3 機能

- WSDfont フォントクラスのインスタンスの管理

13.4 注意事項

13.5 関数一覧

- WSCfontSet* WSGIappFontSet(); :WSCfontSet
- WSDfont* getFont(short fno); :WSCfontSet
- short getDefaultFontNo(); :WSCfontSet
- WSDfont* getDefaultFont(); :WSCfontSet

13.6 関数仕様

13.6.1 WSGIappFontSet 関数の説明

書式 WSCfontSet* WSGIappFontSet()

機能 アプリケーションにひとつ存在するフォント管理クラスのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 フォント管理クラスインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル

```
//フォント番号を指定して、フォントインスタンスを取得します。
WSDfont* font = WSGIappFontSet()->getFont(fno);
```

13.6.2 getFont 関数の説明

書式 WSDfont* getFont(short fno)

機能 フォント番号からフォントインスタンスを取得します。

処理概要

引数

(in)short fno	フォント番号

復帰値 フォントインスタンス

注意事項 返されたフォントインスタンスを開放してはなりません。

サンプル WSGIappFontSet() を参照してください。

13.6.3 getDefaultFontNo 関数の説明

書式 short getDefaultFontNo()
機能 指定されたデフォルトフォントのフォント番号を取得します。
処理概要
引数 なし。
復帰値 フォント番号
注意事項
サンプル

```
short fno = WSGIappFontSet()->getDefaultFontNo();
```

13.6.4 getDefaultFont 関数の説明

書式 WSDfont* getDefaultFont()
機能 デフォルトフォントのインスタンスを取得します。
処理概要
引数 なし。
復帰値 フォントインスタンス
注意事項 返されたフォントインスタンスを解放してはなりません。
サンプル

```
WSDfont* font = WSGIappFontSet()->getDefaultFont();
```

14 WSCform

14.1 継承元

次のオブジェクトを継承しています。
WSCbase

14.2 概要

矩形のウィンドウ領域で、他のオブジェクトを内部に配置し、子として管理する機能を提供します。

14.3 機能

- 上位のウィンドウとは独立した矩形の領域の提供
- 他のオブジェクトの配置と管理機能
- 枠の表示機能
- 背景ピクスマップの表示機能

14.4 注意事項

WSClistData* getChildren() 関数により、内部に配置している子オブジェクトを取得することができます。

14.5 プロパティー覧

クラス WSCform のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2

short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

14.6 トリガー一覧

WSCform クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

14.7 関数一覧

14.8 関数仕様

15 WSCgrid

15.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

15.2 概要

デリミタで区切られた文字列データを表形式に表示します。

15.3 機能

- 文字列データの表形式による表示
- 罫線の表示
- 各セルへの直接入力

15.4 注意事項

15.5 プロパティ一覧

クラス WSCgrid のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNlineX	線のX座標	25,50
char*	WSNlineY	線のY座標	30,70
unsigned long	WSNvcolumns	縦カラム数	3
unsigned long	WSNhcolumns	横カラム数	3
char*	WSNdata	データ	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNenableInput	入力可否	0
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	

	右配置	WS_RIGHT (1)	
unsigned char	WSNlineWidth	線幅	1
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF11
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	

WSCbool	WSNexport 不可 可	エクスポート False (0) True (1)	0

15.6 トリガー一覧

WSCgrid クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
ITEM-SELECTED	項目が選択された場合	WSEV_ITEM_SELECTED

15.7 関数一覧

- long getCellGeometry(WSCulong cx,WSCulong cy,short* x,short* y, WSCushort* w,WSCushort* h) :WSCgrid
- long getHColumns() :WSCgrid
- long getVColumns() :WSCgrid
- long setItem(WSCulong cx,WSCulong cy,WSCvariant) :WSCgrid
- long setCellForeColor(WSCulong cx,WSCulong cy,short color) :WSCgrid
- long setCellForeColor(WSCulong cx,WSCulong cy,char* cname) :WSCgrid
- long setCellBackColor(WSCulong cx,WSCulong cy,short color) :WSCgrid
- long setCellBackColor(WSCulong cx,WSCulong cy,char* cname) :WSCgrid
- long getItemAlignment(WSCulong hpos,WSCulong vpos) :WSCgrid
- WSCvariant getItem(WSCulong hpos,WSCulong vpos) :WSCgrid

15.8 関数仕様

15.8.1 getCellGeometry 関数の説明

書式 long getCellGeometry(WSCulong cx,WSCulong cy, short* x,short* y, WSCushort* w,WSCushort* h)

機能 指定されたセルの大きさを取得します。

処理概要 横:cx 、縦:cy で指定されたセルの大きさをグリッドインスタンスの座標系での取得します。

引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置
(out)short* x	X座標
(out)short* y	Y座標
(out)WSCshort* w	横幅
(out)WSCshort* h	縦幅

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
#include <WSCgrid.h>
void sample_proc(WSCbase* object){
  WSCgrid* grid = (WSCgrid*)object->cast("WSCgrid");
  if (grid == NULL){
    return;
  }
  //セルの数を取得し、個々のセルの大きさを表示。
  short xcells = grid->getHColumns();
  short ycells = grid->getVColumns();
  short cx,cy;
```

```
  for(cx=0; cx < xcells; cx++){
    for(cy=0; cy < ycells; cy++){
      short x,y;
      WSCushort w,h;
      //サイズを取得
      grid->getCellGeometry(cx,cy,\&x,\&y,\&w,\&h);
      //値を取得
      WSCvariant val = grid->getItem(cx,cy);
      printf("cell(%d,%d):  x,y=%d,%d w,h=%d,%d val=%s\n",cx,cy,x,y,w,h,
                                                          (char*)val);
    }
  }
}
```

15.8.2 getHColumns 関数の説明

書式 long getHColumns();
機能 横方向のセルの数を取得します。
処理概要
引数 なし。
復帰値 セル数
注意事項
サンプル getCellGeometry() を参照してください。

15.8.3 getVColumns 関数の説明

書式 long getVColumns();
機能 縦方向のセルの数を取得します。
処理概要
引数 なし。
復帰値 セル数
注意事項
サンプル getCellGeometry() を参照してください。

15.8.4 setItem 関数の説明

書式 long setItem(WSCulong cx,WSCulong,cy,WSCvariant val);
機能 指定されたセルに値を設定します。
処理概要 横:cx 、縦:cy で指定されたセルに、WSCvariant 型で値を指定します。
引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置
(in)WSCvariant val	値

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル getCellGeometry() を参照してください。

15.8.5 setCellForeColor 関数の説明

書式 long setCellForeColor(WSCulong cx,WSCulong cy,short color);
機能 指定されたセルに表示色を設定します。
処理概要 横:cx 、縦:cy で指定されたセルに、色番号で表示色を指定します。
引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置
(in)short color	色番号

復帰値　WS_NO_ERR= 正常、それ以外はエラー。

サンプル
```
//黒の色番号を取得し、左上のセルの表示色に設定。
short cno = WSGIappColorSet()->getColorNo("#000000");
newgrid_000->setCellForeColor(0,0,cno);
```

注意事項

15.8.6 setCellForeColor 関数の説明

書式　long setCellForeColor(WSCulong cx,WSCulong cy,char* cname);

機能　指定されたセルに表示色を設定します。

処理概要　横:cx 、縦:cy で指定されたセルに、色名称で表示色を指定します。

引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置
(in)char* cname	色名称

色名称は、次のフォーマットで指定します。
#RRGGBB
RR: ００～ｆｆの範囲で１６進数で赤の輝度を指定します。
GG: ００～ｆｆの範囲で１６進数で緑の輝度を指定します。
BB: ００～ｆｆの範囲で１６進数で青の輝度を指定します。

復帰値　WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル
```
//左上のセルの表示色に黒を設定。
newgrid_000->setCellForeColor(0,0,"#ffffff");
```

15.8.7 setCellBackColor 関数の説明

書式　long setCellBackColor(WSCulong cx,WSCulong cy,short color);

機能　指定されたセルに背景色を設定します。

処理概要　横:cx 、縦:cy で指定されたセルに、色番号で表示色を指定します。

引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置
(in)short color	色番号

復帰値　WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル
```
//白の色番号を取得し、左上のセルの背景色に設定。
short cno = WSGIappColorSet()->getColorNo("#ffffff");
newgrid_000->setCellBackColor(0,0,cno);
```

15.8.8 setCellBackColor 関数の説明

書式　long setCellBackColor(WSCulong cx,WSCulong cy,char* cname);

機能　指定されたセルに表示色を設定します。

処理概要　横:cx 、縦:cy で指定されたセルに、色名称で表示色を指定します。

引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置
(in)char* cname	色名称

色名称は、次のフォーマットで指定します。
#RRGGBB
RR: ００～ｆｆの範囲で１６進数で赤の輝度を指定します。
GG: ００～ｆｆの範囲で１６進数で緑の輝度を指定します。
BB: ００～ｆｆの範囲で１６進数で青の輝度を指定します。

復帰値　WS_NO_ERR= 正常、それ以外はエラー。

注意事項
```
//左上のセルの背景色に白を設定。
newgrid_000->setCellBackColor(0,0,"#000000");
```

15.8.9 getItemAlignment 関数の説明

書式 long getItemAlignment(WSCulong cx,WSCulong cy);

機能 指定されたセルのアラインメントを取得します。

処理概要 横:cx 、縦:cy で指定されたセルのアラインメントを取得します。

引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置

復帰値 アラインメント
アラインメントには次のような値があります。

値	意味
WS_LEFT	左
WS_RIGHT	右
WS_CENTER	中央
WS_TOP	上
WS_BOTTOM	下
WS_LEFT_TOP	左上
WS_LEFT_BOTTOM	左下
WS_RIGHT_TOP	右上
WS_RIGHT_BOTTOM	右下

注意事項

サンプル

```
//左上のセルのアラインメントを取得。
WSCuchar alignment = newgrid_000->getItemAlignment(0,0);
```

15.8.10 getItem 関数の説明

書式 WSCvariant getItem(WSCulong cx,WSCulong cy);

機能 指定されたセルの値を取得します。

処理概要 横:cx 、縦:cy で指定されたセルの値を取得します。

引数

(in)long cx	セルの横位置
(in)long cy	セルの縦位置

復帰値 セルの値

サンプル getCellGeometry() を参照してください。

注意事項

16 WSChorzForm

16.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform WSCvertForm

16.2 概要

配列機能を持ったフォームです。このフォームは、このフォームに配置されたインスタンスを横に並べます。サイズが縮められた場合、プロパティ WSNminimum で指定されたサイズまで縮められます。

16.3 機能

- インスタンスを横に並べます。

16.4 注意事項

16.5 プロパティー覧

クラス WSChorzForm のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNalignmentV	縦配置	0
	なし	(0)	
	上配置	(1)	
	中央配置	WS_CENTER (2)	
	下配置	WS_UP (3)	
	アジャスト	WS_DOWN (4)	
unsigned short	WSNminimum	最小値	20
unsigned char	WSNmargin	マージン	4
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	

	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

16.6 トリガー一覧

WSChorzForm クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

16.7 関数一覧

16.8 関数仕様

17 WSCimageSet

17.1 継承元

次のオブジェクトを継承しています。

17.2 概要

WSDimage 画像クラスのインスタンスを管理します。画像に関する情報を取得したい場合にこのクラスを利用します。

17.3 機能

- WSDimage 画像クラスのインスタンスの管理

17.4 注意事項

17.5 関数一覧

- WSCimageSet* WSGIappImageSet(); :WSCimageSet
- WSDimage* getImage(short no); :WSCimageSet
- WSDimage* getImage(char* name); :WSCimageSet
- short getImageNo(char*); :WSCimageSet
- char* getImageName(short no); :WSCimageSet
- long destroyImage(short no); :WSCimageSet
- long destroyImage(char* name); :WSCimageSet

17.6 関数仕様

17.6.1 WSGIappImageSet 関数の説明

書式 WSCimageSet* WSGIappImageSet()

機能　アプリケーションにひとつ存在する画像管理クラスのグローバルインスタンスを取得します。
処理概要
引数　なし。
復帰値　画像管理クラスインスタンスへのポインタ
注意事項　この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。
サンプル

```
//画像ファイル名称を指定して、イメージインスタンスを取得します。
WSDimage* image = WSGIappImageSet()->getImage("001.jpg");
```

17.6.2 getImage 関数の説明

書式　WSDimage* getImage(short no)
機能　画像番号から画像インスタンスを取得します。
処理概要
引数

(in)short no	画像番号

復帰値　色インスタンス
注意事項　返された画像インスタンスを開放してはなりません。
サンプル

```
//画像ファイル名称を指定して、イメージ番号を取得します。
short ino = WSGIappImageSet()->getImageNo("001.jpg");
//イメージ番号を指定して、イメージインスタンスを取得します。
WSDimage* image = WSGIappImageSet()->getImage(ino);
```

17.6.3 getImage 関数の説明

書式　WSDimage* getImage(char* name)
機能　画像ファイル名称から画像インスタンスを取得します。
処理概要
引数

(in)char* name	画像ファイル名称

復帰値　画像インスタンス
注意事項　返された画像インスタンスを開放してはなりません。
サンプル　WSGIappImageSet() を参照してください。

17.6.4 getImageNo 関数の説明

書式　short getImageNo(char* name)
機能　画像ファイル名称から画像番号を取得します。
処理概要
引数

(in)char*	画像ファイル名称

復帰値　画像番号
注意事項
サンプル　WSGIappImageSet() を参照してください。

17.6.5 getImageName 関数の説明

書式　char* getImageName(short no)
機能　指定された画像番号の画像ファイル名称を取得します。
処理概要
引数

(in)short no	画像番号

復帰値　画像ファイル名称

注意事項 返された画像ファイル名称を解放してはなりません。
サンプル
```
//イメージ番号を指定して、画像ファイル名称を取得します。
char* fname = WSGIappImageSet()->getImageName(ino);
```

17.6.6 destroyImage 関数の説明

書式 long destroyImage(char* fname)
機能 指定された画像ファイル名称によって、読み込んだ画像を破棄します。
処理概要 読み込んだ領域を破棄し、再読み込みを可能にします。画像ファイルが更新される場合の再読み込みや、既に表示の必要がなくなったものを開放する場合に、呼び出します。
引数

(in)char* fname	画像ファイル名称

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル
```
//読み込んだ画像ファイル名称を指定して、イメージ領域を破棄します。
WSGIappImageSet()->destroyImage("001.jpg");
```

17.6.7 destroyImage 関数の説明

書式 long destroyImage(short no)
機能 指定された画像番号によって、読み込んだ画像を破棄します。
処理概要
引数

(in)short no	画像番号

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル
```
//イメージ番号を指定して、読み込んだ画像ファイルのイメージ領域を破棄します。
short ino = WSGIappImageSet()->getImageNo("001.jpg");
WSGIappImageSet()->destroyImage(ino);
```

18 WSCindexData

18.1 継承元

次のオブジェクトを継承しています。

18.2 概要

汎用リストデータ構造体です。メモリ管理を内部で自動的に行いますので、簡単にデータの追加等が行えます。WSClistDataとは違い、データにインデックスを割り当てることができます。

18.3 機能

- 自由なデータのリスト化
- リストのメモリ管理
- データに対するインデックスの割り当て

18.4 注意事項

18.5 関数一覧

- WSCindexData(); :WSCindexData
- WSCindexData(WSCindexData&); :WSCindexData
- long setData(char* data,void* data); :WSCindexData

- char* getIndex(long pos); :WSCindexData
- void* getData(char*); :WSCindexData
- void* getData(long pos); :WSCindexData
- long getNum(); :WSCindexData
- long del(char* data); :WSCindexData
- long delPos(long i); :WSCindexData
- void clear(); :WSCindexData
- operator [] :WSCindexData
- operator = :WSCindexData

18.6 関数仕様

18.6.1 WSCindexData 関数の説明

書式 WSCindexData()

機能 インデックス付きリストデータ構造体のコンストラクタです。インデックス付きリストデータのインスタンスを一つ作成します。

処理概要

引数 なし。

復帰値 インデックス付きリストデータのインスタンスへのポインタ

注意事項

サンプル

```
WSCindexData  indexdata;
long data1 = 100;
long data2 = 200;

//値をインデックスを付けて記憶させます。
indexdata["data1"]  = (void*)data1;
indexdata["data2"]  = (void*)data2;

//同じことを setData() でおこなうと、次のようになります。
indexdata.setData("data1",(void*)data1);
indexdata.setData("data2",(void*)data2);

//インデックスを指定して、記憶したデータを取り出します。
void* val1 = indexdata["data1"];
void* val2 = indexdata["data2"];

//同じことを getData() でおこなうと、次のようになります。
void* val3 = indexdata.getData("data1");
void* val4 = indexdata.getData("data2");

//インデックスを取得するには次のようにします。
char* index1 = indexdata.getIndex(0); //"data1" が返されます。
char* index2 = indexdata.getIndex(1); //"data2" が返されます。

//位置で値を取得するには次のようにします。
void* val5 = indexdata.getData(0);
void* val6 = indexdata.getData(1);

//データの数を取得するには、次のようにします。
long num = indexdata.getNum();

//インデックスを指定してデータを削除するには次のようにします。
indexdata.del("data2");

//位置を指定してデータを削除するには次のようにします。
indexdata.delPos(0);

//全てのデータを一度にクリアする場合は、次のようにします。
indexdata.clear();
```

18.6.2 WSCindexData 関数の説明

書式 WSCindexData(WSCindexData& src)

機能 インデックス付きリストデータ構造体のコピーコンストラクタです。与えられたリストと同じものを複製します。

処理概要

引数

(in)WSCindexData& src	コピー元インスタンス

復帰値 インデックス付きリストデータのインスタンスへのポインタ

注意事項

サンプル

```
//値をインデックスを付けて記憶させます。
WSCindexData  indexdata;
indexdata ["data1"]  = (void*)data1;
indexdata ["data2"]  = (void*)data2;
//べつの indexdata2 にコピーコンストラクタでコピーします。
WSCindexData  indexdata2(indexdata);
```

18.6.3 setData 関数の説明

書式 long setData(char* index,void* data)

機能 指定したインデックスでデータを格納します。

処理概要

引数

(in)char* index	インデックス名
(in)void* data	データ

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル WSCindexData() を参照してください。

18.6.4 getIndex 関数の説明

書式 char* getIndex(long pos)

機能 指定した位置のインデックス名を取得します。

処理概要

引数

(in)long pos	データの位置 (先頭は0)

復帰値 インデックス名

注意事項 取得されたインデックス名を解放してはなりません。

サンプル WSCindexData() を参照してください。

18.6.5 getData 関数の説明

書式 void* getData(char* index)

機能 指定したインデックスに対応したデータを取得します。

処理概要

引数

(in)char* index	インデックス

復帰値 格納したデータ

注意事項 インデックスに対応したデータがない場合、0 を返します。

サンプル WSCindexData() を参照してください。

18.6.6 getData 関数の説明

書式 void* getData(long pos)
機能 指定した位置のデータを取得します。
処理概要
引数

(in)long pos	データの位置 (先頭は 0)

復帰値 データ
注意事項
サンプル WSCindexData() を参照してください。

18.6.7 getNum 関数の説明

書式 long getNum()
機能 リストの持つデータ数を取得します。
処理概要
引数 なし。
復帰値 データ数
注意事項
サンプル WSCindexData() を参照してください。

18.6.8 del 関数の説明

書式 long del(char* index);
機能 指定したインデックスに対応したデータを削除します。
処理概要
引数

(in)char* index	インデックス

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項

18.6.9 delPos 関数の説明

書式 long delPos(long pos);
機能 指定した位置のデータを削除します。
処理概要
引数

(in)long pos	データの位置 (先頭は 0)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル WSCindexData() を参照してください。

18.6.10 clear 関数の説明

書式 void clear();
機能 保持しているデータを全て削除します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル WSCindexData() を参照してください。

18.6.11 [ ] オペレータの説明

書式 void*& WSCindexData::operator [] (char* index)
機能 指定されたインデックスに対応したデータを取得します。
処理概要
引数

(in)char* index	インデックス

復帰値 データ
注意事項
サンプル WSCindexData() を参照してください。

18.6.12 = オペレータの説明

書式 WSCindexData& operator = (WSCindexData& src)
機能 コピーオペレータです。
処理概要
引数

(in)WSCindexData& src	コピー元のインスタンス

復帰値 コピー後のインスタンス
注意事項
サンプル

```
//値をインデックスを付けて記憶させます。
WSCindexData  indexdata;
indexdata ["data1"]  = (void*)data1;
indexdata ["data2"]  = (void*)data2;
//べつの indexdata2 にコピーコンストラクタでコピーします。
WSCindexData  indexdata2;
//コピーオペレータが実行されます。
indexdata2 = indexdata;
```

19 WSCindexForm

19.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

19.2 概要

インデックスタブを持った、切替え表示機能を提供します。

19.3 機能

- インデックスタブによる切替え表示機能

19.4 注意事項

インデックスフォーム上に配置されたオブジェクトは、プロパティWSNuserValue(ユーザ設定値) が使えなくなります。インデックスフォームは、その WSCuserValue の値を使って、切替え表示の制御を行います。したがって、配置されたオブジェクトの WSNuserValue プロパティを変更すると、表示が正しく行われないことがあります。

19.5 プロパティー覧

クラス WSCindexForm のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNbarThickness	バー幅	20
char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
char*	WSNmenuItems	メニュー項目	index1,index2
unsigned char	WSNvalue	値	0
WSCbool	WSNflip	反転表示	0
	不可	False (0)	
	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0

不可	False (0)
可	True (1)

19.6 トリガー一覧

WSCindexForm クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

19.7 関数一覧

19.8 関数仕様

20 WSCindexVariantData

20.1 継承元

次のオブジェクトを継承しています。

20.2 概要

汎用リストデータ構造体です。メモリ管理を内部で自動的に行いますので、簡単にデータの追加等が行えます。WSCindexDataとは違い、バリアント型データを使用します。

20.3 機能

- 自由なデータのリスト化
- リストのメモリ管理
- バリアント型データに対するインデックスの割り当て

20.4 注意事項

20.5 関数一覧

- WSCindexVariantData(); :WSCindexVariantData
- WSCindexVariantData(WSCindexVariantData&); :WSCindexVariantData
- long setData(char* data,WSCvariant data); :WSCindexVariantData
- char* getIndex(long pos); :WSCindexVariantData
- WSCvariant &getData(char*); :WSCindexVariantData
- WSCvariant &getData(long pos); :WSCindexVariantData
- long getNum(); :WSCindexVariantData
- long del(char* data); :WSCindexVariantData
- long delPos(long i); :WSCindexVariantData
- void clear(); :WSCindexVariantData
- operator [ ] :WSCindexVariantData
- operator = :WSCindexVariantData

20.6 関数仕様

20.6.1 WSCindexVariantData 関数の説明

書式 WSCindexVariantData()

機能 インデックス付きバリアント型データリスト構造体のコンストラクタです。インデックス付きバリアント型データリストのインスタンスを一つ作成します。

処理概要

引数 なし。

復帰値 インデックス付きバリアント型データリストのインスタンスへのポインタ

注意事項

サンプル

```
WSCindexVariantData  indexdata;
long data1 = 100;
char* str1 = "data";

//値をインデックスを付けて記憶させます。
indexdata ["data1"]  = data1;
indexdata ["data2"]  = str1;

//同じことを setData() でおこなうと、次のようになります。
indexdata.setData("data1",data1);
indexdata.setData("data2",str1);

//インデックスを指定して、記憶したデータを取り出します。
long val1 = indexdata ["data1"] ;
WSCstring val2 = indexdata ["data2"] ;

//同じことを getData() でおこなうと、次のようになります。
long val3 = indexdata.getData("data1");
WSCstring val4 = indexdata.getData("data2");

//インデックスを取得するには次のようにします。
char* index1 = indexdata.getIndex(0); //"data1" が返されます。
char* index2 = indexdata.getIndex(1); //"data2" が返されます。

//位置で値を取得するには次のようにします。
long val5 = indexdata.getData(0);
WSCstring val6 = indexdata.getData(1);

//データの数を取得するには、次のようにします。
long num = indexdata.getNum();

//インデックスを指定してデータを削除するには次のようにします。
indexdata.del("data2");

//位置を指定してデータを削除するには次のようにします。
indexdata.delPos(0);

//全てのデータを一度にクリアする場合は、次のようにします。
indexdata.clear();
```

20.6.2 WSCindexVariantData 関数の説明

書式 WSCindexVariantData(WSCindexVariantData&)

機能 インデックス付きバリアント型データリスト構造体のコピーコンストラクタです。与えられたリストと同じものを複製します。

処理概要

引数 なし。

復帰値 インデックス付きバリアント型データリストのインスタンスへのポインタ
注意事項
サンプル

```
//値をインデックスを付けて記憶させます。
WSCindexVariantData  indexdata;
indexdata ["data1"]  = (void*)data1;
indexdata ["data2"]  = (void*)data2;

//べつの indexdata2 にコピーコンストラクタでコピーします。
WSCindexVariantData  indexdata2(indexdata);
```

20.6.3 setData 関数の説明

書式 long setData(char* index,WSCvariant data)
機能 指定したインデックスでバリアント型データを格納します。バリアント型なので、いろいろな型を直接してすることができます。
処理概要
引数

(in)char* index	インデックス名
(in)WSCvariant data	データ

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.4 getIndex 関数の説明

書式 char* getIndex(long pos)
機能 指定した位置のインデックス名を取得します。
処理概要
引数

(in)long pos	データの位置 (先頭は 0)

復帰値 インデックス名
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.5 getData 関数の説明

書式 WSCvariant &getData(char* index)
機能 指定したインデックスに対応したデータを取得します。
処理概要
引数

(in)char* index	インデックス

復帰値 格納したデータ
注意事項 インデックスに対応したデータがない場合、0 を返します。
サンプル WSCindexVariantData() を参照してください。

20.6.6 getData 関数の説明

書式 WSCvariant &getData(long pos)
機能 指定した位置のデータを取得します。
処理概要
引数

(in)long pos	データの位置 (先頭は 0)

復帰値 データ
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.7 getNum 関数の説明

書式 long getNum()
機能 リストの持つデータ数を取得します。
処理概要
引数 なし。
復帰値 データ数
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.8 del 関数の説明

書式 long del(char* index);
機能 指定したインデックスに対応したデータを削除します。
処理概要
引数

(in)char* index	インデックス

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.9 delPos 関数の説明

書式 long delPos(long pos);
機能 指定した位置のデータを削除します。
処理概要
引数

(in)long pos	データの位置 (先頭は 0)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.10 clear 関数の説明

書式 void clear();
機能 保持しているデータを全て削除します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.11 [ ] オペレータの説明

書式 WSCvariant& operator [] (char* index)
機能 指定されたインデックスに対応したデータを取得します。
処理概要
引数

(in)char* index	インデックス

次のように使います。
復帰値 データ
注意事項
サンプル WSCindexVariantData() を参照してください。

20.6.12 = オペレータの説明

書式 WSCindexVariantData& operator = (WSCindexVariantData& src)
機能 コピーオペレータです。
処理概要
引数

(in)WSCindexVariant& src	コピー元インスタンス

復帰値 コピー後のインスタンス
注意事項
サンプル

```
//値をインデックスを付けて記憶させます。
WSCindexData  indexdata;
indexdata ["data1"]  = (void*)data1;
indexdata ["data2"]  = (void*)data2;

//べつの indexdata2 にコピーコンストラクタでコピーします。
WSCindexData  indexdata2;

//コピーオペレータが実行されます。
indexdata2 = indexdata;
```

21 WSCinputDialog

21.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow WSCbaseDialog

21.2 概要

キー入力を行うダイアログを提供します。文字列を、WSNlabelString プロパティに指定して、popup() を実行します。

21.3 機能

・ダイアログでのキー入力機能

21.4 注意事項

21.5 プロパティー覧

クラス WSCinputDialog のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNlabelString	表示文字列	
short	WSNworkBackColor	作業領域背景色	DEF10
WSCbool	WSNok	OK ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNno	NO ボタン	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	
char*	WSNokString	OK ボタン表示文字列	OK
char*	WSNnoString	NO ボタン表示文字列	NO
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0

	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	100
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

21.6 トリガー一覧

WSCinputDialog クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

21.7 関数一覧

21.8 関数仕様

22 WSClist

22.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform WSCscrForm

22.2 概要

文字列の一覧表示、ツリー表示、選択等を行う機能を提供します。

22.3 機能

- 文字列の一覧表示機能
- ツリー表示機能
- 文字列の詳細一覧表示とソート機能

22.4 注意事項

プロパティ WSNdata はデータソース対象プロパティとして用いられます。ACTIVATE トリガに関して setEnableActivate 関数で True が指定された場合、スクロールの完了以外に項目が選択された場合にもトリガを発生するようになります。また、スクロールエリア内は、トリガー、WSEV_MOUSE_PRESS、WSEV_MOUSE_MOVE、WSEV_MOUSE_RELEASE は効きません。スクロールエリア内で、これらのトリガーを使用したい場合は、WSEV_SCR_MOUSE_PRESS、WSEV_SCR_MOUSE_MOVE、WSEV_SCR_MOUSE_RELEASE をご使用ください。

22.5 プロパティー覧

クラス WSClist のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNitemHeight	項目縦幅	20
short	WSNiconPixmap	アイコン画	
WSCbool	WSNuseIcon	アイコン使用	0
	不可	False (0)	
	可	True (1)	
unsigned char	WSNfont	フォント番号	0
short	WSNselectColor	選択色	DEF7
short	WSNselectForeColor	選択表示色	DEF18

unsigned char	WSNtype	表示タイプ	0
	ノーマル	(0)	
	ツリー	(1)	
	詳細	(2)	
WSCbool	WSNmultiSelect	複数選択	0
WSCbool	WSNreverseSelect	選択時反転	1
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNtitleHeight	タイトル縦幅	0
char*	WSNtitleString	タイトル文字列	title
char*	WSNbarValue	バー位置	20
char*	WSNdata	データ	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNenableInput	入力可否	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNseparator	セパレータ	,
unsigned short	WSNbarThickness	バー幅	16
unsigned short	WSNworkWidth	仮想領域横幅	1000
unsigned short	WSNworkHeight	仮想領域縦幅	1000
unsigned short	WSNhbarValue	横スクロール位置	0
unsigned short	WSNvbarValue	縦スクロール位置	0
unsigned short	WSNincrement	ボタンスクロール単位	10
unsigned short	WSNpageIncrement	ページスクロール単位	100
WSCbool	WSNhbarVisible	横スクロールバー	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNvbarVisible	縦スクロールバー	1
	不可視	False (0)	
	可視	True (1)	
short	WSNworkBackColor	作業領域背景色	DEF11
short	WSNbarShadowColor	バー背景色	DEF12
unsigned short	WSNscrollWidth	横スクロール単位	30
unsigned char	WSNmargin	マージン	4
WSCbool	WSNvirtualScroll	仮想スクロール	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNframe	フレーム	1
	不可	False (0)	
	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0

unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

22.6 トリガー一覧

WSClist クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
SORT	序列変更となった場合	WSEV_SORT
ITEM-SELECTED	項目が選択された場合	WSEV_ITEM_SELECTED
INPUT-FIXED	入力が確定状態になった場合	WSEV_INPUT_FIXED
SCR-MOUSE-PRESS	スクロールエリア内でマウスのボタンが押された場合	WSEV_SCR_MOUSE_PRESS
SCR-MOUSE-RELEASE	スクロールエリア内でマウスのボタンがはなされた場合	WSEV_SCR_MOUSE_RELEASE
SCR-MOUSE-MOVE	スクロールエリア内でマウスポインタが動いた場合	WSEV_SCR_MOUSE_MOVE

22.7 関数一覧

- void setLabelHeight(WSCshort h) :WSClist
- void delAll() :WSClist
- WSCbase* getLabel(long pos) :WSClist
- long getNum() :WSClist
- WSCstring getItem(long pos) :WSClist
- void addItem(char* item,long pos =-1) :WSClist
- void replaceItem(char* item,long pos) :WSClist
- void delPos(long pos) :WSClist
- long getSelectedPos() :WSClist
- long setSelectPos() :WSClist
- WSCstring getSelectedItem() :WSClist
- void setTopPos(long pos) :WSClist
- void setBottomPos(long pos) :WSClist
- void updateList() :WSClist
- void setLabelClass(char*) :WSClist
- WSCbool getSelectItemChanged() :WSClist
- void setSelectItemChanged(WSCbool) :WSClist
- WSClistData* getLabels() :WSClist
- void getSelectedLabels(WSClistData &) :WSClist
- long setItemVisible(long pos,WSCbool fl) :WSClist
- void setEnableActivate(WSCbool) :WSClist
- WSCbool getEnableActivate() :WSClist
- void setAbsoluteChangeSelectFlag(WSCbool) :WSClist
- WSCbool getAbsoluteChangeSelectFlag() :WSClist
- long setItemValue(long pos,long kind,long val) :WSClist

- void getSortPos() :WSClist
- long getTopPos() :WSClist
- long getBottompPos() :WSClist
- virtual long onSort() :WSClist
- virtual long onItemSelected() :WSClist
- virtual long onInputFixed(WSCstring str,long pos,long column) :WSClist
- virtual long onKey(WSDkeyboard* keyboard,WSCbool keydown) :WSClist
- void cancelInput() :WSClist
- WSCstring getInputString() :WSClist
- void onKey(WSDkeyboard*,WSCbool) :WSClist

22.8 関数仕様

22.8.1 setLabelHeight 関数の説明

書式 void setLabelHeight(WSCushort* height)

機能 表示項目の縦幅を指定します。

処理概要

引数

(in)WSCushort height	項目の縦幅

復帰値 なし。

注意事項 一度、updateList() 関数を呼び出して、リストの表示を更新してください。

サンプル

```
newlist_000->setLabelHeight(25);
newlist_000->updateList();
```

22.8.2 delAll 関数の説明

書式 void delAll()

機能 表示項目を全て削除します。

処理概要

引数 なし。

復帰値 なし。

注意事項 削除した後、項目の追加等が終ったら、updateList() 関数で一度、表示を更新してください。

サンプル

```
newlist_000->delAll();
newlist_000->addItem("item1");
newlist_000->addItem("item2");
   :
newlist_000->updateList();
```

22.8.3 getLabel 関数の説明

書式 WSCbase* getLabel(long pos)

機能 実際に表示項目を表示している、内部のオブジェクトを取得します。

処理概要 項目の表示を行っているオブジェクトを返します。

引数

(in)long pos	項目の位置 (先頭=0 〜 N-1)

復帰値 項目を表示しているオブジェクト

注意事項 指定する位置は、先頭が0で、N 個ある場合は最後は、N-1 を指定します。また、-1 を指定すると N-1 を指定したものとみなされます。

サンプル

```
newlist_000->addItem("item1");
WSCbase* label = newlist_000->getLabel(-1); //末尾の項目のラベルを取得
label->setProperty(WSNforeColor,"#ff0000"); //文字列の表示色を赤に設定
```

22.8.4 getNum 関数の説明

書式 long getNum()

機能 表示項目の数を取得します。

処理概要

引数 なし。
復帰値 なし。
注意事項 なし。
サンプル

```
long i;
long num = newlist_000->getNum();
WSCstring item;
for(i=0; i<num; i++){
  item = newlist_000->getItem(i);
  printf("pos=%d item=%s\n",i,(char*)item);
}
```

22.8.5 getItem 関数の説明

書式 WSCstring getItem(long pos)
機能 指定した位置の項目の文字列を取得します。
処理概要
引数

(in)long pos	項目の位置 (先頭=0 〜 N-1)

復帰値 表示文字列
注意事項 返される文字列を char* で受け止めてはいけません。必ず、WSCstring クラスで受け、それから char* を取り出してください。
サンプル getNum() を参照してください。

22.8.6 addItem 関数の説明

書式 void addItem(char* item,long pos = -1)
機能 指定した位置に項目の文字列を挿入します。
処理概要 指定した位置 (先頭=1〜 N-1、-1=末尾) に与えられた文字列を表示項目に加えます。
引数

(in)char*	表示文字列
(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)

復帰値 なし。
注意事項 位置を指定しない場合、-1、すなわち、項目をリストの末尾に追加するものとみなします。なお、表示項目を変更した場合は、最後に一度、updateList() 関数を呼び出して、リストの表示を更新してください。
サンプル

```
newlist_000->delAll();
newlist_000->addItem("item0"); //末尾に追加
newlist_000->addItem("item1"); //末尾に追加
newlist_000->addItem("item2",0); //先頭に追加
newlist_000->addItem("item3",1); //先頭の次に追加
newlist_000->updateList();
```

結果：

```
item2
item3
item0
item1
```

22.8.7 replaceItem 関数の説明

書式 void replaceItem(char* item,long pos)
機能 指定した位置に項目の文字列を置き換えます。
処理概要 指定した位置 (先頭=1〜 N-1、-1=末尾) を与えられた文字列で置き換えます。
引数

(in)char*	表示文字列
(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)

復帰値 なし。
注意事項 指定する位置は、先頭が0で、N 個ある場合は最後は、N-1 を指定します。また、-1 を指定すると N-1 を指定したものとみなされます。なお、表示項目を変更した場合は、最後に一度、updateList() 関数を呼び出して、リストの表示を更新してください。

サンプル

```
newlist_000->delAll();
newlist_000->addItem("item1"); //末尾に追加
newlist_000->addItem("item2"); //末尾に追加
newlist_000->replaceItem("item0",-1); //末尾のものを置換
newlist_000->updateList();
```

結果：

```
item1
item2
```

22.8.8 delPos 関数の説明

書式 void delPos(long pos);

機能 指定した位置の項目を削除します。

処理概要 指定した位置 (先頭=1〜 N-1、-1=末尾) を表示項目リストから削除します。

引数

(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)

復帰値 なし。

注意事項 指定する位置は、先頭が0で、N 個ある場合は最後は、N-1 を指定します。また、-1 を指定すると N-1 を指定したものとみなされます。

サンプル

```
newlist_000->delAll(); //クリア
newlist_000->addItem("item0"); //末尾に追加
newlist_000->addItem("item1"); //末尾に追加
newlist_000->addItem("item2"); //末尾に追加
newlist_000->delPos(1);
newlist_000->updateList();
```

結果

```
item0
item2
```

22.8.9 getSelectedPos 関数の説明

書式 long getSelectedPos();

機能 選択された項目の位置を取得します。

処理概要 マウスにより選択状態になった項目の位置を取得します。

引数 なし。

復帰値 項目位置。(先頭=0 〜 N-1)

注意事項

サンプル

```
long pos = newlist_000->getSelectedPos();
printf("selected item = %d\n",pos);
```

22.8.10 setSelectPos 関数の説明

書式 void setSelectPos(long pos);

機能 指定した位置の項目を選択状態にします。

処理概要 指定した位置 (先頭=1〜 N-1、-1=末尾) を選択状態にします。

引数

(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)

復帰値 なし。

注意事項

サンプル

```
newlist_000->delAll();
newlist_000->addItem("item0");
newlist_000->addItem("item1");
newlist_000->addItem("item2");
newlist_000->setSelectedPos(1); // item1 を選択状態に設定
```

22.8.11 getSelectedItem 関数の説明

書式 WSCstring getSelectedItem();
機能 選択された項目の文字列を返します。
処理概要
引数 なし。
復帰値 選択状態になっている項目の文字列を返します。
注意事項 返された文字列は、開放してはいけません。また、複数選択されている場合は、最後に選択されたものが返されます。選択されたもののすべてを取得したい場合は、getSelectedLabels() 関数を使用してください。
サンプル

```
WSCstring item;
item = newlist_000->setSelectedItem();
printf("selected item = %s",(char*)item);
```

22.8.12 setTopPos 関数の説明

書式 void setTopPos(long pos);
機能 指定した位置の項目がリストの一番上に表示されるように表示位置をスクロールします。
処理概要
引数

(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)

復帰値 なし。
注意事項
サンプル

```
newlist_000->setTopPos(0); //先頭 1 行目を再上段に来るようスクロール
newlist_000->setTopPos(10); //先頭 11 行目を再上段に来るようスクロール
```

22.8.13 setBottomPos 関数の説明

書式 void setBottomPos(long pos);
機能 指定した位置の項目がリストの一番下に表示されるように表示位置をスクロールします。
処理概要
引数

(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)

復帰値 なし。
注意事項
サンプル

```
newlist_000->setBottomPos(100); //先頭 101 行目を再下段に来るようスクロール
```

22.8.14 updateList 関数の説明

書式 void updateList();
機能 表示項目に加えられた変更を表示に反映します。
処理概要
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル addItem() を参照してください。

22.8.15 setLabelClass 関数の説明

書式 void setLabelClass(char* class_name);
機能 表示項目に使用されるオブジェクトのクラスを指定します。
処理概要 指定されたクラスで、表示項目を表示します。デフォルトは、 WSCvlabel クラスです。
引数

(in)char* class_name	クラス名称を指定

復帰値 なし。
注意事項 クラスを指定する場合、プロパティ WSNlabelString を持ったものを指定してください。リストクラスは、表示すべき文字列を WSNlabelString プロパティに格納します。
サンプル

```
newlist_000->setLabelClass("WSCvklabel"); //クラス名を設定
```

22.8.16 getSelectItemChanged 関数の説明

書式 WSCbool getSelectItemChanged()

機能 項目の選択状態が変化したかどうかを返します。

処理概要 マウス等により選択状態でない項目が選択状態になったりして、項目の選択状態が変化した場合、返り値が True となります。

引数 なし。

復帰値 True = 変化あり、False = 変化なし。

注意事項 getSelectItemChanged() 関数を呼び出すに先だって、setSelectItemChanged() 関数で、フラグを False にしておく必要があります。setSelectItemChanged() でフラグを落した時点から、選択状態に変化があった場合、getSelectedItemChanged() の返り値が True となります。

サンプル

```
WSCbool changed = newlist_000->getSelectItemChanged();
if (changed == False){
  //選択状態に変化なし。
}else{
  //選択状態に変化あり。
}
```

22.8.17 setSelectItemChanged 関数の説明

書式 void setSelectItemChanged(WSCbool fl);

機能 選択状態フラグに値をセットします。

処理概要

引数

(in)WSCbool fl	True = 選択状態変更あり、False = なし

復帰値 なし。

注意事項 この関数は、状態フラグの値を設定するだけです。getSelectItemChanged() 関数を呼び出すのに使用します。

サンプル

```
newlist_000->setSelectItemChanged(True); //選択状態変更フラグをセットします。
```

22.8.18 getLabels 関数の説明

書式 WSClistData* getLabels();

機能 項目を表示しているオブジェクトのリストを返します。

処理概要

引数 なし。

復帰値 オブジェクトのリスト

注意事項 返されたリストを開放してはいけません。getChildren() 関数の子リストと同じように、リストの各要素を WSCbase* キャストしてオブジェクトにアクセスします。

サンプル

```
WSClistData* labels = newlist_000->getLabels();
long i;
long num = labels->getNum();
for(i=0; i<num; i++){
  WSCbase* label = (WSCbase*)labels->getData(i);
  //label に対して処理。
}
```

22.8.19 getSelectedLabels 関数の説明

書式 void getSelectedLabels(WSClistData & list);

機能 与えられたリストに、選択状態にある項目表示オブジェクトを格納します。

処理概要

引数

(out)WSClistData & list	オブジェクトを格納するリスト

復帰値 なし。

注意事項 返されたリストの各要素を WSCbase* キャストしてオブジェクトにアクセスします。

サンプル

```
WSClistData selected_labels;
newlist_000->getSelectedLabels(selected_labels);
long i;
long num = selected_labels.getNum();
for(i=0; i<num; i++){
  WSCbase* label = (WSCbase*)selected_labels［i］;
  //label に対して処理。
}
```

22.8.20 setItemVisible 関数の説明

書式 long setItemVisible(long pos,WSCbool fl);

機能 指定された位置の項目の表示状態を指定します。

処理概要

引数

(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)
(in)WSCbool fl	True = 表示、False = 非表示

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 項目を削除せずに、非表示としたい場合等に使用します。

サンプル

```
newlist_000->setItemVisible(0,False); //先頭の項目を不可視化。
```

22.8.21 setItemValue 関数の説明

書式 long setItemValue(long pos,long kind,long bal);

機能 指定された位置の項目の項目属性に値を指定します。

処理概要 項目属性には、次の様なものが存在します。

WS_OPEN	ツリー表示モード時の項目の開閉状態 (0=閉、1=開)
WS_ABSOLUTE_OPEN	ツリー表示モード時の項目の開閉状態 (0=閉、1=開)
WS_INDENT_LEVEL	ツリー表示モード時の項目の階層レベル (0=最上位,1,2,...)
WS_ICON	アイコン用画像ファイルのファイル名 (char*)

WS_ABSOLUTE_OPEN は、深い階層にある表示項目を開いた状態にした場合、上位の階層を含めて開いた状態にします。
アイコンを指定する場合は、アイコン表示プロパティを設定しなければなりません。

引数

(in)long pos	項目の位置 (先頭=0 〜 N-1、-1=末尾)
(in)long kind	項目属性
(in)long val	設定したい属性値

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
//ツリー表示の場合
newlist_000->delAll();
newlist_000->addItem("item1");
newlist_000->addItem("item2");
newlist_000->addItem("item3");
newlist_000->addItem("item4");
newlist_000->addItem("item5");
newlist_000->setItemValue(0,WS_INDENT_LEVEL,0);
newlist_000->setItemValue(1,WS_INDENT_LEVEL,1);
newlist_000->setItemValue(2,WS_INDENT_LEVEL,2);
newlist_000->setItemValue(3,WS_INDENT_LEVEL,1);
newlist_000->setItemValue(4,WS_INDENT_LEVEL,2);
newlist_000->setItemValue(0,WS_OPEN,1);
newlist_000->setItemValue(1,WS_OPEN,1);
newlist_000->setItemValue(2,WS_OPEN,1);
newlist_000->setItemValue(3,WS_OPEN,0);
```

結果:

```
---item1
```

```
+--item2
|   +--item3
+--item4
  ( +--item5 ) 不可視
```

22.8.22 getSortPos() 関数の説明

書式 long getSortPos();
機能 押されたソートボタンの番号を返します。
処理概要 詳細表示モードの時の、ソートボタンの左から、0,1,2,... が返されます。
引数 なし。
復帰値 ソートボタンの番号
注意事項
サンプル

```
//詳細表示、タイトルバー領域のソートボタンの番号の取得
long pos = newlist_000->getSortPos();
```

22.8.23 getTopPos() 関数の説明

書式 long getTopPos();
機能 現在表示中の最上段に位置する項目の番号を返します。
処理概要 現在表示中の最上段に位置する項目の番号を 0,1,2,... で返します。
引数 なし。
復帰値 最上段の項目の番号
注意事項
サンプル

```
//現在の最上段の項目位置の取得
long pos = newlist_000->getTopPos();
```

22.8.24 getBottomPos() 関数の説明

書式 long getBottomPos();
機能 現在表示中の最下段に位置する項目の番号を返します。
処理概要 現在表示中の最下段に位置する項目の番号を 0,1,2,... で返します。
引数 なし。
復帰値 最下段の項目の番号
注意事項
サンプル

```
//現在の最下段の項目位置の取得
long pos = newlist_000->getBottomPos();
```

22.8.25 setEnableActivate() 関数の説明

書式 void setEnableActivate(WSCbool fl);
機能 項目選択時に、ACTIVATE トリガを発生するかどうか設定します。
処理概要 True が設定された場合、項目選択時に、ACTIVATE トリガを発生させます。
引数

(in)WSCbool fl	トリガ発生指定

復帰値 なし。
注意事項
サンプル

```
//項目選択時に ACTIVATE を発生させるようにします。
newlist_000->setEnableActivate(True);
```

22.8.26 onSort() 関数の説明

書式 void onSort();

機能 リストタイプ属性が詳細表示の場合表示されるソートボタンが押された場合、onSort( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_SORT) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、データのソートに関するイベント処理を行うことができます。getSortPos() 関数を用いて、どのソートボタンが押されたかを確認し、データのソートを行います。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onSort(){
  //ソートボタンがおされた場合に呼び出されます。
  //押されたソートボタン番号の取得
  long btn = getSortPos();
  //押下されたボタン毎に処理を行います。

  //処理を派生元クラスに引き継ぎます。
  old_class::onSort();
}
```

22.8.27 onItemSelected() 関数の説明

書式 void onItemSelected();

機能 項目が選択状態になった場合、onItemSelected( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_ITEM_SELECTED) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、データの選択に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onItemSelected(){
  //項目が選択された場合に呼び出されます。
  long pos = getSelectedPos();

  //処理を派生元クラスに引き継ぎます。
  old_class::onItemSelected();
}
```

22.8.28 onInputFixed() 関数の説明

書式 void onInputFixed(WSCstring str,long pos,long column);

機能 項目にキーボード入力され、確定状態になった場合、onInputFixed( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_INPUT_FIXED) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、文字列の入力に関するイベント処理を行うことができます。

引数

(in)WSCstring str	入力された文字列
(in)long pos	入力のあった項目番号
(in)long column	詳細表示の場合のコラム位置

復帰値 なし。

注意事項

サンプル

```
void new_class::onInputFixed(){
  //項目入力が確定された場合に呼び出されます。
  //入力文字列の取得
  WSCstring str;
  str = getInputString();

  //処理を派生元クラスに引き継ぎます。
  old_class::onInputFixed();
}
```

22.8.29 cancelInput() 関数の説明

書式 void cancelInput();
機能 現在入力中のキーボード入力をキャンセルします。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
//現在、入力されている文字列を破棄します。
newlist_000->cancelInput();
```

22.8.30 getInputString() 関数の説明

書式 void getInputString();
機能 現在入力中のキーボード入力を取得します。
処理概要
引数 なし。
復帰値 現在の入力中の文字列
注意事項
サンプル onInputFixed() を参照してください。

22.8.31 onKey() 関数の説明

書式 void onKey(WSDkeyboard* keyboard,WSCbool keydown);
機能 項目にキーボードが押下または離された場合に onKey( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_KEY_PRESS / WSEV_KEY_RELEASE によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、キーボード入力に関するイベント処理を行うことができます。
引数

(in)WSDkeyboard* keyboard	キーボードインスタンス
(in)WSCbool keydown	True = Press、False = Release

復帰値 なし。
注意事項
サンプル

```
void new_class::onKey(WSDkeyboard* keyboard,WSCbool keydown){
  //キー入力された場合に呼び出されます。
  if (keydown != False){   //キー押下
    //キーの取得
    long key = keyboard->getKey();
    //入力文字列の取得
    WSCstring str = keyboard->getText();
  }

  //処理を派生元クラスに引き継ぎます。
  old_class::onKey(keyboard,keydown);
}
```


23 WSClistData

23.1 継承元

次のオブジェクトを継承しています。

23.2 概要

汎用リストデータ構造体です。メモリ管理を内部で自動的に行いますので、簡単にデータの追加等が行えます。

23.3 機能

- 自由なデータのリスト化
- リストのメモリ管理

23.4 注意事項

23.5 関数一覧

- WSClistData(); :WSClistData
- WSClistData(long); :WSClistData
- WSClistData(WSClistData &); :WSClistData
- long add(void* data,long pos=-1); :WSClistData
- void setData(long pos,void* data); :WSClistData
- void* getData(long pos); :WSClistData
- long getNum(); :WSClistData
- void** getBuf(); :WSClistData
- long del(void*); :WSClistData
- long delPos(long pos); :WSClistData
- void clear(); :WSClistData
- WSClistData & operator = (WSClistData &); :WSClistData
- void* & operator [] (long pos); :WSClistData
- void* getTopData(); :WSClistData
- void* getBottomData(); :WSClistData

23.6 関数仕様

23.6.1 WSClistData 関数の説明

書式 WSClistData()

機能 リストデータ構造体のコンストラクタです。リストデータのインスタンスを一つ作成します。

処理概要

引数 なし。

復帰値 リストデータのインスタンスへのポインタ

注意事項

サンプル

```
WSClistData  listdata;
long data1 = 100;
long data2 = 200;

//データを記憶させます。
listdata.add((void*)data1);
listdata.add((void*)data2);

//データを取り出します。
long d1 = (long)listdata [0] ;
long d2 = (long)listdata [1] ;

//指定した位置のデータを置き換えます。
long data3 = 300;
long data4 = 400;
listdata.setData(0,(void*)data3);
listdata.setData(1,(void*)data4);

//getData でデータの取り出しを行うと、次のようになります。
long d1 = (long)listdata.getData(0);
long d2 = (long)listdata.getData(1);

//データの数を取得するには、次のようにします。
long num = listdata.getNum();
```

```
//指定したデータを消すには、次のようにします。
listdata.del((void*)400);

//指定した位置のデータを消すには、次のようにします。
listdata.delPos(0);

//データ一括でを消すには、次のようにします。
listdata.clear();
```

23.6.2 WSClistData 関数の説明

書式 WSClistData(long segment_size)

機能 リストデータ構造体のコンストラクタです。セグメントサイズを指定して、リストデータのインスタンスを一つ作成します。セグメントサイズは、リスト構造体の内部で利用される領域を拡張する単位です。一度にたくさんのデータを保持する場合、データの出し入れが多い場合には、セグメントサイズを大きくすると、パフォーマンスが良くなります。デフォルトは、16 です。

処理概要

引数

(in)long segment_size	セグメントサイズ

復帰値 リストデータのインスタンスへのポインタ

注意事項 セグメントサイズにマイナスの値、0、256 以上を指定すると、256 として扱われます。

サンプル

```
//データバッファサイズ 64 としてリストを作成します。
//64 サイズ単位にリストが拡張されます。
WSClistData  indexdata(64);
```

23.6.3 add 関数の説明

書式 long add(void* data,long pos = -1)

機能 リストデータにデータを一つ追加します。

処理概要 データの挿入位置 (pos) を指定しない場合、または、-1 を指定した場合は、データの末尾に追加されます。また、位置を指定した場合には、その位置にデータが挿入され、それ以降のデータは、一つ後ろにずらされます。先頭に追加する場合は、0 を指定します。

引数

(in)void* data	追加したいデータ
(in)long pos	挿入位置 (指定しない場合は、末尾)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 指定した位置が、保持しているデータ個数よりも大きい場合は、リストの末尾に追加されます。

サンプル WSClistData() を参照してください。

23.6.4 setData 関数の説明

書式 void setData(long pos,void* data)

機能 指定した位置のデータを置き換えます。

処理概要

引数

(in)long pos	入れ換えたいデータの位置 (先頭は 0)
(in)void* data	データ

復帰値 なし。

注意事項 指定した位置にデータが無い場合は、無視されます。

サンプル WSClistData() を参照してください。

23.6.5 getData 関数の説明

書式 void* getData(long pos)
機能 指定した位置のデータを取得します。
処理概要
引数

(in)long pos	データの位置 (先頭は 0)

復帰値 保持しているデータ
注意事項 指定した位置にデータが存在しない場合、NULL が返されます。
サンプル WSClistData() を参照してください。

23.6.6 getNum 関数の説明

書式 long getNum()
機能 リストデータが保持するデータの個数を返します。
処理概要
引数 なし。
復帰値 保持しているデータの個数
注意事項
サンプル WSClistData() を参照してください。

23.6.7 getBuf 関数の説明

書式 void** getBuf()
機能 リストデータが保持するデータを保持するデータ領域を返します。
処理概要
引数 なし。
復帰値 データ領域
注意事項 返された領域を開放してはいけません。
サンプル WSClistData() を参照してください。

```
WSClistData  listdata;
long data1 = 100;
long data2 = 200;
long data3 = 300;

//データを記憶させます。
listdata.add((void*)data1);
listdata.add((void*)data2);
listdata.add((void*)data3);

//内部のバッファを取得します。
void** ptr =  listdata.getBuf();

//バッファからデータを取り出してみます。
long d1 = *(long*)(ptr[0]); //data1 の 100
long d2 = *(long*)(ptr[1]); //data2 の 200
long d3 = *(long*)(ptr[2]); //data3 の 300
```

23.6.8 del 関数の説明

書式 long del(void*)
機能 指定したデータをリストから削除します。
処理概要

引数

(in)void* data	削除したいデータ

復帰値 削除成功の場合は０、存在しない場合は－１。
注意事項 複数存在する場合は、先頭に近いものから順に削除されていきます。
サンプル

```
WSClistData  listdata;
long data1 = 100;
long data2 = 200;
long data3 = 300;

//データを記憶させます。
listdata.add((void*)data1);
listdata.add((void*)data2);
listdata.add((void*)data3);

//指定したデータを消すには、次のようにします。
listdata.del((void*)200);

//指定したデータを消すには、次のようにします。
listdata.delPos(1);

//データ一括でを消すには、次のようにします。
listdata.clear();
```

23.6.9 delPos 関数の説明

書式 long delPos(long pos)
機能 指定した位置のデータをリストから削除します。
処理概要 位置に -1 を指定すると、末尾のデータを削除します。
引数

(in)long pos	削除したいデータの位置 (先頭は０)

復帰値 削除成功の場合は０、存在しない場合は－１。
注意事項
サンプル del() を参照してください。

23.6.10 clear 関数の説明

書式 void clear()
機能 保持しているデータをリセットし、データを持たない状態に初期化します。
処理概要 データ個数は０となります。
引数 なし。
復帰値 なし。
注意事項
サンプル del() を参照してください。

23.6.11 = オペレータの説明

書式 WSClistData & operator = (WSClistData &)
機能 リストデータをコピーします。
処理概要
引数

(in)WSClistData & src	コピーしたいリストデータ

復帰値
注意事項

サンプル

```
WSClistData  listdata;
long data1 = 100;
long data2 = 200;
long data3 = 300;

//リストをコピーするには、次のようにします。
WSClistData  listdata2;
listdata2 = listdata;
```

23.6.12 [] オペレータの説明

書式 void* & operator [] (long pos)
機能 リストデータを配列の様に扱えるようにします。
処理概要
引数

(in)long pos	データの位置 (先頭は0)

復帰値 データ
注意事項 次の様に使用します。
サンプル WSClistData() を参照してください。

23.6.13 getTopData 関数の説明

書式 void getTopData()
機能 リストの先頭のデータを返します。
処理概要 getData(0) と同じです。
引数 なし。
復帰値 先頭のデータ
注意事項 データが存在しない場合、NULL が返されます。
サンプル

```
WSClistData  listdata;
long data1 = 100;
long data2 = 200;
long data3 = 300;

//リストの先頭のデータを取得するには、次のようにします。
long tval = (void*)listdata.getTopData();
```

23.6.14 getBottomData 関数の説明

書式 void getBottomData()
機能 リストの最後尾のデータを返します。
処理概要 getData(-1) と同じです。
引数 なし。
復帰値 最後尾のデータ
注意事項 データが存在しない場合、NULL が返されます。
サンプル

```
WSClistData  listdata;
long data1 = 100;
long data2 = 200;
long data3 = 300;

//リストの末尾のデータを取得するには、次のようにします。
long bval = (void*)listdata.getBottomData();
```

24 WSClocaleSet

24.1 継承元

次のオブジェクトを継承しています。

24.2 概要

言語情報を管理します。エンコーディングやロケールにする情報を取得したい場合にこのクラスを利用します。

24.3 機能

- エンコーディングやロケールにする情報の管理

24.4 注意事項

24.5 関数一覧

- WSClocaleSet* WSGIappLocaleSet(); :WSClocaleSet
- char* getDefaultLocaleName(); :WSClocaleSet
- long getDefaultEncoding(); :WSClocaleSet
- void setDefaultEncoding(long int); :WSClocaleSet
- long getSystemLocaleEncoding(); :WSClocaleSet
- char* getLocaleString(char* locale,char* index,long encode); :WSClocaleSet

24.6 関数仕様

24.6.1 WSGIappLocaleSet 関数の説明

書式 WSClocaleSet* WSGIappLocaleSet()

機能 アプリケーションにひとつ存在するエンコード情報管理クラスのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 エンコード情報管理クラスインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル

```
//プロセスに1つ存在するエンコード情報管理クラスインスタンスを取得します。
WSClocaleSet* localeset = WSGIappLocaleSet();
```

24.6.2 getDefaultLocaleName 関数の説明

書式 char* getDefaultLocaleName()

機能 デフォルトのロケール名を取得します。

処理概要

引数 なし。

復帰値 ロケール名

注意事項 返されたロケール名を開放してはなりません。

サンプル

```
//ロケール名を取得するには次のようにします。
char* lname = WSGIappLocaleSet()->getDefaultLocaleName();
```

24.6.3 setDefaultEncoding 関数の説明

書式 void setDefaultEncoding(long encoding)

機能 デフォルトのエンコーディングを設定します。

処理概要

引数

(in)long encoding	エンコーディング

エンコーディングには次のものを指定します。

エンコーディング	意味
WS_EN_DEFAULT	現在の設定を指定 (省略時の値)
WS_EN_LOCALE	現在の LANG 環境変数の設定を指定
WS_EN_NONE	設定を行わない
WS_EN_ISO8859_1	ISO8859(1) を指定
WS_EN_ISO8859_2	ISO8859(2) を指定
WS_EN_ISO8859_3	ISO8859(3) を指定
WS_EN_ISO8859_4	ISO8859(4) を指定
WS_EN_ISO8859_5	ISO8859(5) を指定
WS_EN_ISO8859_6	ISO8859(6) を指定
WS_EN_ISO8859_7	ISO8859(7) を指定
WS_EN_ISO8859_8	ISO8859(8) を指定
WS_EN_ISO8859_9	ISO8859(9) を指定
WS_EN_ISO8859_10	ISO8859(10) を指定
WS_EN_ISO8859_11	ISO8859(11) を指定
WS_EN_ISO8859_12	ISO8859(12) を指定
WS_EN_ISO8859_13	ISO8859(13) を指定
WS_EN_ISO8859_14	ISO8859(14) を指定
WS_EN_ISO8859_15	ISO8859(15) を指定
WS_EN_UTF8	UTF8 を指定
WS_EN_KOI8R	KOI8R を指定
WS_EN_EUCJP	EUCJP を指定
WS_EN_SJIS	SJIS を指定
WS_EN_EUCKR	EUCKR を指定
WS_EN_EUCCN	EUCCN を指定
WS_EN_BIG5	BIG5 を指定

復帰値 なし。

注意事項

サンプル

```
//デフォルトのエンコーディングを EUCJP に設定するには、次のようにします。
WSGIappLocaleSet()->setDefaultEncoding(WS_EN_EUCJP);
```

24.6.4 getDefaultEncoding 関数の説明

書式 long getDefaultEncoding()

機能 デフォルトのエンコーディングを取得します。

処理概要

引数 なし。

復帰値 エンコーディング
setDefaultEncoding 関数の引き数を参照ください。

注意事項

サンプル

```
//デフォルトのエンコーディングを取得するには、次のようにします。
long encoding = WSGIappLocaleSet()->getDefaultEncoding();
```

24.6.5 getSystemLocaleEncoding 関数の説明

書式 long getSystemLocaleEncoding()

機能 システム入出力で使用されるエンコーディングを取得します。

処理概要

引数 なし。

復帰値 エンコーディング
setDefaultEncoding 関数の引き数を参照ください。

注意事項

サンプル
```
//システム入出力のエンコーディングを取得するには、次のようにします。
long encoding = WSGIappLocaleSet()->getSystemLocaleEncoding();
```

24.6.6 getLocaleString 関数の説明

書式 char* getLocaleString(char* locale,char* index,long encode)

機能 指定したロケールの言語別の文字列を取得します。

処理概要 あらかじめ言語別にインデックス文字列に対して言語別に文字列を登録されたもののうち、指定されたインデックス文字列に対応した文字列を取得します。

引数

(in)char* locale	ロケール名
(in)char* index	インデックス文字列
(in)long encoding	取得する文字列のエンコーディング

setDefaultEncoding 関数の引き数を参照ください。

復帰値 言語別の文字列

注意事項 返された文字列を解放してはなりません。

サンプル
```
//あらかじめ登録されたロケール変換文字列を取得するには、次のようにします。
char* str = WSGIappLocaleSet()->getLocaleString("EUCJP","test string",WS_EN_EUCJP);
```

25 WSCmainWindow

25.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow

25.2 概要

一つの独立したウィンドウを提供します。通常、アプリケーションウィンドウのベースとして使われます。WSCwindow のプロパティ WSNexit を True に再定義したものです。その他は WSCwindow と機能的に変わりはありません。

25.3 機能

- 独立したウィンドウの提供
- ウィンドウマネージャの装飾
- 他のオブジェクトの配置と管理機能
- 背景ピクスマップの表示機能

25.4 注意事項

25.5 プロパティー覧

クラス WSCmainWindow のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100

unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	0
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	1
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNindicatorOn	インジケータ使用	1
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNmargin	マージン	2
unsigned short	WSNmarginLeft	左マージン	2
WSCbool	WSNemboss	エンボス	0
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNbarThickness	バー幅	16
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

25.6 トリガー一覧

WSCmainWindow クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

25.7 関数一覧

25.8 関数仕様

26 WSCmenuArea

26.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

26.2 概要

矩形のウィンドウ領域で、プルダウンメニュー用の領域を提供します。親ウィンドウのサイズに連動して、幅が自動的に追従します。プロパティ WSNx 、WSNy は、左上隅に固定されます。

26.3 機能

・ウィンドウに連動したサイズ変更機能

26.4 注意事項

26.5 プロパティー一覧

クラス WSCmenuArea のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	398
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF5
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1

	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

26.6 トリガー一覧

WSCmenuArea クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

26.7 関数一覧

26.8 関数仕様

27 WSCmessageDialog

27.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow WSCbaseDialog

27.2 概要

メッセージを表示するダイアログを提供します。メッセージ文字列を、WSNlabelString プロパティに指定して、popup() を実行します。

27.3 機能

- メッセージの表示

27.4 注意事項

27.5 プロパティー覧

クラス WSCmessageDialog のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNlabelString	表示文字列	
short	WSNworkBackColor	作業領域背景色	DEF10
WSCbool	WSNok	OK ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNno	NO ボタン	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	
char*	WSNokString	OK ボタン表示文字列	OK
char*	WSNnoString	NO ボタン表示文字列	NO
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	100
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	

	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

27.6 トリガー一覧

WSCmessageDialog クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

27.7 関数一覧

27.8 関数仕様

28 WSCngbase

28.1 継承元

次のオブジェクトを継承しています。
WSCbase

28.2 概要

ウィンドウ資源を持たないオブジェクトの基本となるサブクラスで、このオブジェクトを基に派生すると実行時になにも表示しないオブジェクトを作成することができます。例えば、タイマー (WSCvtimer) のような、実行時に表示されたくないようなものに使われています。

28.3 機能

・実行時になにも表示しないクラスの基本機能の提供

28.4 注意事項

28.5 プロパティー覧

クラス WSCngbase のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

28.6 トリガー覧

WSCngbase クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

28.7 関数一覧

28.8 関数仕様

29 WSCnwbase

29.1 継承元

次のオブジェクトを継承しています。
WSCbase

29.2 概要

ウィンドウ資源を持たないオブジェクトの基本となるサブクラスです。このオブジェクトを元にするとウィンドウを持ちませんが、ウィンドウが存在するときと同じように振舞うことができます。

29.3 機能

- ブリンク機構の提供
- マウスイベントのマネージメント

29.4 注意事項

29.5 プロパティ一覧

クラス WSCnwbase のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

29.6 トリガー一覧

WSCnwbase クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

29.7 関数一覧

- void setBlinkFore(WSCbool) :WSCnwbase
- void setOutSideMousePress(WSCbool) :WSCnwbase
- void setOutSideMouseMove(WSCbool) :WSCnwbase
- void setOutSideMouseRelease(WSCbool) :WSCnwbase
- void setMouseGrabed(WSCbool) :WSCnwbase
- WSCbool getOutSideMousePress() :WSCnwbase
- WSCbool getOutSideMouseMove() :WSCnwbase
- WSCbool getOutSideMouseRelease() :WSCnwbase
- WSCbool getMouseGrabed() :WSCnwbase
- long getBlinkRate() :WSCnwbase
- WSCbool getBlinkRefresh() :WSCnwbase

29.8 関数仕様

29.8.1 setBlinkFore 関数の説明

書式 void setBlinkFore(WSCbool fl)

機能 ブリンクの点滅の裏表を設定します。

処理概要 True を指定すると表、False を指定すると裏に設定されます。

引数

(in)WSCbase fl	ブリンクの裏表の指定

復帰値 なし。

注意事項 他のオブジェクトと、ブリンクの裏表の同期を取るときなどに使用します。

サンプル

```
//呼び出されたインスタンスのブリンクの裏表を一致させるには、次のようにします。
newvlab_000->setBlinkFore(True);
newvlab_001->setBlinkFore(True);
newvlab_002->setBlinkFore(True);
newvlab_003->setBlinkFore(True);
```

29.8.2 setOutSideMousePress 関数の説明

書式 void setOutSideMousePress(WSCbool fl)

機能 親ウィンドウマウスの押下イベントを、自分の領域以外の場合でも通知するか否かを設定します。

処理概要

引数

(in)WSCbase fl	領域外のイベント通知の有無

復帰値 なし。

注意事項

サンプル

```
//インスタンスのエリアの外に出ても、親フォーム内のマウスボタン押下イベントを
//取得できるようにするには次のようにします。
newvlab_000->setOutSideMousePress();
```

29.8.3 setOutSideMouseMove 関数の説明

書式 void setOutSideMouseMove(WSCbool fl)
機能 親ウィンドウマウスの移動イベントを、自分の領域以外の場合でも通知するか否かを設定します。
処理概要
引数

(in)WSCbase fl	領域外のイベント通知の有無

復帰値 なし。
注意事項
サンプル

```
//インスタンスのエリアの外に出ても、親フォーム内のマウス移動イベントを
//取得できるようにするには次のようにします。
newvlab_000->setOutSideMouseMove();
```

29.8.4 setOutSideMouseRelease 関数の説明

書式 void setOutSideMouseRelease(WSCbool fl)
機能 親ウィンドウマウスのリリースイベントを、自分の領域以外の場合でも通知するか否かを設定します。
処理概要
引数

(in)WSCbase fl	領域外のイベント通知の有無

復帰値 なし。
注意事項
サンプル

```
//インスタンスのエリアの外に出ても、親フォーム内のマウスボタンの
//リリースイベントを取得できるようにするには次のようにします。
newvlab_000->setOutSideMouseRelease();
```

29.8.5 getOutSideMousePress 関数の説明

書式 WSCbool getOutSideMousePress()
機能 親ウィンドウマウスの押下イベントを、自分の領域以外の場合でも通知するか否かを取得します。
処理概要
引数 なし。
復帰値 True= 通知、False=非通知。
注意事項
サンプル

```
//エリア外マウスボタン押下イベントの取得設定状態を取得します。
WSCbool bool = newvlab_000->getOutSideMousePress();
```

29.8.6 getOutSideMouseMove 関数の説明

書式 WSCbool getOutSideMouseMove()
機能 親ウィンドウマウスの移動イベントを、自分の領域以外の場合でも通知するか否かを取得します。
処理概要
引数 なし。
復帰値 True= 通知、False=非通知。
注意事項
サンプル

```
//エリア外マウス移動イベントの取得設定状態を取得します。
WSCbool bool = newvlab_000->getOutSideMouseMove();
```

29.8.7 getOutSideMouseRelease 関数の説明

書式 WSCbool getOutSideMouseMove()
機能 親ウィンドウマウスのリリースイベントを、自分の領域以外の場合でも通知するか否かを取得します。
処理概要
引数 なし。
復帰値 True= 通知、False=非通知。
注意事項
サンプル

```
//エリア外マウスボタンのリリースイベントの取得設定状態を取得します。
WSCbool bool = newvlab_000->getOutSideMouseRelease();
```

29.8.8 getBlinkRate 関数の説明

書式 long getBlinkRate()
機能 ブリンクの時間間隔を取得します。
処理概要 WSNblinkRate プロパティの値を返します。
引数 なし。
復帰値 ブリンクレート

WS250MS	250 ms 後
WS500MS	500 ms 後
WS750MS	750 ms 後
WS1000MS	1000 ms 後
WS1250MS	1250 ms 後
WS1500MS	1500 ms 後
WS1750MS	1750 ms 後
WS2000MS	2000 ms 後
WS2250MS	2250 ms 後
WS2500MS	2500 ms 後
WS2750MS	2750 ms 後
中略	中略
WS19750MS	19750 ms 後
WS20000MS	20000 ms 後

注意事項
サンプル

```
//ブリンクレートを取得します。
long rate = newvlab_000->getBlinkRate();
```

29.8.9 getBlinkRefresh 関数の説明

書式 WSCbool getBlinkRefresh()
機能 ブリンクリフレッシュ属性を返します。
処理概要 ブリンクリフレッシュ属性が True の場合、ブリンク時に再描画イベントを発生させます。他のインスタンスが重なって表示されている場合、ブリンクによって他のインスタンスの表示が欠けてしまうことを防ぎます。WSNblinkRefreshing プロパティの値を返します。
引数 なし。
復帰値 ブリンクリフレッシュ属性
注意事項
サンプル

```
//ブリンクリフレッシュ属性を取得します。
WSCbool bool = newvlab_000->getBlinkRefresh();
```

30 WSCoption

30.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel

30.2 概要

ボタン型のメニューによる値の選択を行います。

30.3 機能

・メニュー形式の値の選択機能

30.4 注意事項

30.5 プロパティー覧

クラス WSCoption のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNmenuItems	メニュー項目	item1:1:ep1_name,item2:2:ep2_name,item3:3:ep3_na
long	WSNvalue	値	0
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1

	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

30.6 トリガー一覧

WSCoption クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT

30.7 関数一覧

- long setSelectValue(long value,WSCbool flag) :WSCoption
- long setItemSensitive(long no,WSCbool flag) :WSCoption
- long getValue( ) :WSCoption
- long getItems( ) :WSCoption
- WSCbool* getSelectStatus( ) :WSCoption
- void onActivate( ) :WSCoption
- void onValueChange(long) :WSCoption

30.8 関数仕様

30.8.1 setSelectValue 関数の説明

書式 long setSelectValue(long value,WSCbool flag)

機能 指定した値の項目の選択属性を指定します。

処理概要 指定された値のメニュー項目を探し、True ならば、その項目が選択可能に、False ならば、選択不可能に設定します。

引数

(in)long value	選択属性を指定したいメニューの値
(in)WSCbool fl	選択属性

復帰値 WS_NO_ERR = 正常、WS_ERR = 存在しない値が指定された場合。
注意事項
サンプル

```
//例えばメニューで設定されている値 100 に対応する項目を不選択とする場合、
//次のようにします。
newopti_000->setSelectValue(100,False);
```

30.8.2 setItemSensitive 関数の説明

書式 long setItemSensitive(long no,WSCbool flag)
機能 上から順に 0,1,2,... で指定された項目の選択属性を指定します。
処理概要 True ならば、その項目が選択可能に、False ならば、選択不可能に設定します。
引数

(in)long no	0,1,2,... で指定されるメニュー項目
(in)WSCbool fl	選択属性

復帰値 WS_NO_ERR = 正常、WS_ERR = 存在しない項目が指定された場合。
注意事項
サンプル

```
//例えば、先頭の項目を不選択とする場合、次のようにします。
newopti_000->setItemSensitive(0,False);
```

30.8.3 getValue 関数の説明

書式 long getValue()
機能 現在選択されている値を返します。
処理概要
引数 なし。
復帰値 現在選択されているメニューの値。
注意事項
サンプル

```
//現在選択されている値を取得します。
long val = newopti_000->getValue();
```

30.8.4 getItems 関数の説明

書式 long getItems()
機能 現在のメニュー項目数を取得します。
処理概要
引数 なし。
復帰値 メニュー項目数
注意事項
サンプル

```
//現在の各項目の選択状態を取得します。
WSCbool* statuslist = newopti_000->getSelectStatus();
long num = newopti_000->getItems();
long i;
for(i=0; i<num; i++){
  WSCbool status = statuslist[i];
  printf("menu%d status=%d\n",i,status);
}
```

30.8.5 getSelectStatus 関数の説明

書式 WSCbool* getSelectStatus()
機能 上から順に 0,1,2,... の順でで選択属性 (選択可能状態) の配列を返します。
処理概要
引数 なし。
復帰値 選択属性の配列
注意事項
サンプル getItems() を参照してください。

30.8.6 onActivate 関数の説明

書式 void onActivate()
機能 値が選択された場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる代わりに、この関数をオーバーライドすることでも、値の選択に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル

```
void new_class::onActivate(){
  //値が選択された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

30.8.7 onValueChange 関数の説明

書式 void onValueChange(long value)
機能 値が変化した場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_VALUE_CH) によるイベントプロシージャを用いる代わりに、この関数をオーバーライドすることでも、値の変化に関するイベント処理を行うことができます。
引数

(in)long value	新たな値

復帰値 なし。
注意事項
サンプル

```
void new_class::onValueChange(){
  //値が選択され、選択値が変化した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onValueChange();
}
```

31 WSCpopupMenu

31.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow WSCpulldownMenuPopup

31.2 概要

マウス押下により表示されるポップアップメニューです。メニューを表示させたいオブジェクトをポップアップメニューに登録するとメニューが表示されるようになります。

31.3 機能

- マウス押下によりポップアップメニューが表示されます。

31.4 注意事項

31.5 プロパティー覧

クラス WSCpopupMenu のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値

void*	WSNclient	クライアント	0
char*	WSNmenuItems	メニュー項目	
unsigned short	WSNmenuItemHeight	メニュー項目縦幅	30
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	0
short	WSNforeColor	表示色	DEF6
short	WSNbackColor	背景色	DEF5
short	WSNtopShadowColor	上影色	DEF8
short	WSNbottomShadowColor	下影色	DEF9
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	1
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNemboss	エンボス	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

31.6 トリガー一覧

WSCpopupMenu クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

31.7 関数一覧

- long registerClient(WSCbase*) :WSCpopupMenu
- long unregisterClient(WSCbase*) :WSCpopupMenu
- void onActivate() :WSCbaseDialog

31.8 関数仕様

31.8.1 registerClient 関数の説明

書式 long registerClient(WSCbase* client)

機能 ポップアップメニューを表示したいオブジェクトを指定します。

処理概要 引数に指定されたオブジェクト上で右マウスボタンが押された場合、そのマウスが押された場所にポップアップメニューが表示されるようになります。

引数

(in)WSCbase* client	メニューを表示させたいオブジェクト

復帰値 なし。

注意事項 ２度以上、同じメニューに対して登録しないでください。また、一度登録したオブジェクトを開放 (デストロイ) する場合、登録を抹消 (unregisterClient() 関数を参照) してから開放してください。

サンプル //初期化トリガでのイベントプロシージャの例です。

```
#include <WSCpopupMenu.h>
void menu1_proc(WSCbase* client){
  //メニュー１が選択されると起動されます。
}
void menu2_proc(WSCbase* client){
  //メニュー２が選択されると起動されます。
}
void menu3_proc(WSCbase* client){
  //メニュー３が選択されると起動されます。
}
void init_event_proc(WSCbase* client){
  //ポップアップメニューの生成
  WSCpopupMenu*  pmenu = new WSCpopupMenu(NULL,"menu");
  pmenu->initialize();
  //ポップアップのプロパティを設定します。
  pmenu->setPropertyV(WSNmenuItemHeight,(WSCushort)20);
  pmenu->setPropertyV(WSNshadowThickness,(WSCuchar)1);
  pmenu->setPropertyV(WSNbackColor,"gray85");
  pmenu->setPropertyV(WSNtopShadowColor,"gray95");
  pmenu->setPropertyV(WSNbottomShadowColor,"gray55");
  //ポップアップメニューのメニュー項目を指定します。
  pmenu->setPropertyV(WSNmenuItems,
       "メニュー１:menu1, メニュー２:menu2, メニュー３:menu3");

  WSCprocedure*
```

```
  pmop = new WSCprocedure("menu1",WSEV_NONE);
  pmop->setFunction(menu1_proc,"_menu1_proc");
  pmenu->addProcedure(pmop);

  pmop = new WSCprocedure("menu2",WSEV_NONE);
  pmop->setFunction(menu2_proc,"menu2_proc");
  pmenu->addProcedure(pmop);

  pmop = new WSCprocedure("menu3",WSEV_NONE);
  pmop->setFunction(menu3_work,"menu3_proc");
  pmenu->addProcedure(pmop);

  pmenu->registerClient(client);
  client->setUserData("popmenu",(void*)pmenu);
}
void delete_event_proc(WSCbase* client){
  WSCpopupMenu* pmenu = (WSCpopupMenu*)client->getUserData("popmenu");
  pmenu->unregisterClient(client);
  delete pmenu;
  client->setUserData("popmenu",(void*)NULL);
}
```

31.8.2 unregisterClient 関数の説明

書式 long unregisterClient(WSCbase*)
機能 一度登録されたオブジェクトの登録を抹消します。
処理概要
引数

(in)WSCbase* client	登録を抹消させたいオブジェクト

復帰値 なし。
注意事項 なし。
サンプル registerClient() を参照してください。

31.8.3 onActivate 関数の説明

書式 void onActivate()
機能 メニューが選択された場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、メニュー選択に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル

```
void new_class::onActivate(){
  //メニューが選択された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

32 WSCprform

32.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

32.2 概要

矩形のウィンドウ領域で、他のオブジェクトを内部に配置し、子として管理する機能を提供します。また、prdraw() 関数で矩形のウィンドウ領域のポストスクリプトファイル生成や、プリンタ出力等を行うことができます。

32.3 機能

- 上位のウィンドウとは独立した矩形の領域の提供
- 他のオブジェクトの配置と管理機能
- 枠の表示機能
- 背景ピクスマップの表示機能
- プリンタ出力、ポストスクリプトファイルの生成

32.4 注意事項

Windows 版の表示タイププロパティ(WSNtype) には、プロパティ値 2 (Printer 出力) があります。このプロパティの値を Printer に設定することで、直接プリンターに出力することができます。

32.5 プロパティー覧

クラス WSCprform のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNfileName	ファイル名	tmp.ps
short	WSNworkX	仮想領域X座標	10
short	WSNworkY	仮想領域Y座標	10
unsigned short	WSNworkWidth	仮想領域横幅	512
unsigned short	WSNworkHeight	仮想領域縦幅	724
unsigned char	WSNformat	フォーマット	0
	A4	(0)	
	A5	(1)	
	A3	(2)	
	B4	(3)	
	B5	(4)	
	Letter	(5)	
	Legal	(6)	
	Statement	(7)	
	Tabloid	(8)	
	Ledger	(9)	
	Folio	(10)	
	Quarto	(11)	
	10x14	(12)	
	Executive	(13)	
unsigned char	WSNtype	表示タイプ	0
	PostScript	(0)	
	EPS	(1)	
unsigned char	WSNorientation	表示方向	1
	横	WS_HORIZONTAL (0)	
	縦	WS_VERTICAL (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF11
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	

	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

32.6 トリガー一覧

WSCprform クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

32.7 関数一覧

- virtual long prdraw() :WSCprform

32.8 関数仕様

32.8.1 prdraw 関数の説明

書式 long prdraw()
機能 ポストスクリプトファイルの生成もしくは、印字出力を行います。
処理概要 子インスタンスを含めて描画イメージをポストスクリプトファイルに出力します。
引数 なし。
復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項 子インスタンスの内、WSCform などのウィンドウ資源を持つものの出力は行いません。
サンプル

```
//プリント出力を行います。
newprfo_000->prdraw();
```

33 WSCj3wform

33.1 継承元

次のオブジェクトを継承しています。
WSCform WSCbase

33.2 概要

J3W コンピュータグラフィックスライブラリを使用した、J3D ファイルの表示を行います。プロパティWSNfileName に J3W で作成した J3D ファイルを指定すると、そのグラフィックスの内容を表示します。

33.3 機能

- J3D ファイルの表示

33.4 注意事項

33.5 プロパティー覧

クラス WSCj3wform のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNfileName	ファイル名	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF5
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	

unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

33.6 トリガー一覧

WSCj3wform クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

33.7 関数一覧

33.8 関数仕様

34 WSCopenglForm

34.1 継承元

次のオブジェクトを継承しています。
WSCform WSCbase

34.2 概要

OpenGL ライブラリを使ったコンピュータグラフィックスを描画、表示するためのフォームです。OpenGL の API を使って、いろいろなコンピュータグラフィックスを表示することが出来ます。

34.3 機能

・OpenGL によるコンピュータグラフィックスの表示

34.4 注意事項

34.5 プロパティー覧

クラス WSCopenglForm のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	

WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

34.6 トリガー一覧

WSCopenglForm クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

34.7 関数一覧

34.8 関数仕様

35 WSCvssocket

35.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

35.2 概要

サーバ側のＴＣＰソケットクラスです。クライアント側ソケットクラスの接続を受付、ソケット通信を行います。プロパティ WSNport にバインドするポート番号を指定します。また、プロパティ WSNip は通常、指定する必要はありませんが、ホストマシンに複数のホストアドレスが存在し、そのうちのどれかに限定してバインドしたい場合、プロパティ WSNip にそのアドレスを指定し、アドレスを限定することができます。

クライアントからの接続があると、ACTIVATE イベントが発生します。通信は、その ACTIVATE イベントで起動するプロシージャ内で行います。

35.3 機能

- クライアント側ソケットの接続の受付
- ソケット通信

35.4 注意事項

35.5 プロパティー覧

クラス WSCvssocket のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNip	ＩＰ	
unsigned short	WSNport	ポート	9000
unsigned long	WSNtimeout	タイムアウト	60
WSCbool	WSNrunning	実行中	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

35.6 トリガー覧

WSCvssocket クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

35.7 関数一覧

- long read(WSCuchar* buffer,long size) :WSCvcsocket
- long write(WSCuchar* buffer,long size) :WSCvcsocket

35.8 関数仕様

35.8.1 read 関数の説明

書式 long read(WSCuchar* buffer,long size)

機能 ACTIVATE で起動されるイベントプロシージャ内で接続されたソケットからデータを読み込みます。

処理概要

引数

(in)WSCuchar* buffer	読み込みデータ領域
(in)long len	読み込みデータ領域長

復帰値 読み込んだデータ長 (バイト数)

注意事項 ACTIVATE イベントで起動されるプロシージャ以外で使用した場合、接続が確立されておらず、エラーとなります。

サンプル

```
char buffer［128］;
long ret = newvsso_000->read(buffer,128);
```

35.8.2 write 関数の説明

書式 long write(WSCuchar* buffer,long size)

機能 ACTIVATE で起動されるイベントプロシージャ内で接続されたソケットにデータを書き込みます。

処理概要

引数

(in)WSCuchar* buffer	書き込みデータ領域
(in)long len	書き込みデータ領域長

復帰値 書き込んだデータ長 (バイト数)

注意事項 ACTIVATE イベントで起動されるプロシージャ以外で使用した場合、接続が確立されておらず、エラーとなります。

サンプル

```
char buffer［128］;
strcpy(buffer,"send data...");
long ret = newvsso_000->write(buffer,128);
```

36 WSCvcsocket

36.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

36.2 概要

サーバ側ソケットクラスに接続を行い、ソケット通信を行うクライアント側ソケットクラスです。プロパティ WSNip に接続先ホストアドレス、WSNport に接続先ポート番号を指定します。

36.3 機能

- サーバ側ソケットへの接続
- ソケット通信

36.4 注意事項

36.5 プロパティ一覧

クラス WSCvcsocket のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNip	ＩＰ	
unsigned short	WSNport	ポート	9000
unsigned long	WSNtimeout	タイムアウト	60
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

36.6 トリガー一覧

WSCvcsocket クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

36.7 関数一覧

- long exec() :WSCvcsocket
- long read(WSCuchar* buffer,long size) :WSCvcsocket
- long write(WSCuchar* buffer,long size) :WSCvcsocket

36.8 関数仕様

36.8.1 exec 関数の説明

書式 long exec()

機能 プロパティWSNip 、WSNport に指定されたサーバ側ソケットに接続を行い、接続が成功した場合に、ACTIVATE イベントを発生させます。実際の通信処理は、ACTIVATE で起動されるイベントプロシージャ内で実装します。

処理概要

引数 なし。

復帰値 WS_NO_ERR = 成功,WS_ERR = 接続失敗

注意事項 なし。

サンプル

```
//接続と、ACTIVATE イベントの発生。
long ret = newvcso_000->exec();
```

36.8.2 read 関数の説明

書式 long read(WSCuchar* buffer,long size)

機能 ACTIVATE で起動されるイベントプロシージャ内で接続されたソケットからデータを読み込みます。

処理概要

引数

(in)WSCuchar* buffer	読み込みデータ領域
(in)long len	読み込みデータ領域長

復帰値 読み込んだデータ長 (バイト数)

注意事項 ACTIVATE イベントで起動されるプロシージャ以外で使用した場合、接続が確立されておらず、エラーとなります。

サンプル

```
char buffer［128］;
long ret = newvcso_000->read(buffer,128);
```

36.8.3 write 関数の説明

書式 long write(WSCuchar* buffer,long size)

機能 ACTIVATE で起動されるイベントプロシージャ内で接続されたソケットにデータを書き込みます。

処理概要

引数

(in)WSCuchar* buffer	書き込みデータ領域
(in)long len	書き込みデータ領域長

復帰値 書き込んだデータ長 (バイト数)

注意事項　ACTIVATE イベントで起動されるプロシージャ以外で使用した場合、接続が確立されておらず、エラーとなります。

サンプル

```
char buffer［128］;
strcpy(buffer,"send data...");
long ret = newvcso_000->write(buffer,128);
```

37　WSCvudpsocket

37.1　継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

37.2　概要

ＵＤＰソケットを使った同報通信を行うクラスです。プロパティWSNip に同報通信を行うブロードキャストアドレス、プロパティWSNport にポート番号を指定します。他のＵＤＰソケットからの同報があると、ACTIVATE イベントが発生します。データの受け取りは、その ACTIVATE イベントで起動するプロシージャ内で行います。

37.3　機能

・ＵＤＰソケット同報通信

37.4　注意事項

37.5　プロパティー覧

クラス WSCvudpsocket のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNip	ＩＰ	
unsigned short	WSNport	ポート	9001
unsigned long	WSNtimeout	タイムアウト	60
WSCbool	WSNrunning	実行中	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

37.6　トリガー覧

WSCvudpsocket クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

37.7 関数一覧

- long read(WSCuchar* buffer,long size) :WSCvcsocket
- long write(WSCuchar* buffer,long size) :WSCvcsocket

37.8 関数仕様

37.8.1 read 関数の説明

書式 long read(WSCuchar* buffer,long size)

機能 ACTIVATE で起動されるイベントプロシージャ内で同報通信で送られたデータをＵＤＰソケットから読み込みます。

処理概要

引数

(in)WSCuchar* buffer	読み込みデータ領域
(in)long len	読み込みデータ領域長

復帰値 読み込んだデータ長 (バイト数)

注意事項 ACTIVATE イベントで起動されるプロシージャ以外で使用した場合、データが到着するまで、または、タイムアウトが発生するまで待ちます。

サンプル

```
char buffer［128］;
long ret = newvudp_000->read(buffer,128);
```

37.8.2 write 関数の説明

書式 long write(WSCuchar* buffer,long size)

機能 ソケットに同報通信データを書き込み送信します。

処理概要

引数

(in)WSCuchar* buffer	読み込みデータ領域
(in)long len	読み込みデータ領域長

復帰値 書き込んだデータ長 (バイト数)

注意事項 UDP を利用して一度に送信できるデータ長には限りがあります。512 バイトを目安に上限としてしてください。

サンプル

```
char buffer［128］;
strcpy(buffer,"send data...");
long ret = newvudp_000->write(buffer,128);
```

38 WSCvremoteServer

38.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

38.2 概要

リモートインスタンスサーバを構築するためのクラスです。リモートインスタンスを管理するエージェントと通信を行い、サーバプロセス内に存在するリモートクラスの情報をエージェントに渡します。リモートインスタンスサーバを構築する際に、WSCvremoteServer クラスのインスタンスを一つだけ、プロジェクト内のいずれかのアプリケーションウィンドウに配置します。プロパティ WSNport に 0 を指定した場合、エージェントから自動的にポートを割り当てられます。

プロパティ WSNport は、に 0 を指定した場合、エージェントから自動的にポートを割り当てられます。プロパティ WSNip には通常、特に設定を行いませんが、もし同じマシンに複数のアドレスが存在し、そのアドレスのうち受け付けるアドレスを特定する場合のみ、WSNip を指定します。

38.3 機能

- リモートインスタンス情報のエージェントへの引き渡し。
- サーバプロセスのバインドポートの指定。

38.4 注意事項

38.5 プロパティー覧

クラス WSCvremoteServer のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNip	ＩＰ	
unsigned short	WSNport	ポート	0
WSCbool	WSNrunning	実行中	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30

38.6 トリガー覧

WSCvremoteServer クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

38.7 関数一覧

38.8 関数仕様

39 WSCvremoteClient

39.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

39.2 概要

リモートインスタンスを管理しているエージェントと通信を行う際、エージェントの通信ポート番号を指定します。
リモートインスタンスにアクセスを行う際、エージェントと通信を行いリモートインスタンスの所在情報を取得するのに利用されます。ユーザーからは直接この WSCvremoteClient クラスのインスタンスに対して、アクセスすることはありません。

39.3 機能

・エージェントの通信ポートの指定

39.4 注意事項

39.5 プロパティー覧

クラス WSCvremoteClient のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNip	ＩＰ	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30

39.6 トリガー覧

WSCvremoteClient クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

39.7 関数一覧

39.8 関数仕様

40 WSCvdb

40.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

40.2 概要

ODBC インターフェースや、DB インターフェースを通じて、データーベースのデータの参照、更新を行います。データ参照、データ更新は SQL 文を使います。プロパティ WSNtype によって使用するデータベースインターフェースを選択することが出来ます。

40.3 機能

- 使用するデータベースインターフェースの選択
- データベースとの接続
- データの参照、更新

40.4 注意事項

40.5 プロパティー覧

クラス WSCvdb のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNhostname	ホスト名	localhost
char*	WSNusername	ユーザー名	username
char*	WSNpassword	パスワード	password
char*	WSNdbname	データベース名	dbname
unsigned short	WSNport	ポート	0
unsigned char	WSNtype	表示タイプ	0
	ODBC	(0)	
	PostgreSQL	(1)	
	MySQL	(2)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

40.6 トリガー覧

WSCvdb クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

40.7 関数一覧

- long open(char* hostname,char* username,char* passwd,char* dbname,char* port) :WSCvdb
- long open() :WSCvdb
- long close() :WSCvdb
- WSCbool isOpen() :WSCvdb
- long sqlExecute(const char* sql) :WSCvdb
- long beginTran() :WSCvdb
- long commintTran() :WSCvdb
- long abortTran() :WSCvdb
- void getErrorMsg(char* buf,long size) :WSCvdb

40.8 関数仕様

40.8.1 open 関数の説明

書式 long open(char* hostname,char* username,char* passwd,char* dbname,char* port)

機能 データーベースに接続します。

処理概要 ODBC を通じてデーターベースに接続します。

引数

(in)char* hostname	ホスト名を指定します。
(in)char* username	接続するユーザー名を指定します。
(in)char* passwd	パスワードを指定します。
(in)char* dbname	データベース名を指定します。
(in)char* port	ポート番号を文字列で指定します。

dbname、port は、PostgreSQL を利用する場合に指定します。odbc 利用時は、hostname に dsn を指定し、dbname、port は省略するか、NULL を指定します。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
long ret = newvdb__000->open("10.20.30.1","user","passwd","dbname","5432");
if (ret == WS_NO_ERR){
  //接続。
}else{
  //接続失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvdb__000->getErrorMsg(buffer,1024);
}
```

40.8.2 open 関数の説明

書式 long open()

機能 データーベースに接続します。

処理概要 プロパティの設定値を使ってデーターベースに接続します。

引数 なし。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
long ret = newvdb__000->open();
if (ret == WS_NO_ERR){
  //接続。
```

```
}else{
  //接続失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvdb__000->getErrorMsg(buffer,1024);
}
```

40.8.3 close 関数の説明

書式 long close()
機能 データーベースとの接続を切断します。
処理概要 ODBC を通じてデーターベースへの接続を切断します。
引数 なし。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル

```
long ret = newvdb__000->close();
if (ret == WS_NO_ERR){
  //切断成功。
}else{
  //切断失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvdb__000->getErrorMsg(buffer,1024);
}
```

40.8.4 isOpen 関数の説明

書式 WSCbool isOpen()
機能 データーベースとの接続の状態を取得します。
処理概要 ODBC を通じたデーターベースへの接続の状態を取得します。
引数 なし。
復帰値 True:接続状態、False:非接続状態
注意事項 なし。
サンプル

```
WSCbool ret = newvdb__000->isOpen();
if (ret == False){
  //非接続状態。
}else{
  //接続状態。
}
```

40.8.5 sqlExecute 関数の説明

書式 long sqlExecute(const char* sql)
機能 データーベースに対して SQL 文を発行します。
処理概要 ODBC を通じてデーターベースに SQL 文を発行します。
引数

(in)const char* sql	SQL 文を指定します。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 SQL 文発行結果として取得されるデータは、WSCdbRecord クラスを用いてアクセスします。
サンプル

```
//テーブルの作成とデータの格納
char buf1［1024］;
strcpy(buf1, "drop table shinamono");
long ret = newvdb__000->sqlExecute(buf1);
strcpy(buf1,"create table shinamono(code int, hinmei char(20), nedan float)");
newvdb__000->sqlExecute(buf1);
newvdb__000->beginTran();
strcpy(buf1,"insert into shinamono values(1, 'みかん', 100)");
newvdb__000->sqlExecute(buf1);
```

```
strcpy(buf1,"insert into shinamono values(2, 'りんご', 200)");
newvdb__000->sqlExecute(buf1);
strcpy(buf1,"insert into shinamono values(3, 'バナナ', 300)");
newvdb__000->sqlExecute(buf1);
strcpy(buf1,"insert into shinamono values(4, 'メロン', 0)");
newvdb__000->sqlExecute(buf1);
newvdb__000->commitTran();

//データの参照
WSCdbRecord rs(newvdb__000);
char var[256];
WSCstring result;
newvdb__000->beginTran();
if(rs.open("select * from shinamono order by code") == WS_NO_ERR) {
  while (!rs.isEOF()) {
    rs.getColValue("code", \&var);
    int code = (int)var;
    result << "code:" << (int)var << " ";
    rs.getColValue("hinmei", \&var);
    result << "hinmei:" << (char*)var << " ";
    rs.getColValue("nedan", \&var);
    char buf[80];
    double nedan = (float)var + 10;
    sprintf(buf, "%f", (float)var);
    result << "nedan:" << buf << "\n";
  }
  printf("result:\n%s",(char*)result);
}
```

40.8.6 beginTran 関数の説明

書式 long beginTran()

機能 トランザクションを開始します。

処理概要 ODBC を通じてトランザクションを開始します。

引数 なし。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 データーベースへ接続をしていなければなりません。

サンプル

```
long ret = newvdb__000->beginTran();
if (ret == WS_NO_ERR){
  //トランザクション開始成功。
}else{
  //トランザクション開始失敗、エラーメッセージを取得。
  char buffer[1024];
  newvdb__000->getErrorMsg(buffer,1024);
}
```

40.8.7 commitTran 関数の説明

書式 long commitTran()

機能 トランザクションを確定 (コミット) します。

処理概要 ODBC を通じてトランザクションをコミットします。

引数 なし。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 データーベースへ接続をしていなければなりません。

サンプル

```
long ret = newvdb__000->commitTran();
if (ret == WS_NO_ERR){
  //トランザクションコミット成功。
```

```
}else{
  //トランザクションコミット失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvdb__000->getErrorMsg(buffer,1024);
}
```

40.8.8 abortTran 関数の説明

書式 long abortTran()
機能 トランザクションを途中で中断します。
処理概要 ODBC を通じてトランザクションを中断します。
引数 なし。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 トランザクションを開始していなければなりません。
サンプル

```
long ret = newvdb__000->abortTran();
if (ret == WS_NO_ERR){
  //トランザクション中断成功。
}else{
  //トランザクション中断失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvdb__000->getErrorMsg(buffer,1024);
}
```

40.8.9 getErrorMsg 関数の説明

書式 long getErrorMsg(char* buffer,long buflen)
機能 エラー文字列を取得します。
処理概要
引数

(in/out)char* buf	エラー文字列を格納するバッファ。
(in)long buflen	バッファ長を指定します。

復帰値 なし。
注意事項
サンプル open のサンプルを参照ください。

41 WSCvodbc

41.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCngbase

41.2 概要

ODBC インターフェースを通じて、データーベースのデータの参照、更新を行います。データ参照、データ更新は SQL 文を使います。

41.3 機能

- データベースとの接続
- データの参照、更新

41.4 注意事項

41.5 プロパティー覧

クラス WSCvodbc のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNdsn	DNS	dsn
char*	WSNusername	ユーザー名	username
char*	WSNpassword	パスワード	password
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

41.6 トリガー覧

WSCvodbc クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

41.7 関数一覧

- long open(char* dsn,char* username,char* passwd) :WSCvodbc
- long open() :WSCvodbc
- long close() :WSCvodbc
- WSCbool isOpen() :WSCvodbc
- long sqlExecute(const char* sql) :WSCvodbc
- long beginTran() :WSCvodbc
- long commintTran() :WSCvodbc
- long abortTran() :WSCvodbc
- void getErrorMsg(char* buf,long size) :WSCvodbc

41.8 関数仕様

41.8.1 open 関数の説明

書式 long open(char* dsn,char* username,char* passwd)

機能 データーベースに接続します。

処理概要 ODBC を通じてデーターベースに接続します。

引数

(in)char* dsn	DSN 名を指定します。
(in)char* username	接続するユーザー名を指定します。
(in)char* passwd	パスワードを指定します。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
long ret = newvodb_000->open("postgres","user","passwd");
if (ret == WS_NO_ERR){
  //接続。
}else{
  //接続失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvodb_000->getErrorMsg(buffer,1024);
}
```

41.8.2 open 関数の説明

書式 long open()
機能 データーベースに接続します。
処理概要 プロパティの設定値を使ってデーターベースに接続します。
引数 なし。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル

```
long ret = newvodb_000->open();
if (ret == WS_NO_ERR){
  //接続。
}else{
  //接続失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvodb_000->getErrorMsg(buffer,1024);
}
```

41.8.3 close 関数の説明

書式 long close()
機能 データーベースとの接続を切断します。
処理概要 ODBC を通じてデーターベースへの接続を切断します。
引数 なし。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル

```
long ret = newvodb_000->close();
if (ret == WS_NO_ERR){
  //切断成功。
}else{
  //切断失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvodb_000->getErrorMsg(buffer,1024);
}
```

41.8.4 isOpen 関数の説明

書式 WSCbool isOpen()
機能 データーベースとの接続の状態を取得します。
処理概要 ODBC を通じたデーターベースへの接続の状態を取得します。
引数 なし。
復帰値 True:接続状態、False:非接続状態
注意事項 なし。
サンプル

```
WSCbool ret = newvodb_000->isOpen();
if (ret == False){
  //非接続状態。
}else{
  //接続状態。
}
```

41.8.5 sqlExecute 関数の説明

書式 long sqlExecute(const char* sql)

機能 データーベースに対して SQL 文を発行します。

処理概要 ODBC を通じてデーターベースに SQL 文を発行します。

引数

(in)const char* sql	SQL 文を指定します。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 SQL 文発行結果として取得されるデータは、WSCodbcRecord クラスを用いてアクセスします。

サンプル

```
//テーブルの作成とデータの格納
char buf1[1024];
strcpy(buf1, "drop table shinamono");
long ret = newvodb_000->sqlExecute(buf1);
strcpy(buf1,"create table shinamono(code int, hinmei char(20), nedan float)");
newvodb_000->sqlExecute(buf1);
newvodb_000->beginTran();
strcpy(buf1,"insert into shinamono values(1, 'みかん', 100)");
newvodb_000->sqlExecute(buf1);
strcpy(buf1,"insert into shinamono values(2, 'りんご', 200)");
newvodb_000->sqlExecute(buf1);
strcpy(buf1,"insert into shinamono values(3, 'バナナ', 300)");
newvodb_000->sqlExecute(buf1);
strcpy(buf1,"insert into shinamono values(4, 'メロン', 0)");
newvodb_000->sqlExecute(buf1);
newvodb_000->commitTran();

//データの参照
WSCodbcRecord rs(newvodb_000);
char var[256];
WSCstring result;
newvodb_000->beginTran();
if(rs.open("select * from shinamono order by code") == WS_NO_ERR) {
  while (!rs.isEOF()) {
    rs.getColValue("code", \&var);
    int code = (int)var;
    result << "code:" << (int)var << " ";
    rs.getColValue("hinmei", \&var);
    result << "hinmei:" << (char*)var << " ";
    rs.getColValue("nedan", \&var);
    char buf[80];
    double nedan = (float)var + 10;
    sprintf(buf, "%f", (float)var);
    result << "nedan:" << buf << "\n";
  }
  printf("result:\n%s",(char*)result);
}
```

41.8.6 beginTran 関数の説明

書式 long beginTran()

機能 トランザクションを開始します。

処理概要 ODBC を通じてトランザクションを開始します。

引数 なし。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 データーベースへ接続をしていなければなりません。

サンプル

```
long ret = newvodb_000->beginTran();
if (ret == WS_NO_ERR){
```

```
  //トランザクション開始成功。
}else{
  //トランザクション開始失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvodb_000->getErrorMsg(buffer,1024);
}
```

41.8.7 commitTran 関数の説明

書式 long commitTran()

機能 トランザクションを確定 (コミット) します。

処理概要 ODBC を通じてトランザクションをコミットします。

引数 なし。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 データーベースへ接続をしていなければなりません。

サンプル

```
long ret = newvodb_000->commitTran();
if (ret == WS_NO_ERR){
  //トランザクションコミット成功。
}else{
  //トランザクションコミット失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvodb_000->getErrorMsg(buffer,1024);
}
```

41.8.8 abortTran 関数の説明

書式 long abortTran()

機能 トランザクションを途中で中断します。

処理概要 ODBC を通じてトランザクションを中断します。

引数 なし。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 トランザクションを開始していなければなりません。

サンプル

```
long ret = newvodb_000->abortTran();
if (ret == WS_NO_ERR){
  //トランザクション中断成功。
}else{
  //トランザクション中断失敗、エラーメッセージを取得。
  char buffer［1024］;
  newvodb_000->getErrorMsg(buffer,1024);
}
```

41.8.9 getErrorMsg 関数の説明

書式 long getErrorMsg(char* buffer,long buflen)

機能 エラー文字列を取得します。

処理概要

引数

(in/out)char* buf	エラー文字列を格納するバッファ。
(in)long buflen	バッファ長を指定します。

復帰値 なし。

注意事項

サンプル open のサンプルを参照ください。

42 WSCprocedure

42.1 継承元

次のオブジェクトを継承しています。

42.2 概要

イベントプロシージャクラスです。イベントプロシージャ関数、関数名、起動トリガを管理します。

42.3 機能

- イベントプロシージャ関数の保持
- イベントプロシージャ関数の関数名称の保持
- 起動トリガの保持

42.4 注意事項

42.5 関数一覧

- WSCprocedure(char*,long trg); :WSCprocedure
- long setFunction(void(*)(WSCbase*),char*); :WSCprocedure
- long setProcName(char*); :WSCprocedure
- long setTrigger(long); :WSCprocedure
- void setInternal(WSCbool); :WSCprocedure
- char* getProcName(); :WSCprocedure
- char* getFunctionName(); :WSCprocedure
- long getTrigger(); :WSCprocedure
- WSCbool getInternal(); :WSCprocedure

42.6 関数仕様

42.6.1 WSCprocedure 関数の説明

書式 WSCprocedure(char* epname,long trigger)

機能 イベントプロシージャインスタンスを生成します。

処理概要

引数

(in)char* epname	イベントプロシージャ名称
(in)long trigger	起動トリガ

トリガには次のようなものがあります。

WSEV_NONE	イベントなし
WSEV_INITIALIZE	初期化トリガ
WSEV_DELETE	開放トリガ
WSEV_ACTIVATE	アクティベートトリガ
WSEV_VALUE_CH	値変化トリガ
WSEV_FOCUS_CH	フォーカス変化トリガ
WSEV_VISIBLE_CH	可視属性変化トリガ
WSEV_PARENT_VISIBLE_CH	親インスタンス可視属性変化トリガ
WSEV_SENSITIVE_CH	選択属性変化トリガ
WSEV_PARENT_SENSITIVE_CH	親インスタンス選択属性変化トリガ
WSEV_EXPOSE	露出描画トリガ
WSEV_RESIZE	リサイズトリガ
WSEV_MOUSE_IN	マウスイントリガ
WSEV_MOUSE_OUT	マウスアウトトリガ
WSEV_MOUSE_PRESS	マウスプレストリガ
WSEV_MOUSE_RELEASE	マウスリリーストリガ
WSEV_MOUSE_MOVE	マウスムーブトリガ
WSEV_KEY_PRESS	キープレストリガ
WSEV_KEY_RELEASE	キーリリーストリガ
WSEV_KEY_HOOK	キーフックトリガ

復帰値 なし。
注意事項
サンプル

```
newvbtn_004 = new  WSCvfbtn(newmenu_003,"newvfbt_004");
newvbtn_004->initialize();
newvbtn_004->setProperty(WSNname,"newvfbt_004");
newvbtn_004->setProperty(WSNvis,True);
newvbtn_004->setProperty(WSNx,10);
newvbtn_004->setProperty(WSNy,10);
//プロシージャの作成
WSCprocedure* op = new WSCprocedure("activate proc",WSEV_ACTIVATE);
extern void proc1(WSCbase*);
op->setFunction(proc1,"proc1");
newvbtn_004->addProcedure(op);
```

42.6.2 setFunction 関数の説明

書式 long setFunction(void(*func)(WSCbase*),char* fname)
機能 イベントプロシージャ関数を設定します。
イベントプロシージャ関数の形式は、次の通りです。
void foo (WSCbase*)
処理概要
引数

(in)void(*func)(WSCbase*)	イベントプロシージャ関数アドレス
(in)char* fname	関数名称

復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。
注意事項
サンプル WSCprocedure() を参照してください。

42.6.3 setProcName 関数の説明

書式 long setProcName(char* pname)
機能 イベントプロシージャ名を指定します。
処理概要
引数

(in)char* pname	イベントプロシージャ名

復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。
注意事項
サンプル

```
//プロシージャ名を変更するには次のようにします。
op->setProcName("activate proc2");
```

42.6.4 setTrigger 関数の説明

書式 long setTrigger(long trigger)
機能 トリガーを指定します。
処理概要
引数

(in)long trigger	トリガ

指定する値は、コンストラクタを参照ください。
復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。
注意事項
サンプル

```
//トリガを例えば WSEV_VALUE_CH に変更するには次のようにします。
op->setTrigger(WSEV_VALUE_CH);
```

42.6.5 setInternal 関数の説明

書式 void setInternal(WSCbool fl)
機能 内部属性を指定します。
処理概要 インスタンスが存在することを隠したい場合に指定します。
引数

(in)WSCbool fl	内部属性指定 True=内部、False=通常

復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。
注意事項
サンプル

```
//内部属性を設定するには次のようにします。
op->setInternal(True);
```

42.6.6 getProcName 関数の説明

書式 char* getProcName()
機能 イベントプロシージャ名を取得します。
処理概要
引数 なし。
復帰値 イベントプロシージャ名
注意事項
サンプル

```
//イベントプロシージャ名を取得するには次のようにします。
char* pname = op->getProcName();
```

42.6.7 getFunctionName 関数の説明

書式 char* getFunctionName()
機能 イベントプロシージャに設定されている関数名を取得します。
処理概要
引数 なし。
復帰値 関数名
注意事項
サンプル

```
//関数名を取得するには次のようにします。
char* pname = op->getFunctionName();
```

42.6.8 getTrigger 関数の説明

書式 long getTrigger()
機能 トリガーを取得します。
処理概要
引数 なし。
復帰値 トリガー
注意事項
サンプル

```
//トリガを取得するには次のようにします。
long trigger = op->getTrigger();
```

42.6.9 getInternal 関数の説明

書式 WSCbool getInternal()
機能 内部属性を取得します。
処理概要
引数 なし。
復帰値 内部属性 True: 内部、False:通常
注意事項
サンプル

```
//内部属性を取得するには次のようにします。
WSCbool fl = op->getInternal();
```

43 WSCpulldownMenu

43.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel

43.2 概要

プルダウンメニューです。マウスボタンを押下するとメニューが表示されます。

プロパティ `WSNmenuItems` の設定方法を示します。メニューに設定する文字列は、
メニュー文字列１:起動ＥＰ名称１:キー:値１, メニュー文字列２:起動ＥＰ名称２:キー:値２,...
の様に設定します。

カスケードする場合は、

```
menu1:{menu11:ep1:key:val11,menu12:ep12:key:val12},menu2:ep2:key:val2,...
```

の様に設定します。

メニューの項目毎にそれぞれ、その項目を選択すると起動される起動イベントプロシージャ名称、ショートカットキー、選択された時の値を指定することができます。例えば、「メニュー文字列１:起動ＥＰ名称１:キー:値１」と設定された項目を選択すると、「起動ＥＰ名称１」なる名称を持つイベントプロシージャを起動し、getValue() で取得される値は、値１となります。

43.3 機能

- マウスボタン押下によるメニューの表示機能
- メニューのカスケード表示機能

43.4 注意事項

43.5 プロパティ一覧

クラス WSCpulldownMenu のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNmenuItems	メニュー項目	item1(key1):ep1_name:key1,item2(key2):ep2_name:
char*	WSNshortcut	ショートカットキー	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF5
short	WSNtopShadowColor	上影色	DEF8
short	WSNbottomShadowColor	下影色	DEF9
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	

char*	WSNlabelString	表示文字列	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned char	WSNfont	フォント番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF6
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

43.6 トリガー一覧

WSCpulldownMenu クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT

43.7 関数一覧

- long getItems() :WSCpulldownMenu
- long setItemSensitive(short no,WSCbool flag) :WSCpulldownMenu
- long getValue() :WSCpulldownMenu
- long setValueSensitive() :WSCpulldownMenu
- WSCbool* getItemSensitive() :WSCpulldownMenu
- virtual void onActivate() :WSCpulldownMenu
- long beginCascade(char* lb) :WSCpulldownMenu
- long addItem(char* lb,char* op,char* shortc,long id) :WSCpulldownMenu
- long endCascade() :WSCpulldownMenu

43.8 関数仕様

43.8.1 getItems 関数の説明

書式 long getItems()

機能 メニュー項目数を返します。

処理概要

引数 なし。

復帰値 メニュー項目数。

注意事項 なし。

サンプル

```
long num = newpull_000->getItems();
long i;
//全項目を選択可能状態に設定します。
for(i=0; i<num; i++){
  newpull_000->setItemSensitive(i,True);
}
```

43.8.2 setItemSensitive 関数の説明

書式 long setItemSensitive(short no,WSCbool flag)

機能 上から順に 0,1,2,... でしていされた項目の選択属性を指定します。

処理概要 True ならば、その項目が選択可能に、False ならば、選択不可能に設定します。

引数

(in)short no	0,1,2,... で指定されるメニュー項目
(in)WSCbool fl	選択属性

復帰値 WS_NO_ERR = 正常、WS_ERR = 存在しない項目が指定された場合。

注意事項

サンプル getItems() を参照してください。

43.8.3 getItemSensitive 関数の説明

書式 WSCbool* getItemSensitive()
機能 上から順に 0,1,2,... の順でで選択属性 (選択可能状態) の配列を返します。
処理概要
引数 なし。
復帰値 選択属性の配列
注意事項
サンプル

```
long num = newpull_000->getItems();
//全項目を選択可否属性を取得します。
WSCbool* statuslist = newpull_000->getItemSensitive();
long i;
for(i=0; i<num; i++){
  WSCbool status = statuslist[i] ;
  printf("item=%d status=%d\n",i,status);
}
```

43.8.4 getValue 関数の説明

書式 long getValue()
機能 現在選択されている値を返します。
処理概要
引数 なし。
復帰値 現在選択されているメニューの値。
注意事項
サンプル

```
//選択された項目の選択値を取得します。
long val = newpull_000->getValue();
```

43.8.5 setValueSensitive 関数の説明

書式 long setValueSensitive(short value,WSCbool flag)
機能 指定した値の項目の選択属性を指定します。
処理概要 指定された値のメニュー項目を探し、True ならば、その項目が選択可能に、False ならば、選択不可能に設定します。
引数

(in)short value	選択属性を指定したいメニューの値
(in)WSCbool fl	選択属性

復帰値 WS_NO_ERR = 正常、WS_ERR = 存在しない値が指定された場合。
注意事項
サンプル

```
//指定された値を持つ項目の選択属性を設定します。
//例えば 100 のメニュー値を持つ項目を選択不可にする場合
newpull_000->getValueSensitive(100,False);
```

44 WSCpulldownMenuPopup

44.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow

44.2 概要

ポップアップメニューの基底となるクラスです。

44.3 機能

・ポップアップウィンドウを定義します。

44.4 注意事項

44.5 プロパティー覧

クラス WSCpulldownMenuPopup のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNmenuItems	メニュー項目	
unsigned short	WSNmenuItemHeight	メニュー項目縦幅	30
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	0
short	WSNforeColor	表示色	DEF6
short	WSNbackColor	背景色	DEF5
short	WSNtopShadowColor	上影色	DEF8
short	WSNbottomShadowColor	下影色	DEF9
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	1
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNemboss	エンボス	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

44.6 トリガー一覧

WSCpulldownMenuPopup クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

44.7 関数一覧

44.8 関数仕様

45 WSCradioGroup

45.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

45.2 概要

ラジオボタングループを持つフォームです。ラジオボタングループは、WSCvradio クラスのグループで、択一選択属性になっています。複数選択できる WSCcheckGroup クラスもあります。

45.3 機能

・ラジオボタングループの提供

45.4 注意事項

45.5 プロパティー一覧

クラス WSCradioGroup のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned long	WSNvalue	値	0
char*	WSNtitleString	タイトル文字列	
char*	WSNmenuItems	メニュー項目	item1,item2,item3
unsigned short	WSNmenuItemHeight	メニュー項目縦幅	20
unsigned char	WSNfont	フォント番号	0
unsigned char	WSNmargin	マージン	4
WSCbool	WSNindicatorOn	インジケータ使用	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNindicatorType	インジケータ TYPE	0
	IN-OUT	WS_IN_OUT (0)	
	OUT	WS_OUT (1)	
	IN	WS_IN (2)	
	NONE	WS_NONE (3)	
unsigned char	WSNindicatorSize	インジケータ SIZE	16

unsigned char	WSNindicatorShadow	インジケータ影幅	2
short	WSNindicatorColor	インジケータ色	DEF11
short	WSNindicatorPixmap	インジケータ画	
short	WSNselectColor	選択色	DEF7
short	WSNselectPixmap	インジケータ選択画	
short	WSNlabelPixmap	表示画	
unsigned char	WSNorientation	表示方向	1
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

45.6 トリガー一覧

WSCradioGroup クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

45.7 関数一覧

- long setItemSensitive(short no, WSCbool flag) :WSClist
- WSCbool getItemSensitive(short no) :WSClist
- WSCvradio* getItem(short no) :WSClist
- virtual void onValueChange() :WSClist

45.8 関数仕様

45.8.1 setItemSensitive 関数の説明

書式 long setItemSensitive(short no, WSCbool flag)

機能 指定された番号のラジオボタンの選択属性を設定します。

処理概要

引数

(in)short no	ラジオ番号 (0,1,2,〜)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項 なし。

サンプル

```
//ラジオボタンの選択属性を個々に設定します。
//例えば、先頭のラジオボタンを選択不可にするには次のようにします。
newradi_000->setItemSensitive(0,False);
```

45.8.2 getItemSensitive 関数の説明

書式 WSCbool getItemSensitive(short no);

機能 指定された番号のラジオボタンの選択属性を取得します。

処理概要

引数

(in)short no	ラジオ番号 (0,1,2,〜)

復帰値 ラジオ選択属性

注意事項 なし。

サンプル

```
//個々のラジオボタンの選択属性を取得します。
//例えば、先頭のラジオボタンの選択属性を取得するには次のようにします。
WSCbool fl = newradi_000->getItemSensitive(0);
```

45.8.3 getItem 関数の説明

書式 WSCvradio* getItem(short no);

機能 指定された番号のラジオボタンインスタンスを取得します。

処理概要

引数

(in)short no	ラジオ番号 (0,1,2,〜)

復帰値 ラジオボタンインスタンス

注意事項 なし。

サンプル

```
//個々のラジオボタンを取得します。
//例えば、先頭のラジオボタンを取得するには次のようにします。
WSCvradio* radio = newradi_000->getItem(0);
```

45.8.4 onValueChange 関数の説明

書式 void onValueChange()

機能 値が変化した場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_VALUE_CH) によるイベントプロシージャを用いる代わりに、この関数をオーバーライドすることでも、値の変化に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onValueChange(){
  //ラジオが選択され値が変化した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onValueChange();
}
```

46 WSCscrForm

46.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

46.2 概要

スクロールバーを持った、実際よりも広い領域を持つフォームです。スクロールバーによって、領域をスクロール表示させることができます。

46.3 機能

- 仮想的な大きさの領域の提供
- スクロールバーによる領域の移動

46.4 注意事項

スクロールエリア内は、トリガー、WSEV_MOUSE_PRESS、WSEV_MOUSE_MOVE、WSEV_MOUSE_RELEASE は効きません。スクロールエリア内で、これらのトリガーを使用したい場合は、WSEV_SCR_MOUSE_PRESS、WSEV_SCR_MOUSE_MOVE、WSEV_SCR_MOUSE_RELEASE をご使用ください。

46.5 プロパティー覧

クラス WSCscrForm のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNbarThickness	バー幅	16
unsigned short	WSNworkWidth	仮想領域横幅	1000
unsigned short	WSNworkHeight	仮想領域縦幅	1000
unsigned short	WSNhbarValue	横スクロール位置	0
unsigned short	WSNvbarValue	縦スクロール位置	0
unsigned short	WSNincrement	ボタンスクロール単位	10
unsigned short	WSNpageIncrement	ページスクロール単位	100
WSCbool	WSNhbarVisible	横スクロールバー	1

	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNvbarVisible	縦スクロールバー	1
	不可視	False (0)	
	可視	True (1)	
short	WSNworkBackColor	作業領域背景色	DEF11
short	WSNbarShadowColor	バー背景色	DEF12
unsigned short	WSNscrollWidth	横スクロール単位	30
unsigned short	WSNscrollHeight	縦スクロール単位	30
unsigned char	WSNmargin	マージン	4
WSCbool	WSNvirtualScroll	仮想スクロール	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNframe	フレーム	1
	不可	False (0)	
	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

46.6 トリガー覧

WSCscrForm クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
SCR-MOUSE-PRESS	スクロールエリア内でマウスのボタンが押された場合	WSEV_SCR_MOUSE_PRESS
SCR-MOUSE-RELEASE	スクロールエリア内でマウスのボタンがはなされた場合	WSEV_SCR_MOUSE_RELEASE
SCR-MOUSE-MOVE	スクロールエリア内でマウスポインタが動いた場合	WSEV_SCR_MOUSE_MOVE

46.7 関数一覧

- long getVisibleWidth() :WSCscrForm
- long getVisibleHeight() :WSCscrForm
- WSCbase* getScrFrame() :WSCscrForm
- void onActivate() :WSCscrForm

46.8 関数仕様

46.8.1 getVisibleWidth 関数の説明

書式 long getVisibleWidth()

機能 実際に表示されている表示領域の横幅を返します。

処理概要

引数 なし。

復帰値 実際の表示領域の横幅をドット単位で返します。

注意事項 仮想領域長は、プロパティ WSNworkWidth, スクロールフォームの横幅を取得したい場合は、WSNwidth を参照してください。

サンプル

```
//実際に表示されている表示領域の横幅を取得します。
long vw = newscrF_000->getVisibleWidth();
```

46.8.2 getVisibleHeight 関数の説明

書式 long getVisibleHeight()

機能 実際に表示されている表示領域の縦幅を返します。

処理概要

引数 なし。

復帰値 実際の表示領域の縦幅をドット単位で返します。

注意事項 仮想領域長は、プロパティ WSNworkHeight, スクロールフォームの縦幅を取得したい場合は、WSNheight を参照してください。

サンプル

```
//実際に表示されている表示領域の縦幅を取得します。
long vh = newscrF_000->getVisibleHeight();
```

46.8.3 getScrFrame 関数の説明

書式 WSCbase* getScrFrame()

機能 内部のスクロール領域のインスタンス (WSCscrFrame クラス) を返します。

処理概要

引数 なし。

復帰値 内部のスクロール領域のインスタンス。

注意事項 スクロールフォーム内の領域のインスタンスに、直接アクセスしたい場合に使用します。
サンプル
```
//内部のスクロール領域のインスタンスを返します。
WSCbase* scrform = newscrF_000->getScrFrame();
```

46.8.4 onActivate 関数の説明

書式 void onActivate()
機能 スクロールが完了した場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、スクロール完了に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル
```
void new_class::onActivate(){
  //一連のスクロール動作が完了した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

47 WSCscrFrame

47.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

47.2 概要

WSCscrForm クラスの内部で使用されているクラスで、仮想領域を実現します。WSCscrForm の getScrFrame() メンバ関数で取得できます。

47.3 機能

- 仮想領域の提供

47.4 注意事項

47.5 プロパティー覧

クラス WSCscrFrame のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNtitleHeight	タイトル縦幅	0
unsigned short	WSNhbarValue	横スクロール位置	0
unsigned short	WSNvbarValue	縦スクロール位置	0
unsigned short	WSNworkWidth	仮想領域横幅	1000
unsigned short	WSNworkHeight	仮想領域縦幅	1000
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1

short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

47.6 トリガー一覧

WSCscrFrame クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

47.7 関数一覧

47.8 関数仕様

48 WSCsform

48.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

48.2 概要

複数の領域を持ったフォームで、セパレータを使ったマウス操作により、自由に領域を区分できます。下図の例では、セパレータフォーム上に、スクロールフォームを二つ配置したところです。

48.3 機能

- セパレータに分割された２つ以上の領域の提供
- セパレータによる自由な領域区分の大きさ変更

48.4 注意事項

48.5 プロパティ一覧

クラス WSCsform のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNbarThickness	バー幅	5
char*	WSNbarValue	バー位置	50
unsigned char	WSNorientation	表示方向	1
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
WSCbool	WSNrefreshing	更新	0
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNminimum	最小値	30
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0

	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

48.6 トリガー一覧

WSCsform クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

48.7 関数一覧

- long getStatus() :WSCsform
- long setManaged(WSCbase*,WSCbool) :WSCsform

48.8 関数仕様

48.8.1 getStatus 関数の説明

書式 long getStatus()
機能 状態の取得をおこないます。
処理概要 現在、セパレータによって、領域のサイズが変更中であるか否かを返します。
引数 なし。
復帰値 WS_SFORM_FIXED = 変更中でない (もしくは、サイズ変更が終了)、WS_SFORM_UNDER_RESIZING = サイズ変更中。
注意事項 サイズ変更中は、たくさんのリサイズイベントが発生します。したがって、サイズ変更に伴う処理が追い付かない場合等、この関数で状態を調べることによって、途中の不要な処理を回避することができます。
サンプル

```
long status = newsfor_000->getStatus();
if (status == WS_SFORM_FIXED){
  //サイズ変更中でありません。
}else
if (status == WS_SFORM_UNDER_RESIZING){
  //サイズ変更中です。
}
```

48.8.2 setManaged 関数の説明

書式 long setManaged(WSCbase* inst,WSCbool fl)
機能 指定した子インスタンスの管理属性を指定します。
処理概要 管理属性を False が指定されると、サイズ変更対象から外されます。
引数

(in)WSCbool* inst	子インスタンス
(in)WSCbool fl	管理属性

復帰値 WS_NO_ERR = 正常、それ以外の場合はエラー。
注意事項
サンプル

```
//sform 上に配置された child をサイズ変更対象から外す場合は、
//次のようにします。
newsfor_000->setManaged(child,False);
```

49 WSCstring

49.1 継承元

次のオブジェクトを継承しています。

49.2 概要

汎用文字列構造体です。メモリ管理を内部で自動的に行いますので、簡単に文字列の操作が行えます。

49.3 機能

- エンコーディングの操作
- 文字列のメモリ管理

49.4 注意事項

49.5 関数一覧

- WSCstring() :WSCstring
- WSCstring(char* str,long encoding) :WSCstring

- WSCstring(WSCstring &) :WSCstring
- void setString(char*,long encode = WS_EN_DEFAULT) :WSCstring
- char* getString(long encode = WS_EN_DEFAULT) :WSCstring
- long getChars() :WSCstring
- long getEncoding() :WSCstring
- long isExist(char*,long encode = WS_EN_DEFAULT) :WSCstring
- void addString(const WSCstring& str) :WSCstring
- void addString(const WSCvariant& str,long encoding) :WSCstring
- void addString(char* str,long encoding) :WSCstring
- void cutString(WSCulong char_pos) :WSCstring
- void insertString(WSCulong char_pos,WSCstring&) :WSCstring
- void deleteChar(WSCulong char_pos) :WSCstring
- void deleteChars(WSCulong char_pos,WSCulong len) :WSCstring
- void clear() :WSCstring
- void delLineFeed() :WSCstring
- void delString(char*,long num,long encode) :WSCstring
- void delHeadSpace() :WSCstring
- void delTailSpace() :WSCstring
- void to_upper() :WSCstring
- void to_lower() :WSCstring
- long replaceString(char* src,char* dest,long num,long encoding = WS_EN_DEFAULT) :WSCstring
- long getWords(char* long encoding=WS_EN_DEFAULT) :WSCstring
- long getWordCharPos(long,char*,long encoding=WS_EN_DEFAULT) :WSCstring
- WSCstring getWord(long num,char*,long encoding=WS_EN_DEFAULT) :WSCstring
- operator =(const WSCstring& str) :WSCstring
- operator =(char* str) :WSCstring
- operator =(const WSCvariant& str) :WSCstring
- operator +=(const WSCstring& str) :WSCstring
- operator +=(char* str) :WSCstring
- operator +=(const WSCvariant& str) :WSCstring
- operator ¡¡const WSCstring& str) :WSCstring
- operator ¡¡char* str) :WSCstring
- operator ¡¡const WSCvariant& str) :WSCstring

49.6 関数仕様

49.6.1 WSCstring 関数の説明

書式 WSCstring()

機能 文字列クラスのコンストラクタです。文字列クラスのインスタンスを一つ作成します。

処理概要

引数 なし。

復帰値 文字列クラスのインスタンスへのポインタ

注意事項

サンプル

```
WSCstring str;
str = "文字列";
printf("str=%s\n",(char*)str);
```

49.6.2 WSCstring 関数の説明

書式 WSCstring(char* str,long encoding = WS_EN_DEFAULT)

機能 文字列クラスのコンストラクタです。

処理概要 文字列とその文字列のエンコーディングを指定します。

引数

(in)char* str	文字列
(in)long encoding	文字列のエンコーディング

エンコーディングに指定できる値には、次のようなものがあります。省略すると WS_EN_DEFAULT が指定

されます。

属性値	意味
WS_EN_DEFAULT	現在の設定を指定 (省略時の値)
WS_EN_LOCALE	現在の LANG 環境変数の設定を指定
WS_EN_NONE	設定を行わない
WS_EN_ISO8859_1	ISO8859(1) を指定
WS_EN_ISO8859_2	ISO8859(2) を指定
WS_EN_ISO8859_3	ISO8859(3) を指定
WS_EN_ISO8859_4	ISO8859(4) を指定
WS_EN_ISO8859_5	ISO8859(5) を指定
WS_EN_ISO8859_6	ISO8859(6) を指定
WS_EN_ISO8859_7	ISO8859(7) を指定
WS_EN_ISO8859_8	ISO8859(8) を指定
WS_EN_ISO8859_9	ISO8859(9) を指定
WS_EN_ISO8859_10	ISO8859(10) を指定
WS_EN_ISO8859_11	ISO8859(11) を指定
WS_EN_ISO8859_12	ISO8859(12) を指定
WS_EN_ISO8859_13	ISO8859(13) を指定
WS_EN_ISO8859_14	ISO8859(14) を指定
WS_EN_ISO8859_15	ISO8859(15) を指定
WS_EN_UTF8	UTF8 を指定
WS_EN_KOI8R	KOI8R を指定
WS_EN_EUCJP	EUCJP を指定
WS_EN_SJIS	SJIS を指定
WS_EN_EUCKR	EUCKR を指定
WS_EN_EUCCN	EUCCN を指定
WS_EN_BIG5	BIG5 を指定

復帰値 文字列クラスのインスタンスへのポインタ

注意事項

サンプル

```
//EUCJP コードで文字列を設定する場合
WSCstring str("文字列",WS_EN_EUCJP);
printf("str=%s\n",(char*)str);
```

49.6.3 WSCstring 関数の説明

書式 WSCstring(WSCstring& str)

機能 文字列クラスのコピーコンストラクタです。文字列クラスのインスタンスを一つ作成します。

処理概要

引数

(in)WSCstring& str	文字列インスタンス

復帰値 文字列クラスのインスタンスへのポインタ

注意事項

サンプル

```
WSCstring str("文字列");
WSCstring str2 = str;
printf("str2=%s\n",(char*)str2);
```

49.6.4 setString 関数の説明

書式 void setString(char* str,long encode = WS_EN_DEFAULT)

機能 文字列を設定します。

処理概要

引数

(in)char* str	文字列
(in)long encoding	エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 なし。

注意事項

サンプル

```
WSCstring str;
str.setString("文字列",WS_EN_EUCJP);
printf("str=%s\n",(char*)str);
```

49.6.5 getString 関数の説明

書式 char* getString(long encode = WS_EN_DEFAULT)

機能 文字列を取得します。

処理概要

引数

(in)long encoding	エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 指定されたエンコーディングの文字列

注意事項 返された文字列を解放してはなりません。

サンプル

```
WSCstring str;
str.setString("文字列",WS_EN_EUCJP);
printf("str=%s\n",str.getString());
```

49.6.6 getChars 関数の説明

書式 long getChars()

機能 現在設定されている文字列のキャラクタ数を取得します。

処理概要

引数 なし。

復帰値 文字列のキャラクタ数

注意事項

サンプル

```
WSCstring str;
str.setString("文字列",WS_EN_EUCJP);
printf("str=%d\n",str.getChars()); //3 が返されます。
```

49.6.7 getEncoding 関数の説明

書式 long getEncoding()

機能 現在設定されている文字列のエンコーディングを取得します。

処理概要

引数 なし。

復帰値 エンコーディング
返されるエンコーディングの種類は、WSCstring(char*,long) を参照ください。

サンプル

```
WSCstring str;
str.setString("文字列",WS_EN_EUCJP);
printf("encoding=%d\n",str.getEncoding()); //WS_EN_EUCJP = 20 が返されます。
```

注意事項

49.6.8 isExist 関数の説明

書式 long isExist(char* str,long encoding = WS_EN_DEFAULT)

機能 文字列を検索します。

処理概要

引数

(in)char* str	文字列
(in)long encoding	エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 キャラクタ数による存在した位置、存在しなかった場合は、-1。

注意事項

サンプル

```
if (str.isExist("文字") == -1){
  //"文字"が存在しなかった場合
}else{
  //"文字"が存在した場合
}
```

49.6.9 addString 関数の説明

書式 void addString(const WSCstring& str)

機能 文字列を追加します。

処理概要

引数

(in)WSCstring& str	文字列

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列");
WSCstring str2("追加");
str.addString(str2);
printf("str=%s\n",(char*)str); //"文字列追加"が表示されます。
```

49.6.10 addString 関数の説明

書式 void addString(const WSCvariant& str,long encoding=WS_EN_DEFAULT)

機能 文字列を追加します。

処理概要

引数

(in)WSCvariant& str	文字列
(in)long encoding	エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
WSCvariant val;
val = "追加";
str.addString(val,WS_EN_EUCJP);
printf("str=%s\n",(char*)str); //"文字列追加"が表示されます。
```

49.6.11 addString 関数の説明

書式 void addString(char* str,long encoding=WS_EN_DEFAULT)

機能 文字列を追加します。

処理概要

引数

(in)cha* str	文字列
(in)long encoding	エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
str.addString("追加",WS_EN_EUCJP);
printf("str=%s\n",(char*)str); //"文字列追加"が表示されます。
```

49.6.12 cutString 関数の説明

書式 void cutString(WSCulong pos);

機能 指定されたキャラクタ位置以降の文字列を削除します。

処理概要

引数

(in)WSCulong pos	先頭を 0 としたカットしたい文字の位置

0 を指定すると、文字列を全て削除します。

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
str.cutString(2);
printf("str=%s\n",(char*)str); //"文字"が表示されます。
```

49.6.13 insertString 関数の説明

書式 void insertString(WSCulong pos,WSCstring& str);

機能 指定されたキャラクタ位置に文字列を挿入します。

処理概要

引数

(in)WSCulong pos	先頭を 0 とした挿入したい文字の位置
(in)WSCstring& str	文字列

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
WSCstring str2("挿入",WS_EN_EUCJP);
str.insertString(0,str2);
printf("str=%s\n",(char*)str); //"挿入文字列"が表示されます。
```

49.6.14 deleteChar 関数の説明

書式 void deleteChar(WSCulong pos)

機能 指定されたキャラクタ位置の 1 文字を削除します。

処理概要

引数

(in)WSCulong pos	先頭を 0 としたカットしたい文字の位置

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
str.deleteChar(2);
printf("str=%s\n",(char*)str); //"文字"が表示されます。
```

49.6.15 deleteChars 関数の説明

書式 void deleteChars(WSCulong pos,WSCulong len)

機能 指定されたキャラクタ位置の文字列を削除します。

処理概要

引数

(in)WSCulong pos	先頭を 0 としたカットしたい文字の位置
(in)WSCulong len	削除したい文字の数

復帰値 なし。

注意事項

サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
str.deleteChars(2,1);
printf("str=%s\n",(char*)str); //"文字"が表示されます。
```

49.6.16 clear 関数の説明

書式 void clear()
機能 文字列をクリアします。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
WSCstring str("文字列",WS_EN_EUCJP);
//設定された文字列をクリアするには、次のようにします。
str.clear();
```

49.6.17 delLineFeed 関数の説明

書式 void delLineFeed()
機能 文字列中の改行文字を削除します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
WSCstring str("文字列\n 改行",WS_EN_EUCJP);
str.delLineFeed();
printf("str=%s\n",(char*)str); //"文字列改行"が表示されます。
```

49.6.18 delString 関数の説明

書式 void delString(char* str,lon num,long encoding = WS_EN_DEFAULT)
機能 指定された文字列を指定された個数分、検索して削除します。削除する個数に 0 を指定すると、一致した文字列を全て削除します。
処理概要
引数

(in)char* str	削除したい文字列
(in)long num	削除する個数
(in)long encoding	エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。
復帰値 なし。
注意事項
サンプル

```
WSCstring str("文字列１文字列２文字列",WS_EN_EUCJP);
str.deleteString("文字列",1,WS_EN_EUCJP);
printf("str=%s\n",(char*)str); //"１文字列２文字列"が表示されます。
str.deleteString("文字列",0,WS_EN_EUCJP);
printf("str=%s\n",(char*)str); //"１２"が表示されます。
```

49.6.19 delHeadSpace 関数の説明

書式 void delHeadSpace();
機能 文字列の先頭にある半角スペースをカットします。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
WSCstring str("   文字列",WS_EN_EUCJP);
str.delHeadSpace();
printf("str=%s\n",(char*)str); //"文字列"が表示されます。
```

49.6.20 to_upper 関数の説明

書式 void to_upper();
機能 英字文字列を大文字に変換します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
WSCstring str("abcABC");
str.to_upper();
printf("str=%s\n",(char*)str); //"ABCABC"が表示されます。
```

49.6.21 to_lower 関数の説明

書式 void to_lower();
機能 英字文字列を小文字に変換します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
WSCstring str("abcABC");
str.to_lower();
printf("str=%s\n",(char*)str); //"abcabc"が表示されます。
```

49.6.22 replaceString 関数の説明

書式 long replaceString(char* src,char* dest,long num,long encoding= WS_EN_DEFAULT)
機能 指定された文字列を置換します。
処理概要 src で指定された文字列を、dest で指定された文字列に変換します。num で指定された数分だけ一致したものを置換します。num が 0 の場合、一致するものすべてを置換します。
引数

(in)char* src	置換前文字列
(in)char* dest	置換後文字列
(in)long num	置換個数
(in)long encoding	文字列のエンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。
復帰値 置換された数
注意事項
サンプル

```
WSCstring str("abcabcabc");
str.replaceString("abc","123",1);
printf("str=%s\n",(char*)str); //"123abcabc"が表示されます。
str.replaceString("abc","123",0);
printf("str=%s\n",(char*)str); //"123123123"が表示されます。
```

49.6.23 getWords 関数の説明

書式 long getWords(char* sep,long encoding = WS_EN_DEFAULT)
機能 sep で指定された文字列で区切られたワード数を取得します。sep と encoding を省略すると、空白で区切られたワード数を取得します。、
処理概要
引数

(in)char* sep	区切り文字列
(in)long encoding	文字列エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。
復帰値 置換された数

注意事項

サンプル

```
WSCstring str("ABC,abc,123");
long words = str.getWords(","); //3 が返されます。
long i;
for(i=0; i < words; i++){
  WSCstring word = str.getWord(i,",");
  long wordpos = str.getWordPos(i,",");
  printf("word%d=%s\n",i,(char*)word); //ABC、abc、123 が順番に表示されます。
  printf("   pos=%d\n",wordpos); //ABC、abc、123 の位置が表示されます。
}
```

49.6.24 getWordCharPos 関数の説明

書式 long getWordCharPos(long no,char* sep,long encoding = WS_EN_DEFAULT)

機能 sep で指定された文字列で区切られたワードで先頭を 0 とした no 番目のワードの文字数位置を取得します。sep と encoding を省略すると、空白で区切られたワードの位置を取得します。、

処理概要

引数

(in)long no	ワード番号 (0,1,2〜)
(in)char* sep	区切り文字列
(in)long encoding	文字列のエンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 指定されたワードの位置

注意事項

サンプル getWords() を参照してください。

49.6.25 getWord 関数の説明

書式 WSCstring getWord(long no,char* sep,long encoding = WS_EN_DEFAULT)

機能 sep で指定された文字列で区切られたワードで先頭を 0 とした no 番目のワードを取得します。sep と encoding を省略すると、空白で区切られたワードを取得します。

処理概要

引数

(in)long no	ワード番号 (0,1,2〜)
(in)char* sep	区切り文字列
(in)long encoding	文字列エンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 指定されたワード

注意事項

サンプル getWords() を参照してください。

49.6.26 getLines 関数の説明

書式 long getLines()

機能 行数を取得します。

処理概要

引数 なし。

復帰値 行数

注意事項

サンプル

```
WSCstring str("abc\nabc\nabc");
long lines = str.getLines(); //3 が返されます。
```

49.6.27 = オペレータの説明

書式 WSCstring& operator = (const WSCstring& str);
機能 代入オペレータです。
処理概要
引数

(in)WSCstring& str	代入したい文字列

復帰値 複写された文字列
注意事項
サンプル

```
WSCstring str;
WSCstring str2("abc");
str = str2;
```

49.6.28 = オペレータの説明

書式 WSCstring& operator = (char* str);
機能 代入オペレータです。
処理概要
引数

(in)char* str	代入したい文字列

復帰値 複写された文字列
注意事項
サンプル

```
WSCstring str;
str = "abc"; //代入
```

49.6.29 = オペレータの説明

書式 WSCstring& operator = (const WSCvariant& str);
機能 代入オペレータです。
処理概要
引数

(in)WSCvariant& str	代入したいデータ

復帰値 複写された文字列
注意事項
サンプル

```
WSCvariant str("abc");
WSCstring str2;
str2 = str; //WSCvarinat の代入
```

49.6.30 += オペレータの説明

書式 WSCstring& operator += (const WSCstring& str);
機能 文字列追加オペレータです。
処理概要
引数

(in)WSCstring& str	追加したい文字列

復帰値 追加された文字列
注意事項
サンプル

```
WSCstring str("abc");
WSCstring str2("ABC");
str += str2;
printf("str=%s\n",(char*)str); //"abcABC" が表示されます。
```

49.6.31 += オペレータの説明

書式 WSCstring& operator += (char* str);
機能 文字列追加オペレータです。
処理概要
引数

(in)char* str	追加したい文字列

復帰値 追加された文字列
注意事項
サンプル

```
WSCstring str("abc");
char* str2 = "ABC";
str += str2;
printf("str=%s\n",(char*)str); //"abcABC" が表示されます。
```

49.6.32 += オペレータの説明

書式 WSCstring& operator += (const WSCvariant& str);
機能 文字列追加オペレータです。
処理概要
引数

(in)WSCvariant& str	追加したいデータ

復帰値 追加された文字列
注意事項
サンプル

```
WSCstring str("abc");
WSCvariant str2;
str2 = "ABC";
str += str2;
printf("str=%s\n",(char*)str); //"abcABC" が表示されます。
```

49.6.33 ¡¡ オペレータの説明

書式 WSCstring& operator ¡¡ (const WSCstring& str);
機能 文字列追加オペレータです。
処理概要
引数

(in)WSCstring& str	追加したい文字列

復帰値 追加された文字列
注意事項
サンプル

```
WSCstring str("abc");
WSCstring str2("ABC");
str << str2;
printf("str=%s\n",(char*)str); //"abcABC" が表示されます。
```

49.6.34 ¡¡ オペレータの説明

書式 WSCstring& operator ¡¡ (char* str);
機能 文字列追加オペレータです。
処理概要
引数

(in)char* str	追加したい文字列

復帰値 追加された文字列
注意事項
サンプル

```
WSCstring str("abc");
char* str2 = "ABC";
str << str2;
printf("str=%s\n",(char*)str); //"abcABC" が表示されます。
```

49.6.35 ¡¡ オペレータの説明

書式 WSCstring& operator ¡¡ (const WSCvariant& str);
機能 文字列追加オペレータです。
処理概要
引数

(in)WSCvariant& str	追加したいデータ

復帰値 追加された文字列
注意事項
サンプル

```
WSCstring str("abc");
WSCvariant str2;
str2 = "ABC";
str << str2;
printf("str=%s\n",(char*)str); //"abcABC" が表示されます。
```

49.6.36 + オペレータの説明

書式 WSCstring& operator + (const WSCstring& str);
機能 文字列合成オペレータです。
処理概要
引数

(in)WSCstring& str	追加したい文字列

復帰値 合成された文字列
注意事項
サンプル

```
WSCstring str("abc");
WSCstring str2("ABC");
WSCstring str3;
str3 = str + str2;
printf("str3=%s\n",(char*)str3); //"abcABC" が表示されます。
```

49.6.37 + オペレータの説明

書式 WSCstring& operator + (char* str);
機能 文字列合成オペレータです。
処理概要
引数

(in)char* str	追加したい文字列

復帰値 合成された文字列
注意事項
サンプル

```
WSCstring str("abc");
char*  str2 = "ABC";
WSCstring str3;
str3 = str + str2;
printf("str3=%s\n",(char*)str3); //"abcABC" が表示されます。
```

49.6.38 + オペレータの説明

書式 WSCstring& operator + (const WSCvariant& str);
機能 文字列合成オペレータです。
処理概要
引数

(in)WSCvariant& str	追加したいデータ

復帰値 合成された文字列
注意事項

サンプル

```
WSCstring str("abc");
WSCvariant str2;
str2 = "ABC";
WSCstring str3;
str3 = str + str2;
printf("str3=%s\n",(char*)str3); //"abcABC" が表示されます。
```

50 WSCtextField

50.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

50.2 概要

スクロールバーを持った、複数行のテキスト入力フィールドです。

50.3 機能

- キーボードからの複数行のテキスト入力の提供
- 日本語入力機能
- スクロールバーによるテキストスクロール機能の提供

50.4 注意事項

スクロールエリア内は、トリガー、WSEV_MOUSE_PRESS、WSEV_MOUSE_MOVE、WSEV_MOUSE_RELEASE は効きません。スクロールエリア内で、これらのトリガーを使用したい場合は、WSEV_SCR_MOUSE_PRESS、WSEV_SCR_MOUSE_MOVE、WSEV_SCR_MOUSE_RELEASE をご使用ください。

50.5 プロパティー覧

クラス WSCtextField のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNbarThickness	バー幅	16
short	WSNworkBackColor	作業領域背景色	DEF11
short	WSNbarShadowColor	バー背景色	DEF12
unsigned char	WSNmargin	マージン	4
WSCbool	WSNenableInput	入力可否	1
	オフ	False (0)	
	オン	True (1)	
unsigned long	WSNvbarValue	縦スクロール位置	0
unsigned long	WSNhbarValue	横スクロール位置	0
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	1
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	

	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	
short	WSNcursorPos	カーソル位置	0
WSCbool	WSNcursorAdjust	カーソル補正	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNnoHatch	活性表示維持	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNreturnKeyFocus	リターンキーフォーカス	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNdisplayOnly	表示のみ	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	

short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	

50.6 トリガー一覧

WSCtextField クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
KEY-HOOK	キーイベントの横取りを行う場合	WSEV_KEY_HOOK
SCR-MOUSE-PRESS	スクロールエリア内でマウスのボタンが押された場合	WSEV_SCR_MOUSE_PRESS
SCR-MOUSE-RELEASE	スクロールエリア内でマウスのボタンがはなされた場合	WSEV_SCR_MOUSE_RELEASE
SCR-MOUSE-MOVE	スクロールエリア内でマウスポインタが動いた場合	WSEV_SCR_MOUSE_MOVE

50.7 関数一覧

- WSCbase* getTextFrame() :WSCtextField
- void addString(char*) :WSCtextField
- WSCushort* getBuf() :WSCtextField
- void onActivate() :WSCtextField
- void onValueChage() :WSCtextField
- void onKey(WSDkeyboard* keyboard,WSCbool fl) :WSCtextField
- void onKeyHook(WSDkeyboard* keyboard) :WSCtextField
- replaceSelectedString(char* str,long encoding) :WSCtextField
- getSelectedString() :WSCtextField
- getString() :WSCtextField
- deleteSelectedString() :WSCtextField
- setSelect(long pos1,long pos2) :WSCtextField
- getSelectedPos() :WSCtextField
- getLines() :WSCtextField
- getTopLine() :WSCtextField
- setTopLine(long pos) :WSCtextField
- getBottomLine() :WSCtextField
- setBottomLine(long pos) :WSCtextField

50.8 関数仕様

50.8.1 getTextFrame 関数の説明

書式 WSCbase* getTextFrame()

機能 テキストフィールド内部の複数行入力テキストオブジェクト (WSCvmifield クラス) を返します。

処理概要

引数 なし。

復帰値 内部の複数行入力テキストオブジェクト。
注意事項
サンプル
```
//内部の WSCvifield インスタンスを取得するには、次のようにします。
WSCbase* ifield = newtext_000->getTextFrame();
```

50.8.2 addString 関数の説明

書式 void addString(char* var,long encoding = WS_EN_DEFAULT)
機能 現在表示中の文字列に指定した文字列を追記します。
処理概要
引数

(in)char* var	追加したい文字列
(in)long encoding	文字列のエンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。
復帰値 なし。
注意事項 文字列を置き換えたい場合は、プロパティ WSNlabelString を用いてください。
サンプル
```
//現在表示中の文字列に指定された文字列を追加します。
newtext_000->addString("文字列の追加");
```

50.8.3 getBuf 関数の説明

書式 WSCushort* getBuf()
機能 内部のテキストバッファを返します。
処理概要
引数 なし。
復帰値 テキストバッファ。
注意事項 返されたバッファを開放してはいけません。また、キー入力が行われ、バッファの内容が変更されると、領域は無効となります。したがって、この関数が返す領域の永続的な利用は避け、なるべく、他の領域にコピーしてください。このバッファの内容は、UCS2 コードです。
サンプル
```
//現在表示中の文字列の UCS2 文字列バッファを取得します。
WSCushort* buf = newtext_000->getBuf();
```

50.8.4 onActivate 関数の説明

書式 void onActivate()
機能 スクロールバーでのスクロール完了した場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、スクロールに関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル
```
void new_class::onActivate(){
  //一連のスクロール動作が完了した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

50.8.5 onValueChange 関数の説明

書式 void onValueChange()
機能 文字列が入力され、変化した場合に実行されます。
処理概要 アプリケーションは、トリガ (WSEV_VALUE_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、文字列入力に関するイベント処理を行うことができます。
引数 なし。

復帰値 なし。

注意事項 なし。

サンプル
```
void new_class::onValueChange(){
  //入力が行われ、文字列が変更された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onValueChange();
}
```

50.8.6 onKey() 関数の説明

書式 void onKey(WSDkeyboard* keyboard,WSCbool keydown);

機能 キーボードが押下または離された場合に onKey( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_KEY_PRESS/WSEV_KEY_RELEASE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、キーボード入力に関するイベント処理を行うことができます。

引数

(in)WSDkeyboard* keyboard	キーボードインスタンス
(in)WSCbool keydown	True = Press、False = Release

復帰値 なし。

注意事項

サンプル
```
void new_class::onKey(WSDkeyboard* keyboard,WSCbool keydown){
  //キー入力された場合に呼び出されます。
  if (keydown != False){   //キー押下
    //キーの取得
    long key = keyboard->getKey();
    //入力文字列の取得
    WSCstring str = keyboard->getText();
  }

  //処理を派生元クラスに引き継ぎます。
  old_class::onKey(keyboard,keydown);
}
```

50.8.7 onKeyHook() 関数の説明

書式 void onKeyHook(WSDkeyboard* keyboard);

機能 キーボードをフックします。キーボード入力が行われた場合、onKeyHook( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_KEY_HOOK) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、キーボード入力に関するイベント処理を行うことができます。

引数

(in)WSDkeyboard* keyboard	キーボードインスタンス

復帰値 なし。

注意事項

サンプル
```
void new_class::onKeyHook(WSDkeyboard* keyboard){
  //キー入力された場合に呼び出されます。
  //入力文字列の取得
  WSCstring str = keyboard->getText();
  //keyboard->setText(...) で、文字列を置き換えることで、
  //文字列入力を操作することができます。

  //処理を派生元クラスに引き継ぎます。
  old_class::onKeyHook(keyboard);
}
```

50.8.8 replaceSelectedString() 関数の説明

書式 void replaceSelectedString(char* str,long encoding = WS_EN_DEFAULT);

機能 選択された状態の文字列を指定された文字列で置き換えます。

処理概要

引数

(in)char* str	置換後の文字列
(in)long encoding	文字列のエンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 なし。

注意事項

サンプル

```
//現在選択状態にある文字列を指定した文字列で置換します。
newtext_000->replaceSelectedString("置換文字列");
```

50.8.9 getSelectedString() 関数の説明

書式 WSCstring getSelectedString()

機能 選択された状態の文字列を取得します。

処理概要

引数 なし。

復帰値 選択文字列

注意事項

サンプル

```
//現在選択状態にある文字列を取得します。
WSCstring stext = newtext_000->getSelectedString();
```

50.8.10 getString() 関数の説明

書式 WSCstring getString()

機能 入力文字列を取得します。

処理概要

引数 なし。

復帰値 入力文字列

注意事項

サンプル

```
//現在の入力文字列を取得します。
WSCstring text = newtext_000->getString();
```

50.8.11 deleteSelectedString() 関数の説明

書式 void deleteSelectedString()

機能 選択文字列を削除します。

処理概要 選択された文字列の部分だけ削除します。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
//現在選択状態にある文字列を削除します。
newtext_000->deleteSelectedString();
```

50.8.12 setSelect() 関数の説明

書式 long setSelect(long pos,long len)

機能 指定された位置の文字列を選択状態にします。

処理概要 文字位置 pos から指定された長さの文字列を選択状態にします。

引数

(in)long pos	文字列開始位置
(in)long len	文字列長

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
//指定した位置の文字列を選択状態にします。
//例えば先頭から５文字目までを選択状態にするには次のようにします。
newtext_000->setSelectedString(0,5);
```

50.8.13 getSelectedPos() 関数の説明

書式 long setSelectedPos()
機能 選択文字列の先頭からの文字数を取得します。
処理概要
引数 なし。
復帰値 先頭からの文字数
注意事項
サンプル

```
//選択状態にある文字列の先頭からの文字数を取得します。
long pos = newtext_000->setSelectedPos();
```

50.8.14 getLines() 関数の説明

書式 long setLines()
機能 行数を取得します。
処理概要
引数 なし。
復帰値 行数
注意事項
サンプル

```
//現在の入力文字列の行数を取得します。
long pos = newtext_000->getLines();
```

50.8.15 getTopLine() 関数の説明

書式 long getTopLine()
機能 現在表示中の最上段の行を取得します。
処理概要
引数 なし。
復帰値 表示中の最上段の行
注意事項
サンプル

```
//現在、最上段に表示されている入力文字列の行を取得します。
long pos = newtext_000->getTopLine();
```

50.8.16 setTopLine() 関数の説明

書式 long setTopLine(long line)
機能 最上段の行を指定し、スクロールさせます。
処理概要
引数

(in)long line	行 (0,1,2〜)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル

```
//最上段に表示すべき入力文字列の行を指定します。
//たとえば、11 行目を最上段にスクロールさせたい場合は次のように指定します。
newtext_000->setTopLine(10);
```

50.8.17 getBottomLine() 関数の説明

書式 long getBottomLine()
機能 現在表示中の最下段の行を取得します。
処理概要
引数 なし。
復帰値 表示中の最下段の行
注意事項
サンプル

```
//現在最下段に表示されている入力文字列の行を取得します。
long line = newtext_000->getBottomLine();
```

50.8.18 setBottomLine() 関数の説明

書式 long setBottomLine(long line)
機能 現在表示中の最下段の行を指定します。
処理概要
引数

(in)long line	行 (0,1,2〜)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項 表示行数の関係から、最下段に届かない場合は近い行でとまります。
サンプル

```
//最下段に表示すべき入力文字列の行を指定します。
//たとえば、11 行目を最下段にスクロールさせたい場合は次のように指定します。
newtext_000->setBottomLine(10);
```

51 WSCtform

51.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

51.2 概要

タイトルと、枠で囲まれたフォームです。このタイトル付きフォームを使うことで、グループ毎にコントロールを整頓して配置することができます。

51.3 機能

- タイトルの表示
- 枠の表示

51.4 注意事項

51.5 プロパティー覧

クラス WSCtform のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNtitleString	タイトル文字列	
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	

WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

51.6 トリガー一覧

WSCtform クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

51.7 関数一覧

51.8 関数仕様

52 WSCtreeList

52.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform WSCscrForm WSClist

52.2 概要

ツリー表示を行うリストです。WSClist クラスの WSNtype プロパティをツリー表示にしたものと同じです。データソースを指定すると、ダイレクトにツリー表示を行うことが出来ます。

データのフォーマット：
アイコンのパス名, インデントレベル,OPEN/CLOSE, 項目文字列
アイコンのパス名, インデントレベル,OPEN/CLOSE, 項目文字列
...

アイコンのパス名は省略可能です。省略すると、WSCiconPixmap で指定されたものが使用されます。インデントレベルには、０、１、２〜を指定します。直接的には項目間に親子関係の依存関係はなく、このインデントを指定することでのみ、ツリー表現されます。また、一つ前の項目よりインデントレベルが＋２以上の場合、＋１になるように自動的に補正されます。OPEN/CLOSE には、1 (OPEN)、0 (CLOSE) のどちらかを指定します。0 を指定すると、その項目よりもインデントレベルが大きいそれ以降のものはクローズ状態になり表示されません。

52.3 機能

- リストのツリー表示機能
- データソース指定によるデータの直接指定

52.4 注意事項

52.5 プロパティー覧

クラス WSCtreeList のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNitemHeight	項目縦幅	20
short	WSNiconPixmap	アイコン画	$(WSDIR)/sys/pixmaps/bi17.xpm
WSCbool	WSNuseIcon	アイコン使用	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNfont	フォント番号	0
short	WSNselectColor	選択色	DEF7
short	WSNselectForeColor	選択表示色	DEF18
WSCbool	WSNmultiSelect	複数選択	0
WSCbool	WSNreverseSelect	選択時反転	1
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNtitleHeight	タイトル縦幅	0
char*	WSNtitleString	タイトル文字列	title
char*	WSNbarValue	バー位置	20
char*	WSNdata	データ	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	

	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNenableInput	入力可否	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNseparator	セパレータ	,
unsigned short	WSNbarThickness	バー幅	16
unsigned short	WSNworkWidth	仮想領域横幅	1000
unsigned short	WSNworkHeight	仮想領域縦幅	1000
unsigned short	WSNhbarValue	横スクロール位置	0
unsigned short	WSNvbarValue	縦スクロール位置	0
unsigned short	WSNincrement	ボタンスクロール単位	10
unsigned short	WSNpageIncrement	ページスクロール単位	100
WSCbool	WSNhbarVisible	横スクロールバー	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNvbarVisible	縦スクロールバー	1
	不可視	False (0)	
	可視	True (1)	
short	WSNworkBackColor	作業領域背景色	DEF11
short	WSNbarShadowColor	バー背景色	DEF12
unsigned short	WSNscrollWidth	横スクロール単位	30
unsigned char	WSNmargin	マージン	4
WSCbool	WSNvirtualScroll	仮想スクロール	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNframe	フレーム	1
	不可	False (0)	
	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0

不可	False (0)
可	True (1)

52.6 トリガー一覧

WSCtreeList クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
SORT	序列変更となった場合	WSEV_SORT
ITEM-SELECTED	項目が選択された場合	WSEV_ITEM_SELECTED
INPUT-FIXED	入力が確定状態になった場合	WSEV_INPUT_FIXED
SCR-MOUSE-PRESS	スクロールエリア内でマウスのボタンが押された場合	WSEV_SCR_MOUSE_PRESS
SCR-MOUSE-RELEASE	スクロールエリア内でマウスのボタンがはなされた場合	WSEV_SCR_MOUSE_RELEASE
SCR-MOUSE-MOVE	スクロールエリア内でマウスポインタが動いた場合	WSEV_SCR_MOUSE_MOVE

52.7 関数一覧

52.8 関数仕様

53 WSCvarc

53.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvpolyAttr

53.2 概要

円、円弧などの描画を行うオブジェクトです。

53.3 機能

- 円、楕円の表示
- 扇型、ハッチパターンによる内部の塗りつぶし表示

53.4 注意事項

53.5 プロパティー一覧

クラス WSCvarc のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNangle1	円弧開始角度	0
short	WSNangle2	円弧終了角度	360
char	WSNarcType	円タイプ	0

	開円弧	(0)	
	閉円弧	(1)	
	扇形	(2)	
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
short	WSNhatchBlinkColor	ハッチパターンブリンク色	DEF13
WSCbool	WSNbackColorFlag	背景色有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackColor	背景色	DEF13
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0

unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

53.6 トリガー一覧

WSCvarc クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE

53.7 関数一覧

53.8 関数仕様

54 WSCvariant

54.1 継承元

次のオブジェクトを継承しています。

54.2 概要

汎用変数クラスです。型を気にすることなく値を管理できます。

54.3 機能

- 値の保持
- 型の管理、型の変換

54.4 注意事項

54.5 関数一覧

- getChar() :WSDvariant
- getUnsignedChar() :WSDvariant
- getShort() :WSDvariant
- getUnsignedShort() :WSDvariant
- getLong() :WSDvariant
- getUnsignedLong() :WSDvariant
- getInt() :WSDvariant
- getUnsignedInt() :WSDvariant
- getVoidPtr() :WSDvariant

- getCharPtr() :WSDvariant
- getFloat() :WSDvariant
- getDouble() :WSDvariant
- setValue(...) :WSDvariant
- WSCvariant(...) :WSDvariant
- operator = :WSDvariant
- long getType() :WSDvariant
- char* getTypeName() :WSDvariant
- void clear() :WSDvariant

54.6 関数仕様

54.6.1 getChar 関数の説明

書式 char getChar()
機能 値を char 型で取得します。
処理概要
引数 なし。
復帰値 char 型に変換された値
注意事項
サンプル

```
WSCvariant variant;
variant = 100;
char val = variant.getChar();
```

54.6.2 getUnsignedChar 関数の説明

書式 WSCuchar getUnsignedChar()
機能 値を unsigned char 型で取得します。
処理概要
引数 なし。
復帰値 unsigned char 型に変換された値
注意事項
サンプル

```
WSCvariant variant;
variant = 100;
WSCuchar val = variant.getUnsignedChar();
```

54.6.3 getShort 関数の説明

書式 short getShort()
機能 値を short 型で取得します。
処理概要
引数 なし。
復帰値 short 型に変換された値
注意事項
サンプル

```
WSCvariant variant;
variant = 100;
short val = variant.getShort();
```

54.6.4 getUnsignedShort 関数の説明

書式 WSCushort getUnsignedShort()
機能 値を unsigned short 型で取得します。
処理概要
引数 なし。

復帰値 unsigned short 型に変換された値
サンプル
```
WSCvariant variant;
variant = 100;
WSCushort val = variant.getUnsignedShort();
```
注意事項

54.6.5 getLong 関数の説明

書式 long getLong()
機能 値を long 型で取得します。
処理概要
引数 なし。
復帰値 long 型に変換された値
注意事項
サンプル
```
WSCvariant variant;
variant = 100;
long val = variant.getLong();
```

54.6.6 getUnsignedLong 関数の説明

書式 WSCulong getUnsignedLong()
機能 値を unsigned long 型で取得します。
処理概要
引数 なし。
復帰値 unsigned long 型に変換された値
注意事項
サンプル
```
WSCvariant variant;
variant = 100;
WSCulong val = variant.getUnsignedLong();
```

54.6.7 getInt 関数の説明

書式 int getInt()
機能 値を int 型で取得します。
処理概要
引数 なし。
復帰値 int 型に変換された値
注意事項
サンプル
```
WSCvariant variant;
variant = 100;
int val = variant.getInt();
```

54.6.8 getUnsignedInt 関数の説明

書式 WSCuint getUnsignedInt()
機能 値を unsigned int 型で取得します。
処理概要
引数 なし。
復帰値 unsigned int 型に変換された値
注意事項
サンプル
```
WSCvariant variant;
variant = 100;
WSCuint val = variant.getUnsignedInt();
```

54.6.9 getVoidPtr 関数の説明

書式 void* getVoidPtr()
機能 値を void* 型で取得します。
処理概要
引数 なし。
復帰値 void* 型に変換された値
注意事項
サンプル

```
WSCvariant variant;
variant = 100;
void* ptr = variant.getVoidPtr();
```

54.6.10 getCharPtr 関数の説明

書式 char* getCharPtr()
機能 値を char* 型で取得します。
処理概要
引数 なし。
復帰値 char* 型に変換された値
注意事項 返された文字列を開放してはいけません。
サンプル

```
WSCvariant variant;
variant = 100;
char* ptr = variant.getCharPtr(); //文字列の "100" が返されます。
```

54.6.11 getFloat 関数の説明

書式 float getFloat()
機能 値を float 型で取得します。
処理概要
引数 なし。
復帰値 float 型に変換された値
注意事項
サンプル

```
WSCvariant variant;
variant = 100;
float val = variant.getFloat(); //100.0 が返されます。
```

54.6.12 getDouble 関数の説明

書式 double getDouble()
機能 値を double 型で取得します。
処理概要
引数 なし。
復帰値 double 型に変換された値
注意事項
サンプル

```
WSCvariant variant;
variant = 100;
double val = variant.getDouble(); //100.0 が返されます。
```

54.6.13 setValue 関数の説明

書式 void setValue(TYPE val)
機能 値を設定します。
TYPE には、char,WSCuchar,short,WSCushort,long,WSCulong, int,WSCuint,char*,void*,float,double が指定できます。

処理概要

引数

(in)TYPE val	設定したい値

復帰値 なし。

注意事項

サンプル

```
WSCvariant variant;
//値をセットします。
variant.setValue(100);
printf("val=%s\n",(char*)variant); // "100" が返されます。
printf("val=%d\n",(long)variant); // 100 が返されます。
```

54.6.14 WSCvariant 関数の説明

書式 WSCvariant(TYPE val)

機能 コンストラクタです。設定したい値を指定できます
TYPE には、char,WSCuchar,short,WSCushort,long,WSCulong, int,WSCuint,char*,void*,float,double が指定できます。

処理概要

引数

(in)TYPE val	設定したい値

復帰値 なし。

注意事項

サンプル

```
WSCvariant variant(100);
printf("val=%s\n",(char*)variant); // "100" が返されます。
printf("val=%d\n",(long)variant); // 100 が返されます。
```

54.6.15 = オペレータの説明

書式 WSCvariant & operator = (TYPE)

機能 代入演算子です。設定したい値を代入できます
TYPE には、char,WSCuchar,short,WSCushort,long,WSCulong, int,WSCuint,char*,void*,float,double が指定できます。

処理概要

引数 通常の代入演算子 = と同じように使用します。

復帰値 なし。

注意事項

サンプル setLong() を参照してください。

54.6.16 getType 関数の説明

書式 long getType()

機能 格納されているデータの型を取得します。

処理概要

引数 なし。

復帰値　型識別子

WSTlong	long 型
WSTulong	unsinged long 型
WSTint	int 型
WSTuint	unsigned int 型
WSTshort	short 型
WSTushort	unsigned short 型
WSTchar	char 型
WSTuchar	unsigned char 型
WSTfloat	float 型
WSTdouble	double 型
WSTvoidptr	voidptr 型
WSTcharptr	charptr 型
WSTvoid	void 型
WSTbool	bool 型

注意事項

サンプル

```
WSCvariant variant(100);
long type = variant.getType();
```

54.6.17 getTypeName 関数の説明

書式　char* getTypeName()

機能　格納されているデータの型名称を取得します。

処理概要

引数　なし。

復帰値　型名称

注意事項

サンプル

```
WSCvariant variant(100);
char* typename = variant.getTypeName();
```

54.6.18 clear 関数の説明

書式　void clear()

機能　格納されているデータを消去します。

処理概要

引数　なし。

復帰値　なし。

注意事項　文字列データをクリアする場合は、空文を代入する必要があります。

サンプル

```
WSCvariant variant(100);
//格納されている値をクリアします。100 が消え 0 になります。
variant.clear();
```

55 WSCvarrow

55.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel WSCvbtn

55.2 概要

矢印を表示するボタンオブジェクトです。矢印を表示する以外、機能においてはボタンクラスと同等です。

55.3 機能

・矢印の表示

55.4 注意事項

55.5 プロパティ一覧

クラス WSCvarrow のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char	WSNarrowDirection	矢印方向	3
	上	WS_UP (3)	
	下	WS_DOWN (4)	
	左	WS_LEFT (0)	
	右	WS_RIGHT (1)	
short	WSNpushPixmap	押下画	
WSCbool	WSNenableFocusMove	フォーカス移動	1
	無	False (0)	
	有	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	

	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

55.6 トリガー一覧

WSCvarrow クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE

55.7 関数一覧

55.8 関数仕様

56 WSCvballoonHelp

56.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCngbase

56.2 概要

バルーンヘルプを表示したいオブジェクトをこのバルーンヘルプに登録しておくと、オブジェクト上にマウスが動かされた時に、登録時に指定しておいた文字列でバルーンヘルプが表示されます。このクラスは、Non GUI オブジェクトで、指定されたバルーンヘルプを表示したいオブジェクトを WSCballoonHelp インスタンスに登録を代行するものです。ビルダー上から、簡単にバルーンヘルプの設定が行えます。プロパティ WSNclient に表示したいオブジェクトのインスタンス名を指定し、プロパティ WSNlabelString に表示したい文字列を指定します。

56.3 機能

・バルーンヘルプを表示したい登録されたオブジェクトを記憶します。
・登録されたオブジェクト上にマウスポインタが移動するとバルーンヘルプを表示します。

56.4 注意事項

56.5 プロパティー覧

クラス WSCvballoonHelp のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNclient	クライアント	
char*	WSNlabelString	表示文字列	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

56.6 トリガー一覧

WSCvballoonHelp クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

56.7 関数一覧

56.8 関数仕様

57 WSCvbarGraph

57.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

57.2 概要

棒グラフを表示します。プロパティ WSNvalue に、グラフ表示したいデータをカンマで区切って設定します。

`WSNvalue` 値の設定:
`10,20,30,40,50,60,70,...`
のように値を設定します。

57.3 機能

- 棒グラフの表示
- データソース指定によるデータの直接設定機能

57.4 注意事項

プロパティ WSNvalue はデータソース対象プロパティとして用いられます。WSNdataSource プロパティ(データソースタイプ) がインスタンスの場合、WSNdataSourceName プロパティに指定された値は、インスタンス名として扱われ、その指定されたインスタンスのデータソース対象プロパティから値が取得されます。また、ファイルの場合は、ファイル名として扱われ、ファイルからデータが読み込まれます。

57.5 プロパティ一覧

クラス WSCvbarGraph のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNvalue	値	0,20,40,60,80,100
unsigned short	WSNcolumns	カラム数	10
unsigned char	WSNbarThickness	バー幅	10
unsigned char	WSNorientation	表示方向	1
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
WSCbool	WSNreverseFlag	反転表示有無	0
	False	False (0)	
	True	True (1)	
unsigned char	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	

char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

57.6 トリガー一覧

WSCvbarGraph クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

57.7 関数一覧

57.8 関数仕様

58 WSCvbtn

58.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel

58.2 概要

ボタンクラスです。マウスによりクリックされるとへこみ、そのまま領域内で放されると、アクティベートイベントが発生します。

58.3 機能

- ボタン動作によるイベントの発生

58.4 注意事項

マウスプレス、マウスリリース単独のイベントも発生しますが、マウスの一連の動作としてのプレス、リリースが完了した段階でアクティベートイベントが発生します。ボタンを使用するアプリケーションは、なるべくこの、アクティベートイベントを使用されることを推奨します。

58.5 プロパティー覧

クラス WSCvbtn のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNpushPixmap	押下画	
WSCbool	WSNenableFocusMove	フォーカス移動	1
	無	False (0)	
	有	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	

	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	

	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

58.6 トリガー一覧

WSCvbtn クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE

58.7 関数一覧

- void onActivate() :WSCvbtn
- void setDrawFocusBorder(WSCbool fl) :WSCvbtn

58.8 関数仕様

58.8.1 onActivate 関数の説明

書式 void onActivate()

機能 ボタンが押下された場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる代わりに、この関数をオーバーライドすることでも、ボタン押下に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項 なし。

サンプル

```
void new_class::onActivate(){
  //ボタンが押下された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

58.8.2 setDrawFocusBorder 関数の説明

書式 void setDrawFocusBorder(WSCbool fl)

機能 キーボードフォーカスがあたっている場合に描画されるフォーカスボーダーの描画属性フラグを設定します。

処理概要 False が指定されるとフォーカスボーダーが表示されなくなります。

引数

(in)WSCbool fl	表示属性

復帰値 なし。

サンプル

```
//フォーカスボーダーを描画しないように設定します。
newvbtn_000->setDrawFocusBorder(False);
```

注意事項

59 WSCvclock

59.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel

59.2 概要

時計を表示するオブジェクトです。年月日、時分、秒を表示ができます。

59.3 機能

・時計の表示機能

59.4 注意事項

59.5 プロパティ一覧

クラス WSCvclock のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNdateFormat	日付フォーマット	1
	表示しない	(0)	
	平成年月日	(1)	
	2001 年月日	(2)	
	01 年月日	(3)	
	H 13/1/1	(4)	
	2001/1/1	(5)	
	'01/1/1	(6)	
unsigned char	WSNweekFormat	曜日フォーマット	2
	表示しない	(0)	
	月曜日	(1)	
	(月)	(2)	
unsigned char	WSNclockFormat	時刻フォーマット	3
	表示しない	(0)	
	11 時分秒	(1)	
	23 時分秒	(2)	
	午後時分秒	(3)	
	11 00 00	(4)	
	23 00 00	(5)	
	AM 00 00	(6)	
WSCbool	WSNsecondOn	秒の表示	0
	不可視	False (0)	
	可視	True (1)	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0

	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	

	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

59.6 トリガー一覧

WSCvclock クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

59.7 関数一覧

59.8 関数仕様

60 WSCvdrawingArea

60.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

60.2 概要

自由に描画のできるエリアを提供します。WSEV_EXPOSE イベントで、イベントプロシージャを作成します。そのプロシージャで、各種描画関数を使って、自由に描画します。

60.3 機能

- WSEV_EXPOSE イベントでの描画機能
- 描画メソッドを提供

60.4 注意事項

EXPOSE トリガで、描画するイベントプロシージャ内で、ダイアログを popup( ) 関数を呼び出してはいけません。呼び出した以降の描画が行われなくなります。

60.5 プロパティー覧

クラス WSCvdrawingArea のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

60.6 トリガー覧

WSCvdrawingArea クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

60.7 関数一覧

- long setForeColor(char*) :WSCvdrawingArea
- long setBackColor(char*) :WSCvdrawingArea
- long setLineWidth(short) :WSCvdrawingArea
- long setLineDashType(char) :WSCvdrawingArea
- long setHatchPattern(char) :WSCvdrawingArea
- long setRegion(short x,short y,unsigned short w,unsigned short h) :WSCvdrawingArea
- long drawArc(short x,short y,unsinged short w,unsinged short h,short a1,short a2) :WSCvdrawingArea
- long drawFillArc(short x,short y,unsinged short w,unsinged short h,short a1,short a2,char kind) :WSCvdrawingArea
- long drawLine(short x,short y,short x2,short y2) :WSCvdrawingArea

- long drawLines(WSCpoint*,long num) :WSCvdrawingArea
- long drawRect(short x,short y,unsinged short w,unsiged short h) :WSCvdrawingArea
- long drawFillRect(short x,short y,unsinged short w,unsiged short h) :WSCvdrawingArea
- long drawRects(WSCrect*,long num) :WSCvdrawingArea
- long drawFillRects(WSCrect*,long num) :WSCvdrawingArea
- long drawPoly(WSCpoint*,long num) :WSCvdrawingArea
- long drawFillPoly(WSCpoint*,long num) :WSCvdrawingArea
- long drawImage(short,short,WSCushort,WSCushort,WSDimage*,char) :WSCvdrawingArea
- long drawImage(short,short,WSCushort,WSCushort,char*,char) :WSCvdrawingArea
- long drawStretchedImage(short,short,WSCushort,WSCushort,WSDimage*) :WSCvdrawingArea
- long drawStretchedImage(short,short,WSCushort,WSCushort,char*) :WSCvdrawingArea
- long drawString(long,long,WSCulong,WSCulong,char,char,char*,long) :WSCvdrawingArea
- long drawFillString(long,long,WSCulong,WSCulong,char,char,char*,long) :WSCvdrawingArea

60.8 関数仕様

60.8.1 setForeColor 関数の説明

書式 long setForeColor(char* cname)

機能 描画色を色名称で指定します。

処理概要 drawLine 等のメソッドで用いる描画色を指定します。

引数

(in)char* cname	色名称

色名称は、次のフォーマットで指定します。
#RRGGBB
RR: ００～ｆｆの範囲で１６進数で赤の輝度を指定します。
GG: ００～ｆｆの範囲で１６進数で緑の輝度を指定します。
BB: ００～ｆｆの範囲で１６進数で青の輝度を指定します。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項

サンプル

```
#include <WSCvdrawingArea.h>
void sample_proc(WSCbase* object){
  WSCvdrawingArea* drawing_a =(WSCvdrawingArea*)object->cast("WSCvdrawingArea");
  if (drawing_a == NULL){
    return;
  }

  //表示色の設定と矩形の描画
  drawing_a->setForeColor("#ffff00");
  drawing_a->drawRect(150,50,100,100);

  //表示色の設定と塗りつぶし矩形の描画
  drawing_a->setForeColor("#00ff00");
  drawing_a->drawFillRect(20,100,50,50);

  //表示色の設定と円弧の描画
  drawing_a->setForeColor("#0000ff");
  drawing_a->drawArc(50,200,50,50,60*64,270*64);

  //表示色の設定とハッチパターンによる塗りつぶし円弧(パイ)の描画
  drawing_a->setForeColor("#ff00ff");
  drawing_a->setHatchPattern(7);
  drawing_a->drawFillArc(150,200,100,50,60*64,270*64, 0 /*PIE*/ );

  //表示色の設定とハッチパターンによる塗りつぶし円弧の描画
  drawing_a->setForeColor("#ff00ff");
  drawing_a->setHatchPattern(6);
  drawing_a->drawFillArc(250,200,100,50,60*64,270*64, 1 /*CHORD*/ );

  //多角形の描画
  WSCpoint pt[3];
```

```
  pt[0].x = 300;
  pt[0].y = 300;
  pt[1].x = 330;
  pt[1].y = 330;
  pt[2].x = 300;
  pt[2].y = 330;
  drawing_a->setForeColor("#00ffff");
  drawing_a->setHatchPattern(0);
  drawing_a->drawFillPoly(pt,3);

  //表示色の設定と線の描画
  drawing_a->setForeColor("#ff0000");
  drawing_a->setLineWidth(2); //線幅 2ドット
  drawing_a->setLineType(1);  //点線
  drawing_a->drawLine(50,50,100,100);

  //文字列の描画
  drawing_a->setForeColor("#000000");
  drawing_a->drawString(100,300,100,30,0,WS_CENTER,"ABCDE1234");
  //背景付き文字列の描画
  drawing_a->setBackColor("#808080");
  drawing_a->drawFillString(100,330,100,30,0,WS_CENTER,"ABCDE1234");

  //リージョンによるクリッピング
  drawing_a->setRegion(70,70,80,80);
  drawing_a->drawLine(50,50,100,100); //70,70 - 80,80 が描画されます。
}
```

60.8.2 setBackColor 関数の説明

書式 long setBacKColor(char* cname)

機能 背景色を指定を色名称で指定します。

処理概要 drawFillRect 等のメソッドで用いる背景色を指定します。

引数

(in)char* cname	色名称

色名称は、次のフォーマットで指定します。
#RRGGBB
RR: ００～ｆｆの範囲で１６進数で赤の輝度を指定します。
GG: ００～ｆｆの範囲で１６進数で緑の輝度を指定します。
BB: ００～ｆｆの範囲で１６進数で青の輝度を指定します。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項

サンプル setForeColor() を参照してください。

60.8.3 setLineWidth 関数の説明

書式 long setLineWidth(short linewidth)

機能 描画する線の太さを指定します。

処理概要 drawLine 等のメソッドで用いる線幅を指定します。

引数

(in)short linewidth	線幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.4 setLineDashType 関数の説明

書式 long setLineDashType(char no)

機能 描画する線の点線属性を指定します。

処理概要 drawLine 等のメソッドで用いる線の点線属性を指定します。

引数

(in)char no	点線属性

点線属性に指定できる値には、次のようなものがあります。

属性値	意味
0	実線
1	鎖線
2	長鎖線
3	一点鎖線
4	二点鎖線
5	長一点鎖線
6	長二点鎖線
7	細点線

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項

サンプル setForeColor() を参照してください。

60.8.5 setHatchPattern 関数の説明

書式 long setHatchPattern(char no)

機能 塗りつぶし属性を指定します。

処理概要 drawRect 等のメソッドで使用される塗りつぶし属性を指定します。

引数

(in)char no	塗りつぶし属性

塗りつぶし属性に指定できる値には、次のようなものがあります。

属性値	意味
0	ベタ塗
1	左下斜線
2	右下斜線
3	縦線
4	横線
5	斜め交差
6	縦横交差
7	ドット

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

サンプル setForeColor() を参照してください。

注意事項 なし。

60.8.6 setRegion 関数の説明

書式 long setRegion(short x,short y,unsigned short w,unsigned short h);

機能 描画する領域を指定します。

処理概要 指定した領域以外は描画の対象から外します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.7 drawArc 関数の説明

書式 long drawArc(short x,short y,unsigned short w, unsigned short h,short a1,short a2);

機能 円弧、楕円弧を描画します。

処理概要 横直径 w、縦直径 h を持つ円または楕円の、X 軸右水平を 0 度とした反時計回りで、円弧開始角度、終了角度を指定します。このとき、64 倍した値を指定するのに注意してください。これは、1/64 度の精度確保するためです。完全な円、または円弧を描く場合は、それぞれ 0 * 64、360 * 64 を指定します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅
(in)short a1	円弧の開始角度
(in)short a2	円弧の終了角度

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.8 drawFillArc 関数の説明

書式 long drawFillArc(short x,short y,unsigned short w, unsigned short h,short a1,short a2,char kind);

機能 塗りつぶし円弧、楕円弧を描画します。

処理概要 横直径 w、縦直径 h を持つ円または楕円の、X 軸右水平を 0 度とした反時計回りで、円弧開始角度、終了角度を指定します。このとき、64 倍した値を指定するのに注意してください。これは、1/64 度の精度確保するためです。完全な円、または円弧を描く場合は、それぞれ 0 * 64、360 * 64 を指定します。円弧の種別を指定することによって、扇型の円弧、弓型の円弧を描画することができます。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅
(in)short a1	円弧の開始角度
(in)short a2	円弧の終了角度
(in)char kind	塗りつぶし円弧の種別

塗りつぶし種別に指定できる値には、次のようなものがあります。

属性値	意味
0	扇型
1	弓型

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.9 drawLine 関数の説明

書式 long drawLine(short x1,short y1,short x2,short y2);

機能 線を描画します。

処理概要 指定した座標 x1,y1 から x2,y2 まで線を描画します。

引数

(in)short x1	線の開始座標X
(in)short y1	線の開始座標Y
(in)short x2	線の終了座標X
(in)short y2	線の終了座標Y

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.10 drawLines 関数の説明

書式 long drawLines(WSCpoint* pt,long num);

機能 折れ線を描画します。

処理概要 指定した座標 X1,Y1 から Xn,Yn まで折れ線を描画します。

引数

(in)WSCpoint* pt	端点座標の配列
(in)long num	端点の数

WSCpoint は、メンバ x,y を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCpoint pt[3];
pt[0].x = 300;
pt[0].y = 300;
pt[1].x = 330;
pt[1].y = 330;
pt[2].x = 300;
pt[2].y = 330;
drawing_a->setForeColor("#00ffff");
drawing_a->drawLines(pt,3);
```

60.8.11 drawRect 関数の説明

書式 long drawRect(short x,short y,unsigned short w,unsigned short h);

機能 矩形を描画します。

処理概要 左上角を x,y とした横幅 w、縦幅 h の矩形を描画します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	矩形の横幅
(in)unsigned short h	矩形の縦幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.12 drawFillRect 関数の説明

書式 long drawFillRect(short x,short y,unsigned short w,unsigned short h);

機能 塗りつぶし矩形を描画します。

処理概要 左上角を x,y とした横幅 w、縦幅 h の塗りつぶし矩形を描画します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	矩形の横幅
(in)unsigned short h	矩形の縦幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.13 drawRects 関数の説明

書式 long drawRects(WSCrect* rect,long num);

機能 矩形を複数個描画します。

処理概要 WSCrect 配列で指定された複数の矩形を描画します。

引数

(in)WSCrect* rect	矩形の配列
(in)long num	端点の数

WSCrect は、メンバ x,y,width,height を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCrect rect[3];
rect[0].setRect(300,300,10,10);
rect[1].setRect(320,300,10,10);
```

```
rect［2］.setRect(340,300,10,10);
drawing_a->setForeColor("#00ffff");
drawing_a->drawRects(rect,3);
```

60.8.14 drawFillRects 関数の説明

書式 long drawFillRects(WSCrect* pt,long num);

機能 塗りつぶし矩形を複数個描画します。

処理概要 WSCrect 配列で指定された複数の塗りつぶし矩形を描画します。

引数

(in)WSCrect* rect	矩形の配列
(in)long num	端点の数

WSCrect は、メンバ x,y,width,height を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCrect rect［3］;
rect［0］.setRect(300,300,10,10);
rect［1］.setRect(320,300,10,10);
rect［2］.setRect(340,300,10,10);
drawing_a->setForeColor("#00ffff");
drawing_a->drawFillRects(rect,3);
```

60.8.15 drawPoly 関数の説明

書式 long drawPoly(WSCpoint* pt,long num);

機能 多角形を描画します。

処理概要 指定した座標 X1,Y1 から Xn,Yn までを結ぶ多角形を描画します。

引数

(in)WSCpoint* pt	端点座標の配列
(in)long num	端点の数

WSCpoint は、メンバ x,y を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCpoint pt［3］;
pt［0］.x = 300;
pt［0］.y = 300;
pt［1］.x = 330;
pt［1］.y = 330;
pt［2］.x = 300;
pt［2］.y = 330;
drawing_a->setForeColor("#00ffff");
drawing_a->drawPoly(pt,3);
```

60.8.16 drawFillPoly 関数の説明

書式 long drawFillPoly(WSCpoint* pt,long num);

機能 塗りつぶし多角形を描画します。

処理概要 指定した座標 X1,Y1 から Xn,Yn までを結ぶ塗りつぶし多角形を描画します。

引数

(in)WSCpoint* pt	端点座標の配列
(in)long num	端点の数

WSCpoint は、メンバ x,y を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

60.8.17 drawGradation 関数の説明

書式 long drawGradation(long type,short col1,short col2, short col3,short x,short y,WSCushort w,WSCushort h, WSCuchar grad_margin);

機能 グラデーションされた矩形を描画します。

処理概要 指定された３色を使用して、色１～色２～色３へとグラデーションされた矩形を描画します。

引数

(in)long type	グラデーションの種別
(in)short col1	色１
(in)short col2	色２
(in)short col3	色３
(in)short x	座標X
(in)short y	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)WSCuchar grad_margin	色２の幅

グラデーションされた矩形領域を指定し、グラデーションのマージンには、色２の幅をドット指定します。グラデーション種別に指定できる値には、次のようなものがあります。

グラデーション種別	意味
WS_GR_LT_RB	左上→右下
WS_GR_RT_LB	右上→左下
WS_GR_LB_RT	左下→右上
WS_GR_RB_LT	右下→左上
WS_GR_T_B	上→下
WS_GR_B_T	下→上
WS_GR_L_R	左→右
WS_GR_R_L	右→左

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
short cno1 = WSGIappColorSet()->getColorNo("#888888");
short cno2 = WSGIappColorSet()->getColorNo("#aaaaaa");
short cno3 = WSGIappColorSet()->getColorNo("#000000");
drawing_a->drawGradation(WS_GR_T_B,cno1,cno2,cno3,10,10,100,50,10);
```

60.8.18 drawImage 関数の説明

書式 long drawImage(short x,short y,WSCushort w,WSCushort h, WSDimage* img,char align);

機能 画像を表示します。

処理概要 指定された画像構造体を指定矩形内に描画します。

引数

(in)short x	座標X
(in)short y	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)WSDimage* img	表示させたい画像
(in)char align	アラインメント

アラインメントに指定できる値には、次のようなものがあります。

属性値	意味
WS_LEFT	左寄
WS_RIGHT	右寄
WS_CENTER	中央
WS_TOP	上寄
WS_BOTTOM	下寄
WS_LEFT_TOP	左上寄
WS_LEFT_BOTTOM	左下寄
WS_RIGHT_BOTTOM	右下寄
WS_RIGHT_TOP	右上寄

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCimage* image = WSGIappImageSet()->getImage("001.jpg");
drawing_a->drawImage(10,10,100,100,image,WS_CENTER);
```

60.8.19 drawImage 関数の説明

書式 long drawImage(short x,short y,WSCushort w,WSCushort h, char* img,char align);

機能 画像を表示します。

処理概要 指定された画像を指定矩形内に描画します。

引数

(in)short x	座標X
(in)short y	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)char* imgname	表示させたい画像のファイル名
(in)char align	アラインメント

アラインメントに指定できる値には、次のようなものがあります。

属性値	意味
WS_LEFT	左寄
WS_RIGHT	右寄
WS_CENTER	中央
WS_TOP	上寄
WS_BOTTOM	下寄
WS_LEFT_TOP	左上寄
WS_LEFT_BOTTOM	左下寄
WS_RIGHT_BOTTOM	右下寄
WS_RIGHT_TOP	右上寄

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
drawing_a->drawImage(10,10,100,100,"001.jpg",WS_CENTER);
```

60.8.20 drawStretchedImage 関数の説明

書式 long drawStretchedImage(short x,short y, WSCushort w,WSCushort h, WSDimage* img);

機能 画像を伸縮して表示します。

処理概要 指定された画像を、指定された矩形に伸縮して表示します。

引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)WSDimage* img	表示させたい画像

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCimage* image = WSGIappImageSet()->getImage("001.jpg");
drawing_a->drawStretchedImage(10,10,100,100,image);
```

60.8.21 drawStretchedImage 関数の説明

書式 long drawStretchedImage(short x,short y, WSCushort w,WSCushort h, char* img);

機能 画像を伸縮して表示します。

処理概要 指定された画像を、指定された矩形に伸縮して表示します。

引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)char* img	表示させたい画像ファイル名

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
drawing_a->drawStretchedImage(10,10,100,100,"001.jpg");
```

60.8.22 drawString 関数の説明

書式 long drawString(short x,short y,WSCushort w,WSCushort h, char font_no,char align,char* string, long encoding = WS_EN_DEFAULT);

機能 指定された文字列を指定された矩形内に表示します。

処理概要 指定された文字列をエンコーディングに従がって指定された矩形内に表示します。

引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)char font_no	フォント番号
(in)char align	アラインメント
(in)char* string	表示させたい文字列
(in)long encoding	文字列のエンコーディング

フォント番号には、0番からのフォント番号を指定します。アラインメントに指定できる値には、次のようなものがあります。

属性値	意味
WS_LEFT	左寄
WS_RIGHT	右寄
WS_CENTER	中央
WS_TOP	上寄
WS_BOTTOM	下寄
WS_LEFT_TOP	左上寄
WS_LEFT_BOTTOM	左下寄
WS_RIGHT_BOTTOM	右下寄
WS_RIGHT_TOP	右上寄

エンコーディングに指定できる値には、次のようなものがあります。省略すると WS_EN_DEFAULT が指定されます。

属性値	意味
WS_EN_DEFAULT	現在の設定を指定 (省略時の値)
WS_EN_LOCALE	現在の LANG 環境変数の設定を指定
WS_EN_NONE	設定を行わない
WS_EN_ISO8859_1	ISO8859(1) を指定
WS_EN_ISO8859_2	ISO8859(2) を指定
WS_EN_ISO8859_3	ISO8859(3) を指定
WS_EN_ISO8859_4	ISO8859(4) を指定
WS_EN_ISO8859_5	ISO8859(5) を指定
WS_EN_ISO8859_6	ISO8859(6) を指定
WS_EN_ISO8859_7	ISO8859(7) を指定
WS_EN_ISO8859_8	ISO8859(8) を指定
WS_EN_ISO8859_9	ISO8859(9) を指定
WS_EN_ISO8859_10	ISO8859(10) を指定
WS_EN_ISO8859_11	ISO8859(11) を指定
WS_EN_ISO8859_12	ISO8859(12) を指定
WS_EN_ISO8859_13	ISO8859(13) を指定
WS_EN_ISO8859_14	ISO8859(14) を指定
WS_EN_ISO8859_15	ISO8859(15) を指定
WS_EN_UTF8	UTF8 を指定
WS_EN_KOI8R	KOI8R を指定
WS_EN_EUCJP	EUCJP を指定
WS_EN_SJIS	SJIS を指定
WS_EN_EUCKR	EUCKR を指定
WS_EN_EUCCN	EUCCN を指定
WS_EN_BIG5	BIG5 を指定

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。
サンプル setForeColor() を参照してください。

60.8.23 drawFillString 関数の説明

書式 long drawFillString(short x,short y,WSCushort w,WSCushort h, char font_no,char align,char* string, long encoding = WS_EN_DEFAULT);
機能 指定された文字列を指定された矩形内に背景色付き表示します。
処理概要 指定された文字列をエンコーディングに従がって指定された矩形内に表示します。
引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)char font_no	フォント番号
(in)char align	アラインメント
(in)char* string	表示させたい文字列
(in)long encoding	文字列のエンコーディング

引数の詳細は、drawString の項を参照ください。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル setForeColor() を参照してください。

61 WSCverbList

61.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform WSCscrForm WSClist

61.2 概要

詳細表示を行うリストです。WSClist クラスの WSNtype プロパティを詳細表示にしたものと同じです。データソースを指定すると、ダイレクトに詳細リスト表示を行うことが出来ます。

アイコンのパス名, 第１項目, 第２項目, 第３項目, . . .
アイコンのパス名, 第１項目, 第２項目, 第３項目, . . .
. . .

アイコンのパス名は省略可能です。省略すると、WSNiconPixmap で指定されたものが使用されます。

61.3 機能

- リストの詳細表示機能
- データソース指定によるデータの直接指定

61.4 注意事項

61.5 プロパティ一覧

クラス WSCverbList のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値

unsigned char	WSNitemHeight	項目縦幅	20
short	WSNiconPixmap	アイコン画	$(WSDIR)/sys/pixmaps/bi17.xpm
WSCbool	WSNuseIcon	アイコン使用	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNfont	フォント番号	0
short	WSNselectColor	選択色	DEF7
short	WSNselectForeColor	選択表示色	DEF18
WSCbool	WSNmultiSelect	複数選択	0
WSCbool	WSNreverseSelect	選択時反転	1
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNtitleHeight	タイトル縦幅	20
char*	WSNtitleString	タイトル文字列	title
char*	WSNbarValue	バー位置	20
char*	WSNdata	データ	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNenableInput	入力可否	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNseparator	セパレータ	,
unsigned short	WSNbarThickness	バー幅	16
unsigned short	WSNworkWidth	仮想領域横幅	1000
unsigned short	WSNworkHeight	仮想領域縦幅	1000
unsigned short	WSNhbarValue	横スクロール位置	0
unsigned short	WSNvbarValue	縦スクロール位置	0
unsigned short	WSNincrement	ボタンスクロール単位	10
unsigned short	WSNpageIncrement	ページスクロール単位	100
WSCbool	WSNhbarVisible	横スクロールバー	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNvbarVisible	縦スクロールバー	1
	不可視	False (0)	
	可視	True (1)	
short	WSNworkBackColor	作業領域背景色	DEF11
short	WSNbarShadowColor	バー背景色	DEF12
unsigned short	WSNscrollWidth	横スクロール単位	30
unsigned char	WSNmargin	マージン	4
WSCbool	WSNvirtualScroll	仮想スクロール	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNframe	フレーム	1
	不可	False (0)	
	可	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	

	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

61.6 トリガー一覧

WSCverbList クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
SORT	序列変更となった場合	WSEV_SORT
ITEM-SELECTED	項目が選択された場合	WSEV_ITEM_SELECTED
INPUT-FIXED	入力が確定状態になった場合	WSEV_INPUT_FIXED
SCR-MOUSE-PRESS	スクロールエリア内でマウスのボタンが押された場合	WSEV_SCR_MOUSE_PRESS
SCR-MOUSE-RELEASE	スクロールエリア内でマウスのボタンがはなされた場合	WSEV_SCR_MOUSE_RELEASE
SCR-MOUSE-MOVE	スクロールエリア内でマウスポインタが動いた場合	WSEV_SCR_MOUSE_MOVE

61.7 関数一覧

61.8 関数仕様

62 WSCvertForm

62.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCform

62.2 概要

配列機能を持ったフォームです。このフォームは、このフォームに配置されたインスタンスを縦に並べます。サイズが縮められた場合、プロパティ WSNminimum で指定されたサイズまで縮められます。

62.3 機能

・インスタンスを縦に並べます。

62.4 注意事項

62.5 プロパティー覧

クラス WSCvertForm のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNminimum	最小値	20
unsigned char	WSNmargin	マージン	4
unsigned char	WSNalignmentH	横配置	0
	なし	WS_LEFT (0)	
	左配置	WS_RIGHT (1)	
	中央配置	WS_CENTER (2)	
	右配置	(3)	
	アジャスト	(4)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	100
short	WSNy	Y座標	100
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	2
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
unsigned short	WSNanchorTop	上アンカー	0

unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

62.6 トリガー一覧

WSCvertForm クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

62.7 関数一覧

62.8 関数仕様

63 WSCvfbtn

63.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel WSCvbtn

63.2 概要

マウスが領域内に入ると浮きあがって表示されるボタンです。そのほかの機能は、ボタンオブジェクトと同等です。

63.3 機能

- マウス動作による浮き上がり表示

63.4 注意事項

63.5 プロパティ一覧

クラス WSCvfbtn のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNpushPixmap	押下画	
WSCbool	WSNenableFocusMove	フォーカス移動	1
	無	False (0)	
	有	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	

WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

63.6 トリガー一覧

WSCvfbtn クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE

63.7 関数一覧

63.8 関数仕様

64 WSCvgraphMatrix

64.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

64.2 概要

グラフ用の格子を表示します。

64.3 機能

・縦・横格子の表示

64.4 注意事項

64.5 プロパティー覧

クラス WSCvgraphMatrix のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNcolumns	カラム数	11
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	

	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

64.6 トリガー一覧

WSCvgraphMatrix クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

64.7 関数一覧

64.8 関数仕様

65 WSCvgraphScale

65.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

65.2 概要

グラフの目盛を表示するオブジェクトです。プロパティ WSNorientation により縦横を切替えることができます。プロパティ WSNvalue に、目盛に用いられる文字列をカンマで区切って設定します。

`WSNvalue` 値の設定:
`1,2,3,4,5,6,7,...`
のように値を設定します。

また、プロパティ WSNbarThickness には、目盛の長さをカンマで区切って設定します。

`WSNbarThickness` 値の設定:
`20,10,10,10,10,20,10,10,10,...`
のように値を設定します。

65.3 機能

- グラフの目盛の表示
- 縦横切替え機能

65.4 注意事項

65.5 プロパティ一覧

クラス WSCvgraphScale のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNvalue	値	0,1,2,3,4,5,6,7,8,9,10
unsigned short	WSNcolumns	カラム数	11
char*	WSNbarThickness	バー幅	10,5,5,5,5,10,5,5,5,5,10
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
WSCbool	WSNreverseFlag	反転表示有無	0
	False	False (0)	
	True	True (1)	
unsigned char	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
unsigned char	WSNmargin	マージン	10
unsigned char	WSNfont	フォント番号	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	

	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

65.6 トリガー一覧

WSCvgraphScale クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

65.7 関数一覧

65.8 関数仕様

66 WSCvifield

66.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel

66.2 概要

キーボードによる単行のテキスト入力フィールドです。

66.3 機能

- キーボードからのテキスト入力機能
- 日本語入力機能

66.4 注意事項

プロパティ WSNlabelString はデータソース対象プロパティとして用いられます。

66.5 プロパティ一覧

クラス WSCvifield のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNcursorPos	カーソル位置	0
WSCbool	WSNinterCur	挿入モード	1
	不可	False (0)	
	可	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
char*	WSNreturn	リターンキー移動先	
WSCbool	WSNenableFocusMove	フォーカス移動	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNfillSpace	フィルスペース	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNcursorAdjust	カーソル補正	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNifieldSkipMode	フィールド間連係	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNnoHatch	活性表示維持	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNreturnKeyFocus	リターンキーフォーカス	0
	不可	False (0)	
	可	True (1)	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNdisplayOnly	表示のみ	0
	オフ	False (0)	
	オン	True (1)	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF11
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	

short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	

	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

66.6 トリガー一覧

WSCvifield クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
KEY-HOOK	キーイベントの横取りを行う場合	WSEV_KEY_HOOK

66.7 関数一覧

- void addString(char*) :WSCvifield
- WSCushort* getBuf() :WSCvifield
- void setBuf(WSCushort* buf) :WSCvifield
- void onKey(WSDkeyboard* keyboard,WSCbool fl) :WSCvifield
- void onKeyHook(WSDkeyboard* keyboard) :WSCvifield
- void onActivate() :WSCvifield
- void onValueChange() :WSCvifield
- void replaceSelectedString() :WSCvifield
- WSCstring getSelectedString() :WSCvifield
- WSCstring getString() :WSCvifield
- void deleteSelectedString() :WSCvifield
- long setSelect(long pos,long len) :WSCvifield
- long getSelectedPos() :WSCvifield

66.8 関数仕様

66.8.1 addString 関数の説明

書式 void addString(char* var)

機能 現在表示中の文字列に指定した文字列を追記します。

処理概要

引数

(in)char* var	追加したい文字列

復帰値 なし。

注意事項 文字列を置き換えたい場合は、プロパティ WSNlabelString を用いてください。

```
//現在表示中の文字列に指定された文字列を追加します。
newvifi_000->addString("文字列の追加");
```

66.8.2 getBuf 関数の説明

書式 WSCushort* getBuf()

機能 内部のテキストバッファを返します。

処理概要

引数 なし。

復帰値 テキストバッファ。

注意事項 返されたバッファを開放してはいけません。また、キー入力が行われ、バッファの内容が変更されると、領域は無効となります。したがって、この関数が返す領域の永続的な利用は避け、なるべく、他の領域にコピーしてください。このバッファの内容は、UCS2 コードです。

サンプル

```
//現在表示中の文字列の UCS2 文字列バッファを取得します。
WSCushort* buf = newvifi_000->getBuf();
```

66.8.3 setBuf 関数の説明

書式 void setBuf(WSCushort* buf)

機能 WSCvifield 内のデータ管理バッファに UCS2 文字列のデータを直接設定します。

処理概要

引数

(in)WSCushort* buf	設定したい UCS2 文字列

復帰値 なし。

注意事項 指定されたバッファをそのまま内部用のバッファとして利用しますので、指定後にそのバッファを開放しないでください。

サンプル

```
//UCS2 文字列を生成。
WSCushort buf = WSGFgetUCS2("文字列");
//UCS2 文字列バッファを直接指定します。
newvifi_000->setBuf(buf); // buf は、そのまま内部のバッファとして使用。
```

66.8.4 onKey() 関数の説明

書式 void onKey(WSDkeyboard* keyboard,WSCbool keydown);

機能 キーボードが押下または離された場合に onKey( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_KEY_PRESS/WSEV_KEY_RELEASE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、キーボード入力に関するイベント処理を行うことができます。

引数

(in)WSDkeyboard* keyboard	キーボードインスタンス
(in)WSCbool keydown	True = Press、False = Release

復帰値 なし。

注意事項

サンプル

```
void new_class::onKey(WSDkeyboard* keyboard,WSCbool keydown){
  //キー入力された場合に呼び出されます。
  if (keydown != False){   //キー押下
    //キーの取得
    long key = keyboard->getKey();
    //入力文字列の取得
    WSCstring str = keyboard->getText();
  }

  //処理を派生元クラスに引き継ぎます。
  old_class::onKey(keyboard,keydown);
}
```

66.8.5 onKeyHook() 関数の説明

書式 void onKeyHook(WSDkeyboard* keyboard);

機能 キーボードをフックします。キーボード入力が行われた場合、onKeyHook( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_KEY_HOOK) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、キーボード入力に関するイベント処理を行うことができます。

引数

(in)WSDkeyboard* keyboard	キーボード インスタンス

復帰値 なし。

注意事項

サンプル

```
void new_class::onKeyHook(WSDkeyboard* keyboard){
  //キー入力された場合に呼び出されます。
  //入力文字列の取得
  WSCstring str = keyboard->getText();
  //keyboard->setText(...) で、文字列を置き換えることで、
  //文字列入力を操作することができます。

  //処理を派生元クラスに引き継ぎます。
  old_class::onKeyHook(keyboard);
}
```

66.8.6 onActivate() 関数の説明

書式 void onActivate()

機能 リターンキーが入力された場合に、ACTIVATE が発生します。その場合、onActivate( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、リターンキーによる ACTIVATE に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onActivate(){
  //リターンキーが入力された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

66.8.7 onValueChange() 関数の説明

書式 void onValueChange()

機能 キー入力された場合に、VALUE-CH が発生します。その場合、onValueChange( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ (WSEV_VALUE_CH) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、キー入力に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onValueChange(){
  //入力が行われ、文字列が変更された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onValueChange();
}
```

66.8.8 replaceSelectedString() 関数の説明

書式 void replaceSelectedString(char* str,long encoding = WS_EN_DEFAULT);

機能 選択された状態の文字列を指定された文字列で置き換えます。

処理概要

引数

(in)char* str	置換後の文字列
(in)long encoding	文字列のエンコーディング

エンコーディングは、省略すると WS_EN_DEFAULT が指定されます。指定できるエンコーディングは、WSCstring(char*,long) を参照ください。

復帰値 なし。

注意事項

サンプル

```
//現在選択状態にある文字列を指定した文字列で置換します。
newvifi_000->replaceSelectedString("置換文字列");
```

66.8.9 getSelectedString() 関数の説明

書式 WSCstring getSelectedString()

機能 選択された状態の文字列を取得します。

処理概要

引数 なし。

復帰値 選択文字列

注意事項

サンプル

```
//現在選択状態にある文字列を取得します。
WSCstring stext = newvifi_000->getSelectedString();
```

66.8.10 getString() 関数の説明

書式 WSCstring getString()

機能 入力文字列を取得します。

処理概要

引数 なし。

復帰値 入力文字列

注意事項

サンプル

```
//現在の入力文字列を取得します。
WSCstring text = newvifi_000->getString();
```

66.8.11 deleteSelectedString() 関数の説明

書式 void deleteSelectedString()

機能 選択文字列を削除します。

処理概要 選択された文字列の部分だけ削除します。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
//現在選択状態にある文字列を削除します。
newvifi_000->deleteSelectedString();
```

66.8.12 setSelect() 関数の説明

書式 long setSelect(long pos,long len)

機能 指定された位置の文字列を選択状態にします。

処理概要 文字位置 pos から指定された長さの文字列を選択状態にします。

引数

(in)long pos	文字列開始位置
(in)long len	文字列長

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
//指定した位置の文字列を選択状態にします。
//例えば先頭から５文字目までを選択状態にするには次のようにします。
newvifi_000->setSelectedString(0,5);
```

66.8.13 getSelectedPos() 関数の説明

書式 long setSelectedPos()
機能 選択文字列の先頭からの文字数を取得します。
処理概要
引数 なし。
復帰値 先頭からの文字数
注意事項
サンプル

```
//選択状態にある文字列の先頭からの文字数を取得します。
long pos = newvifi_000->setSelectedPos();
```

67 WSCvkslabel

67.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel WSCvifield

67.2 概要

入力フィールドにおいて、キーボード入力、編集機能のみを抑制したクラスです。文字列の編集はできませんが、カーソルキーによる文字列の選択や、マウスによる文字列の選択等が行えます。

67.3 機能

- 入力フィールドの編集機能の抑制
- カーソルキーやマウスによる文字列の選択

67.4 注意事項

67.5 プロパティ一覧

クラス WSCvkslabel のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNcursorPos	カーソル位置	0
WSCbool	WSNinterCur	挿入モード	1
	不可	False (0)	
	可	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
char*	WSNreturn	リターンキー移動先	
WSCbool	WSNenableFocusMove	フォーカス移動	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNfillSpace	フィルスペース	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNcursorAdjust	カーソル補正	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNifieldSkipMode	フィールド間連係	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNnoHatch	活性表示維持	0
	不可	False (0)	
	可	True (1)	

WSCbool	WSNreturnKeyFocus	リターンキーフォーカス	0
	不可	False (0)	
	可	True (1)	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNdisplayOnly	表示のみ	0
	オフ	False (0)	
	オン	True (1)	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	

	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

67.6 トリガー一覧

WSCvkslabel クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
KEY-HOOK	キーイベントの横取りを行う場合	WSEV_KEY_HOOK

67.7 関数一覧

67.8 関数仕様

68 WSCvlabel

68.1 継承元

次のオブジェクトを継承しています。

WSCbase WSCnwbase

68.2 概要

テキストを表示します。

68.3 機能

- テキスト表示
- ピクスマップの表示
- 枠の表示

68.4 注意事項

68.5 プロパティ一覧

クラス WSCvlabel のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	

	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

68.6 トリガー一覧

WSCvlabel クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

68.7 関数一覧

- char getAlignment() :WSCvlabel

68.8 関数仕様

68.8.1 getAlignment 関数の説明

書式 char getAlignment()

機能 テキストのアラインメントを返します。

処理概要 プロパティ WSNalingmentH 、WSNalignmentV の併せた値を返します。

引数 なし。

復帰値 アラインメント。

WS_TOP	上そろえ
WS_LEFT_TOP	左上そろえ
WS_RIGHT_TOP	右上そろえ
WS_CENTER	中央そろえ
WS_LEFT	左そろえ
WS_RIGHT	右そろえ
WS_BOTTOM	下そろえ
WS_LEFT_BOTTOM	左下そろえ
WS_RIGHT_BOTTOM	右下そろえ

注意事項 なし。

サンプル

```
//表示文字列のアラインメントを取得します。
char alignment = newvlab->getAlignment();
```

69 WSCvline

69.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

69.2 概要

折れ線を表示します。

69.3 機能

- 折れ線の表示

69.4 注意事項

69.5 プロパティ一覧

クラス WSCvline のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
char*	WSNlineX	線のX座標	0,10,20,30,40,50,60,70,80,90,100
char*	WSNlineY	線のY座標	0,80,20,60,10,70,40,90,50,30,100
unsigned char	WSNlineNum	端点数	11
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0

不可	False (0)
可	True (1)

69.6 トリガー一覧

WSCvline クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE

69.7 関数一覧

69.8 関数仕様

70 WSCvlineGraph

70.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

70.2 概要

折れ線グラフを表示します。プロパティ WSNvalue に、グラフ表示したいデータをカンマで区切って設定します。

`WSNvalue` 値の設定:
`10,20,30,40,50,60,70,...`
のように値を設定します。

70.3 機能

- 折れ線グラフの表示
- 領域の塗りつぶし表示
- データソース指定によるデータの直接設定機能

70.4 注意事項

プロパティ WSNvalue はデータソース対象プロパティとして用いられます。WSNdataSource プロパティ(データソースタイプ) がインスタンスの場合、WSNdataSourceName プロパティに指定された値は、インスタンス名として扱われ、その指定されたインスタンスのデータソース対象プロパティから値が取得されます。また、ファイルの場合は、ファイル名として扱われ、ファイルからデータが読み込まれます。

70.5 プロパティー一覧

クラス WSCvlineGraph のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNvalue	値	0,30,20,40,10,50,10,80,50,70
unsigned short	WSNcolumns	カラム数	10
unsigned char	WSNbarThickness	バー幅	10
unsigned char	WSNorientation	表示方向	1

	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
WSCbool	WSNreverseFlag	反転表示有無	0
	False	False (0)	
	True	True (1)	
unsigned char	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	

	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

70.6 トリガー一覧

WSCvlineGraph クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

70.7 関数一覧

70.8 関数仕様

71 WSCvmeter

71.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvpolyAttr

71.2 概要

０～１００間の値で、バーによるメータ表示します。開始値と終了値を指定できます。図の例のメータでは、左が開始値で０、右が終了値で６０を指しています。

71.3 機能

- バー型のメータ表示機能

71.4 注意事項

71.5 プロパティー覧

クラス WSCvmeter のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNvalue1	値1	80

unsigned char	WSNvalue2	値2	20
WSCbool	WSNorientation	表示方向	0
	横方向	False (0)	
	縦方向	True (1)	
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
short	WSNhatchBlinkColor	ハッチパターンブリンク色	DEF13
WSCbool	WSNbackColorFlag	背景色有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackColor	背景色	DEF13
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0

unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

71.6 トリガー一覧

WSCvmeter クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE

71.7 関数一覧

71.8 関数仕様

72 WSCvmifield

72.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel WSCvifield

72.2 概要

複数行のテキスト入力フィールドです。

72.3 機能

・キーボードからの複数行のテキスト入力の提供
・日本語入力機能

72.4 注意事項

72.5 プロパティー一覧

クラス WSCvmifield のプロパティー一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNcursorPos	カーソル位置	0
WSCbool	WSNcursorAdjust	カーソル補正	1
	不可	False (0)	
	可	True (1)	

WSCbool	WSNnoHatch	活性表示維持	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNreturnKeyFocus	リターンキーフォーカス	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNdisplayOnly	表示のみ	0
	オフ	False (0)	
	オン	True (1)	
char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF11
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14

WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

72.6 トリガー一覧

WSCvmifield クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
KEY-HOOK	キーイベントの横取りを行う場合	WSEV_KEY_HOOK

72.7 関数一覧

72.8 関数仕様

73 WSCvpifield

73.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel WSCvifield

73.2 概要

パスワードの入力等に使用する、入力文字列を隠蔽した入力フィールドです。

73.3 機能

・入力文字列の隠蔽機能

73.4 注意事項

73.5 プロパティ一覧

クラス WSCvpifield のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
short	WSNcursorPos	カーソル位置	0
WSCbool	WSNinterCur	挿入モード	1
	不可	False (0)	
	可	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
char*	WSNreturn	リターンキー移動先	
WSCbool	WSNenableFocusMove	フォーカス移動	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNfillSpace	フィルスペース	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNcursorAdjust	カーソル補正	1
	不可	False (0)	
	可	True (1)	
WSCbool	WSNifieldSkipMode	フィールド間連係	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNnoHatch	活性表示維持	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNreturnKeyFocus	リターンキーフォーカス	0
	不可	False (0)	
	可	True (1)	
unsigned char	WSNdataSource	データソース	0
	無し	(0)	
	インスタンス	(1)	
	ファイル	(2)	
char*	WSNdataSourceName	データソース名	
WSCbool	WSNdisplayOnly	表示のみ	0
	オフ	False (0)	
	オン	True (1)	

char	WSNshadowType	影タイプ	3
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF11
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0

	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

73.6 トリガー一覧

WSCvpifield クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE
KEY-HOOK	キーイベントの横取りを行う場合	WSEV_KEY_HOOK

73.7 関数一覧

- void setMaskChar() :WSCvpifield

73.8 関数仕様

73.8.1 setMaskChar 関数の説明

書式 void setMaskChar(WSCstring mask);

機能 入力時に表示するマスク文字を指定します。

処理概要

引数

(in)WSCstring mask	マスク文字列

復帰値 なし。

注意事項 なし。

サンプル

```
//マスク文字を設定します。
WSCstring str;
str = "*";
newvpif_000->setMaskChar(str);
```

74 WSCvpoly

74.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvpolyAttr

74.2 概要

多角形を表示するオブジェクトです。

74.3 機能

- 多角形の表示機能
- ハッチパターンによる塗りつぶし機能

74.4 注意事項

74.5 プロパティ一覧

クラス WSCvpoly のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNlineX	線のX座標	0,100,100,0
char*	WSNlineY	線のY座標	0,0,100,100
unsigned char	WSNlineNum	端点数	4
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
short	WSNhatchBlinkColor	ハッチパターンブリンク色	DEF13
WSCbool	WSNbackColorFlag	背景色有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackColor	背景色	DEF13
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0

	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

74.6 トリガー一覧

WSCvpoly クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE

74.7 関数一覧

74.8 関数仕様

75 WSCvpolyAttr

75.1 継承元

次のオブジェクトを継承しています。

WSCbase WSCnwbase

75.2 概要

ハッチパターンによる塗りつぶし、線の幅、ブリンクのタイプなどを保持するプロパティを定義するサブクラスです。

75.3 機能

- パッチパターンによる塗りつぶしプロパティの定義
- 線幅プロパティの定義
- ブリンクタイププロパティの定義

75.4 注意事項

75.5 プロパティ一覧

クラス WSCvpolyAttr のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
short	WSNhatchBlinkColor	ハッチパターンブリンク色	DEF13
WSCbool	WSNbackColorFlag	背景色有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackColor	背景色	DEF13
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30

short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

75.6 トリガー一覧

WSCvpolyAttr クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE

75.7 関数一覧

75.8 関数仕様

76 WSCvradio

76.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel WSCvtoggle

76.2 概要

機能的には、トグルボタンと変わりませんが、デフォルトでチェックイメージを表示します。

76.3 機能

・チェックの表示

76.4 注意事項

デフォルトでは択一属性プロパティはオフににになっており、チェックボタンとして動作します。択一属性をオンに設定すると、ラジオボタンとして動作します。

76.5 プロパティー覧

クラス WSCvradio のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNselectString	選択時文字列	
unsigned char	WSNindicatorSize	インジケータ SIZE	16
short	WSNindicatorColor	インジケータ色	DEF11
short	WSNselectColor	選択色	DEF7
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNselectReset	選択時選択解除	0
	不可	False (0)	
	可	True (1)	
short	WSNid	ＩＤ	0
WSCbool	WSNenableFocusMove	フォーカス移動	1
	不可	False (0)	
	可	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0

short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

76.6 トリガー一覧

WSCvradio クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE

76.7 関数一覧

76.8 関数仕様

77 WSCvrect

77.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvpolyAttr

77.2 概要

矩形表示を行うオブジェクトです。

77.3 機能

- 矩形の表示
- ハッチパターンによる塗りつぶし表示

77.4 注意事項

77.5 プロパティ一覧

クラス WSCvrect のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	0
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0

	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF13
short	WSNhatchBlinkColor	ハッチパターンブリンク色	DEF13
WSCbool	WSNbackColorFlag	背景色有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackColor	背景色	DEF13
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	
	文字/背景	WS_BOTH (2)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

77.6 トリガー一覧

WSCvrect クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE

77.7 関数一覧

77.8 関数仕様

78 WSCvscrBar

78.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

78.2 概要

スクロールバーオブジェクトです。

78.3 機能

- マウスによるバーの移動
- バー、スクロール長の設定機能
- スクロール、ページスクロール単位の設定機能

78.4 注意事項

78.5 プロパティー一覧

クラス WSCvscrBar のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNbarShadowColor	バー背景色	DEF12
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
unsigned long	WSNsliderSize	スライダーサイズ	100
unsigned long	WSNmaximum	最大値	1000
unsigned short	WSNincrement	ボタンスクロール単位	10
unsigned short	WSNpageIncrement	ページスクロール単位	30
unsigned long	WSNvalue	値	0
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
short	WSNbackPixmap	背景画	
WSCbool	WSNgradation	グラデーション	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	

WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

78.6 トリガー一覧

WSCvscrBar クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
BEGIN	スクロールバー等、先頭へ移動となった場合	WSEV_BEGIN
END	スクロールバー等、末尾へ移動となった場合	WSEV_END
INCREMENT	スクロールバー等、値が増した場合	WSEV_INCREMENT
DECREMENT	スクロールバー等、値が減った場合	WSEV_DECREMENT
PAGE-INCREMENT	スクロールバー等、値が１ページ分増した場合	WSEV_PAGE_INCREMENT
PAGE-DECREMENT	スクロールバー等、値が１ページ分減った場合	WSEV_PAGE_DECREMENT

78.7 関数一覧

- virtual void onIncrement() :WSCvscrBar
- virtual void onPageIncrement() :WSCvscrBar
- virtual void onDecrement() :WSCvscrBar
- virtual void onPageDecrement() :WSCvscrBar
- virtual void onBegin() :WSCvscrBar
- virtual void onEnd() :WSCvscrBar
- WSCulong getMaxSlideValue() :WSCvscrBar

78.8 関数仕様

78.8.1 onIncrement() 関数の説明

書式 void onIncrement()

機能 スクロールバーが増方向に移動した場合、INCREMENT イベントが発生し onIncrement( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ WSEV_INCREMENT によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
void new_class::onIncrement(){
  //スクロール値が増えた場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onIncrement();
}
```

78.8.2 onPageIncrement() 関数の説明

書式 void onPageIncrement()
機能 スクロールバーがページ移動量分、増方向に移動した場合、INCREMENT イベントが発生し onPageIncrement( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_PAGE_INCREMENT によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
void new_class::onPageIncrement(){
  //スクロール値がページ移動量分増えた場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onPageIncrement();
}
```

78.8.3 onDecrement() 関数の説明

書式 void onDecrement()
機能 スクロールバーが減方向に移動した場合、DECREMENT イベントが発生し onDecrement( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_DECREMENT によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
void new_class::oneDecrement(){
  //スクロール値が減った場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onDecrement();
}
```

78.8.4 onPageDecrement() 関数の説明

書式 void onPageDecrement()
機能 スクロールバーがページ移動量分、減方向に移動した場合、DECREMENT イベントが発生し onPageDecrement( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_PAGE_DECREMENT によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。

注意事項

サンプル

```
void new_class::onPageDecrement(){
  //スクロール値がページ移動量分減った場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onPageDecrement();
}
```

78.8.5 onBegin() 関数の説明

書式 void onBegin()

機能 先頭移動スクロールボタンが押下されると BEGIN イベントが発生し onBegin( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ WSEV_BEGIN によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、先頭移動に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onBegin(){
  //先頭移動スクロールボタンが押下された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onBegin();
}
```

78.8.6 onEnd() 関数の説明

書式 void onEnd()

機能 末尾移動スクロールボタンが押下されると END イベントが発生し onEnd( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ WSEV_END によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、末尾移動に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onEnd(){
  //末尾移動スクロールボタンが押下された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onEnd();
}
```

78.8.7 onActivate() 関数の説明

書式 void onActivate()

機能 スクロールバーやスクロールボタンが放たれスクロール動作が完了した場合 ACTIVATE イベントが発生し onActivate( ) 関数が実行されます。

処理概要 アプリケーションは、トリガ WSEV_ACTIVATE によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動完了に関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項

サンプル

```
void new_class::onActivate(){
  //一連のスクロール動作が完了した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

78.8.8 onValueChange() 関数の説明

書式 void onValueChange()
機能 スクロール位置が変化した場合、VALUE-CH イベントが発生し onValueChange( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_VALUE_CH によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
void new_class::onValueChange(){
  //値が変化した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onValueChange();
}
```

78.8.9 getMaxSlideValue() 関数の説明

書式 WSCulong getMaxSlideValue()
機能 スクロール位置の最大値を取得します。
処理概要 スクロール位置の最大値は、WSNmaximum - WSNsliderSize となります。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
//スクロール位置の最大値を取得します。
WSCulong scrw = newvscr_000->getMaxSlideValue();
```

79 WSCvslider

79.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase

79.2 概要

値をマウスでスライドすることで指定するオブジェクトです。最大値、最小値を指定することができ、その間の数値を選択します。

79.3 機能

・スライダによる値の選択機能
・最大値、最小値の設定機能

79.4 注意事項

79.5 プロパティー覧

クラス WSCvslider のプロパティー覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
unsigned short	WSNsliderSize	スライダーサイズ	30

short	WSNmaximum	最大値	1000
short	WSNminimum	最小値	0
unsigned short	WSNdragInterval	ドラッグ間隔	30
short	WSNvalue	値	0
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0
unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

79.6 トリガー一覧

WSCvslider クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT

79.7 関数一覧

- virtual void onActivate() :WSCvscrBar
- virtual void onValueChange() :WSCvscrBar

79.8 関数仕様

79.8.1 onActivate() 関数の説明

書式 void onActivate()

機能 スライダーのスクロール動作が完了した場合 ACTIVATE イベントが発生し onActivate( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_ACTIVATE によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動完了に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
void new_class::onActivate(){
  //一連のスライダーの動作が完了した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

79.8.2 onValueChange() 関数の説明

書式 void onValueChange()
機能 スライダー値が変化した場合、VALUE-CH イベントが発生し onValueChange( ) 関数が実行されます。
処理概要 アプリケーションは、トリガ WSEV_VALUE_CH によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、移動に関するイベント処理を行うことができます。
引数 なし。
復帰値 なし。
注意事項
サンプル

```
void new_class::onValueChange(){
  //スライダー値が変化した場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onValueChange();
}
```

80 WSCvspace

80.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCngbase

80.2 概要

スペースを確保するためのクラスです。実際には何も表示しません。プロパティ WSNextension を、True にすると、配置したフォームを配列フォーム (WSNvertForm/WSChorzForm) のように振舞わせます。そして、バネのように伸縮して、他のインスタンスを移動させたりします。また、プロパティ WSNextension を、False に設定したものは、単独では何もしませんが、WSNextension を、True に設定したものと共に配置すると、伸縮はしませんがスペースを確保するようになります。

80.3 機能

- スペースの確保
- 伸縮機能と、他のインスタンスの再配置機能

80.4 注意事項

80.5 プロパティ一覧

クラス WSCvspace のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値

WSCbool	WSNextension	拡張	0
	不可	False (0)	
	可	True (1)	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

80.6 トリガー一覧

WSCvspace クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

80.7 関数一覧

80.8 関数仕様

81 WSCvtimer

81.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCngbase

81.2 概要

このクラスは指定された時間間隔で、WSEV_ACTIVATE(アクティベート) イベントを発生させます。このクラスは、Non GUI オブジェクトです。プロパティ WSNinterval に時間間隔を指定し、プロパティ WSNcont に継続してイベントを発生させるか否かを指定します。タイマーを動作させるには、プロパティ WSNrunning を True にします。停止させる場合は、WSNrunning を False にします。WSNcont が False の場合は、一度、イベントが発生すると、タイマーは停止して、WSNrunning は False になります。

81.3 機能

- タイマー機能
- タイマーの継続起動設定の機能

81.4 注意事項

81.5 プロパティー覧

クラス WSCvtimer のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned long	WSNinterval	時間間隔	1000
WSCbool	WSNcont	継続	1
	False	False (0)	
	True	True (1)	
WSCbool	WSNrunning	実行中	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

81.6 トリガー覧

WSCvtimer クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

81.7 関数一覧

- void onActivate() :WSCvtimer

81.8 関数仕様

81.8.1 onActivate 関数の説明

書式 void onActivate()

機能 設定された時間が経過した場合に実行されます。

処理概要 アプリケーションは、トリガ (WSEV_ACTIVATE) によるイベントプロシージャを用いる変わりに、この関数をオーバーライドすることでも、タイマーに関するイベント処理を行うことができます。

引数 なし。

復帰値 なし。

注意事項 なし。

サンプル

```
void new_class::onActivate(){
  //時間が経過してタイマーが起動された場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::onActivate();
}
```

82 WSCvtoggle

82.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCnwbase WSCvlabel

82.2 概要

２値の選択を行うオブジェクトです。他のトグルボタンとグルーピングして、いろいろな値の選択形態を実現できます。

82.3 機能

- ２値の選択機能
- グルーピングによる択一選択の機能
- グルーピングによる複数選択の機能

82.4 注意事項

82.5 プロパティー覧

クラス WSCvtoggle のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNselectString	選択時文字列	
WSCbool	WSNindicatorOn	インジケータ使用	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNindicatorType	インジケータ TYPE	0
	IN-OUT	WS_IN_OUT (0)	
	OUT	WS_OUT (1)	
	IN	WS_IN (2)	
	NONE	WS_NONE (3)	
unsigned char	WSNindicatorSize	インジケータ SIZE	20
unsigned char	WSNindicatorShadow	インジケータ影幅	2
short	WSNindicatorColor	インジケータ色	DEF13
short	WSNindicatorPixmap	インジケータ画	
short	WSNselectColor	選択色	DEF7
short	WSNselectPixmap	インジケータ選択画	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNselectReset	選択時選択解除	0
	不可	False (0)	
	可	True (1)	
short	WSNid	ＩＤ	0
WSCbool	WSNenableFocusMove	フォーカス移動	1
	不可	False (0)	
	可	True (1)	
char*	WSNupward	上方向移動先	
char*	WSNdownward	下方向移動先	
char*	WSNleftward	左方向移動先	
char*	WSNrightward	右方向移動先	
unsigned char	WSNshadowThickness	影幅	2
short	WSNbackColor	背景色	DEF13
short	WSNtopShadowColor	上影色	DEF15
short	WSNbottomShadowColor	下影色	DEF16
WSCbool	WSNreverseFlag	反転表示有無	0
	無	False (0)	
	有	True (1)	
short	WSNbackBlinkColor	背景ブリンク表示色	DEF13
unsigned char	WSNblinkType	ブリンクタイプ	0
	文字	WS_FORE (0)	
	背景	WS_BACK (1)	

	文字/背景	WS_BOTH (2)	
WSCbool	WSNemboss	エンボス	0
	不可	False (0)	
	可	True (1)	
WSCbool	WSNusePixmap	ちらつき防止	0
	無	False (0)	
	有	True (1)	
char*	WSNlabelString	表示文字列	
unsigned char	WSNorientation	表示方向	0
	横方向	WS_HORIZONTAL (0)	
	縦方向	WS_VERTICAL (1)	
unsigned char	WSNalignmentV	縦文字配置	2
	上配置	WS_UP (3)	
	中央配置	WS_CENTER (2)	
	下配置	WS_DOWN (4)	
unsigned char	WSNalignmentH	横文字配置	2
	左配置	WS_LEFT (0)	
	中央配置	WS_CENTER (2)	
	右配置	WS_RIGHT (1)	
unsigned char	WSNmarginTop	上マージン	2
unsigned char	WSNmarginBottom	下マージン	2
unsigned char	WSNmarginLeft	左マージン	2
unsigned char	WSNmarginRight	右マージン	2
unsigned short	WSNmaxLength	最大長	64
unsigned char	WSNfont	フォント番号	0
short	WSNlabelPixmap	表示画	
short	WSNblinkPixmap	ブリンク画	
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
char*	WSNname	名称	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
WSCbool	WSNblinkRefreshing	ブリンク再描画	0
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
short	WSNforeColor	表示色	DEF14
WSCbool	WSNtwinBlink	２色ブリンク有無	0
	無	False (0)	
	有	True (1)	
short	WSNblinkColor	ブリンク色	DEF13
unsigned char	WSNblinkRate	ブリンク周期	1
	0.25 秒	WS250MS (1)	
	0.50 秒	WS500MS (2)	
	0.75 秒	WS750MS (3)	
	1.00 秒	WS1000MS (4)	
	2.00 秒	WS2000MS (8)	
	3.00 秒	WS3000MS (12)	
	5.00 秒	WS5000MS (20)	
WSCbool	WSNblinkFlag	ブリンク有無	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNanchorTop	上アンカー	0
unsigned short	WSNanchorBottom	下アンカー	0

unsigned short	WSNanchorLeft	左アンカー	0
unsigned short	WSNanchorRight	右アンカー	0
WSCbool	WSNanchorTopFlag	上アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorBottomFlag	下アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorLeftFlag	左アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNanchorRightFlag	右アンカー使用	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

82.6 トリガー一覧

WSCvtoggle クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
VALUE-CH	オブジェクトの値等が変化した場合	WSEV_VALUE_CH
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
EXPOSE	露出し、描画が必要な場合	WSEV_EXPOSE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
KEY-PRESS	フォーカス獲得状態でキーが押された場合	WSEV_KEY_PRESS
KEY-RELEASE	フォーカス獲得状態でキーが離された場合	WSEV_KEY_RELEASE

82.7 関数一覧

- WSCbool getStatus() :WSCvtoggle
- long setStatus(WSCbool state,WSCbool create_event=True) :WSCvtoggle
- long getGroupValue() :WSCvtoggle
- long setGroupValue(long val) :WSCvtoggle
- long clearGroupValue() :WSCvtoggle

82.8 関数仕様

82.8.1 getStatus 関数の説明

書式 WSCbool getStatus()

機能 選択されているか否かを返します。

処理概要 選択されている場合、True を返します。

引数 なし。

復帰値 トグルの選択状態

注意事項 なし。

サンプル

```
WSCbool status = newvtog_000->getStatus();
if (status != False){
  //選択状態
}else{
  //非選択状態
}
```

82.8.2 setStatus 関数の説明

書式 long setStatus(WSCbool state,WSCbool create_event=True)

機能 選択状態を設定します。

処理概要 state = True を指定した場合、選択された状態になります。create_event = True の場合、WSEV_VALUE_CH イベントが発生します。

引数

(in)WSCbool state	設定したい選択状態
(in)WSCbool create_event	イベントの発生の有無

復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。

注意事項 なし。

サンプル

```
//選択状態の設定を行います。
newvtog_000->setStatus(True); //選択状態に設定
newvtog_000->setStatus(False); //非選択状態に設定
```

82.8.3 getGroupValue 関数の説明

書式 long getGroupValue()

機能 グループ化されているトグルの選択されているもののプロパティ WSNid の値を返します。択一属性の場合は、WSNid そのもの、複数選択属性の場合は、選択されているものの WSNid 値がビットＯＲされたものが返されます。

処理概要

引数 なし。

復帰値 トグルグループの選択値が返されます。

注意事項 なし。

サンプル

```
//トグルグループ選択状態を取得します。
long value = newvtog_000->getGroupValue();
```

82.8.4 setGroupValue 関数の説明

書式 long setGroupValue(long value)

機能 グループ選択状態を設定します。

処理概要 択一選択属性になっている場合、プロパティ WSNid がヒットするものが選択状態になります。また、複数選択属性になっている場合は、指定された値の立っているビット番号と WSNid と一致するものが選択状態となります。

引数

(in)long value	設定したい選択状態

復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。

注意事項 なし。

サンプル

```
//グループの選択状態の設定を行います。
newvtog_000->setGroupValue(value);
```

82.8.5 clearGroupValue 関数の説明

書式 long clearGroupValue()

機能 グループ化されているトグルの選択状態を解除します。

処理概要

引数 なし。

復帰値 WS_NO_ERR = 正常、WS_ERR = 異常。

注意事項 なし。

サンプル

```
//グループの選択状態を解除します。
newvtog_000->clear();
```

83 WSCwindow

83.1 継承元

次のオブジェクトを継承しています。
WSCbase

83.2 概要

一つの独立したウィンドウを提供します。通常、アプリケーションウィンドウのベースとして使われます。

83.3 機能

- 独立したウィンドウの提供
- ウィンドウマネージャの装飾
- 他のオブジェクトの配置と管理機能
- 背景ピクスマップの表示機能

83.4 注意事項

83.5 プロパティ一覧

クラス WSCwindow のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	30
unsigned char	WSNshadowThickness	影幅	0
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	

WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNindicatorOn	インジケータ使用	1
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNmargin	マージン	2
unsigned short	WSNmarginLeft	左マージン	2
WSCbool	WSNemboss	エンボス	0
	オフ	False (0)	
	オン	True (1)	
unsigned char	WSNbarThickness	バー幅	16
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

83.6 トリガー一覧

WSCwindow クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE

83.7 関数一覧

- long setMapStatus(long) :WSCwindow
- void adjustForm() :WSCwindow
- long setWindowStatus(long) :WSCwindow
- long getWindowStatus() :WSCwindow

83.8 関数仕様

83.8.1 setMapStatus 関数の説明

書式 long setMapStatus(long)

機能 他のウィンドウとの重なり状態を変えます。

処理概要 WS_TOP を指定した場合、一番上位に表示します。また、WS_BOTTOM を指定した場合、一番下位に表示します。

引数

(in)long status	位置指定

復帰値 WS_NO_ERR = 正常、それ以外はエラー。
注意事項 なし。
サンプル
```
//重なり状態を設定します。
newwin000->setMapStatus(WS_TOP); //ウィンドウをトップに持って来ます。
```

83.8.2 adjustForm 関数の説明

書式 virtual void adjustForm()
機能 サイズが変更された時に呼び出されます。
処理概要 この関数をオーバーライドすることによって、サイズが変更されたときに、処理すべきことを行います。例えば、ウィンドウに配置されているオブジェクトのサイズを追従させて変化させる場合は、この adjustForm 関数をオーバーライドして、その中で行います。
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル
```
void new_class::adjustForm(){
  //フォームがリサイズされた場合に行う処理を記述します。

  //処理を派生元クラスに引き継ぎます。
  old_class::adjustForm();
}
```

83.8.3 getWindowStatus 関数の説明

書式 long getWindowStatus()
機能 ウィンドウのアイコン化、最大化の状態を取得します。
処理概要 ウィンドウの表示状態を取得します。通常の表示状態の場合、WS_WINDOW_NOMAL、アイコン化の表示状態の場合、WS_WINDOW_MINIMIZE、最大化の表示状態の場合、WS_WINDOW_MAXIMIZE が返されます。
引数 なし。
復帰値 WS_WINDOW_NORMAL/WS_WINDOW_MINIMIZE/WS_WINDOW_MAXIMIZE
注意事項 なし。
サンプル
```
//表示状態を指定します。
newwin000->setWindowStatus(WS_WINDOW_MINIMIZE);
```

83.8.4 setWindowStatus 関数の説明

書式 long setWindowStatus(long status)
機能 ウィンドウのアイコン化、最大化の状態を指定します。
処理概要 ウィンドウの表示状態を指定します。通常の表示状態の場合、WS_WINDOW_NOMAL、アイコン化の表示状態の場合、WS_WINDOW_MINIMIZE、最大化の表示状態の場合、WS_WINDOW_MAXIMIZE を指定します。
引数

(in)long status	表示状態指定

復帰値 WS_NO_ERR = 正常、それ以外はエラー。
注意事項 なし。
サンプル
```
//表示状態を指定します。
long status  = newwin000->getWindowStatus();
if (status == WS_WINDOW_NORMAL){
  //通常状態
}else
if (status == WS_WINDOW_MINIMIZE){
  //アイコン化状態
}else
if (status == WS_WINDOW_MAXIMIZE){
```

```
    //最大化状態
}
```

84 WSCwizardDialog

84.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow WSCbaseDialog

84.2 概要

ウィザードダイアログを提供します。戻る、進むボタンで、場面を切替え、完了ボタンで、ダイアログへの入力が完了します。
場面の切替えは、内部のインデックスフォームクラスが行います。プロパティの WSNmenuItems が場面数を指定します。

84.3 機能

- 戻るボタン、進むボタンでの場面の切替え
- ひな型ウィザードダイアログの機能の提供

84.4 注意事項

84.5 プロパティ一覧

クラス WSCwizardDialog のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
char*	WSNlabelString	表示文字列	¡ 戻る, 次へ ¿, 完了
unsigned char	WSNmenuItems	メニュー項目	2
unsigned char	WSNvalue	値	0
short	WSNworkBackColor	作業領域背景色	DEF10
WSCbool	WSNok	OK ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNno	NO ボタン	1
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	
char*	WSNokString	OK ボタン表示文字列	¡ 戻る
char*	WSNnoString	NO ボタン表示文字列	次へ ¿
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	100
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3

short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	
	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

84.6 トリガー一覧

WSCwizardDialog クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

84.7 関数一覧

- WSCbase* getRightBtn() :WSCwizardDialog
- WSCbase* getLeftBtn() :WSCwizardDialog

84.8 関数仕様

84.8.1 getRightBtn 関数の説明

書式 WSCbase* getRightBtn()
機能 右ボタンを取得します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
//右ボタンを取得します。
WSCbase* rbtn = newwiza_000->getRightBtn();
```

84.8.2 getLeftBtn 関数の説明

書式 WSCbase* getLeftBtn()
機能 左ボタンを取得します。
処理概要
引数 なし。
復帰値 なし。
注意事項
サンプル

```
//左ボタンを取得します。
WSCbase* rbtn = newwiza_000->getRightBtn();
```

85 WSCworkingDialog

85.1 継承元

次のオブジェクトを継承しています。
WSCbase WSCwindow WSCbaseDialog

85.2 概要

処理の進み具合を表示するための、プログレッシングインジケータバーをもったダイアログです。プロパティ WSNvalue に 0 〜100 の範囲で値を設定します。

85.3 機能

・プログレッシングインジケータバーの表示

85.4 注意事項

85.5 プロパティ一覧

クラス WSCworkingDialog のプロパティ一覧は次の通りです。

型	プロパティ名称	補足	デフォルト値
unsigned short	WSNvalue	値	0
char*	WSNlabelString	表示文字列	Working...

short	WSNworkBackColor	作業領域背景色	DEF10
unsigned short	WSNlineWidth	線幅	2
unsigned char	WSNlineType	線種	0
	実線	(0)	
	点線	(1)	
	鎖線	(2)	
	長鎖線	(3)	
	一点鎖線	(4)	
	二点鎖線	(5)	
	長一点鎖線	(6)	
	長二点鎖線	(7)	
	細点線	(8)	
WSCbool	WSNstippled	ハッチパターン有無	1
	無	False (0)	
	有	True (1)	
unsigned char	WSNhatchPattern	ハッチパターン	0
	ベタ塗り	(0)	
	左下斜線	(1)	
	右下斜線	(2)	
	縦線	(3)	
	横線	(4)	
	斜め交差	(5)	
	縦横交差	(6)	
	ドット	(7)	
short	WSNhatchColor	ハッチパターン表示色	DEF14
WSCbool	WSNok	OK ボタン	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNno	NO ボタン	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNcancel	CANCEL ボタン	1
	不可視	False (0)	
	可視	True (1)	
short	WSNlabelPixmap	表示画	
char*	WSNokString	OK ボタン表示文字列	OK
char*	WSNnoString	NO ボタン表示文字列	NO
char*	WSNcancelString	CANCEL ボタン表示文字列	Cancel
WSCbool	WSNmodal	モーダル	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNdefaultPosition	中央表示	0
	オフ	False (0)	
	オン	True (1)	
char*	WSNname	名称	
char*	WSNtitleString	タイトル文字列	
char*	WSNuserString	ユーザ設定文字列	
long	WSNuserValue	ユーザ設定値	0
short	WSNx	X座標	0
short	WSNy	Y座標	0
unsigned short	WSNwidth	横幅	100
unsigned short	WSNheight	縦幅	100
short	WSNforeColor	表示色	DEF2
short	WSNbackColor	背景色	DEF1
short	WSNtopShadowColor	上影色	DEF3
short	WSNbottomShadowColor	下影色	DEF4
short	WSNbackPixmap	背景画	
char	WSNtitleBar	タイトル属性	1
	タイトルなし	WS_NO_TITLE (0)	
	全装飾	WS_FULL_TITLE (1)	
	タイトル	WS_ONLY_TITLE (2)	
	最小化ボタン	WS_MINI_BTN (3)	
	最大化ボタン	WS_MAX_BTN (4)	
	枠	WS_FRAME (5)	
	WM管理外	WS_NO_MANAGE (6)	
char	WSNshadowType	影タイプ	0
	陥没	WS_SHADOW_IN (1)	
	突出	WS_SHADOW_OUT (0)	
	陥没枠	WS_SHADOW_EIN (3)	
	突出枠	WS_SHADOW_EOUT (2)	

	ボーダ枠	WS_SHADOW_BORDER (4)	
	なし	WS_SHADOW_TRANS (-1)	
WSCbool	WSNvis	表示属性	0
	不可視	False (0)	
	可視	True (1)	
WSCbool	WSNdet	操作属性	1
	不可	False (0)	
	可	True (1)	
unsigned char	WSNpixmapStyle	描画方法	0
	WINDOW	WS_DIRECT_WINDOW (0)	
	動的 PIXMAP	WS_DYNAMIC_PIXMAP (1)	
char*	WSNgroup	グループ指定	
WSCbool	WSNunique	択一選択属性	0
	オフ	False (0)	
	オン	True (1)	
unsigned short	WSNmouse	マウス番号	0
unsigned char	WSNgradation	グラデーションタイプ	0
	無し	WS_GR_NONE (0)	
	左上→右下	WS_GR_LT_RB (1)	
	右上→左下	WS_GR_RT_LB (2)	
	左下→右上	WS_GR_LB_RT (3)	
	右下→左上	WS_GR_RB_LT (4)	
	上→下	WS_GR_T_B (5)	
	下→上	WS_GR_B_T (6)	
	左→右	WS_GR_L_R (7)	
	右→左	WS_GR_R_L (8)	
unsigned char	WSNgradationMargin	グラデーションマージン	0
WSCbool	WSNexit	終了	0
	オフ	False (0)	
	オン	True (1)	
WSCbool	WSNexport	エクスポート	0
	不可	False (0)	
	可	True (1)	

85.6 トリガー一覧

WSCworkingDialog クラスでは、次のトリガが使用可能です。

トリガ名	トリガ種別	プログラム中での定義値
トリガなし	起動トリガを指定しない	WSEV_NONE
INITIALIZE	初期化された場合	WSEV_INITIALIZE
DELETE	オブジェクトが開放された場合	WSEV_DELETE
ACTIVATE	オブジェクトが活性状態になった場合	WSEV_ACTIVATE
FOCUS-CH	キーボードフォーカスを獲得／喪失し、状態が変化した場合	WSEV_FOCUS_CH
VISIBLE-CH	表示属性が変化した場合	WSEV_VISIBLE_CH
PARENT-VISIBLE-CH	親オブジェクトの表示属性が変化した場合	WSEV_PARENT_VISIBLE_CH
RESIZE	オブジェクトのサイズが変更された場合	WSEV_RESIZE
MOUSE-IN	マウスポインタが領域内に入った場合	WSEV_MOUSE_IN
MOUSE-OUT	マウスポインタが領域外に出た場合	WSEV_MOUSE_OUT
MOUSE-PRESS	マウスのボタンが押された場合	WSEV_MOUSE_PRESS
MOUSE-RELEASE	マウスのボタンがはなされた場合	WSEV_MOUSE_RELEASE
MOUSE-MOVE	マウスポインタが動いた場合	WSEV_MOUSE_MOVE
GUI-POLICY-CH	GUI ポリシーが変更された場合	WSEV_GUI_POLICY_CH

85.7 関数一覧

- virtual void setValue(WSCushort value); :WSCworkingDialog

85.8 関数仕様

85.8.1 setValue 関数の説明

書式 void setValue(WSCushort value)

機能 0〜100 の値を設定します。

処理概要

引数

(in)WSCushort value	インジケータバーの長さ (0〜100)

復帰値 なし。
注意事項
サンプル
```
//処理の進み具合をバーに設定します。
newwork_000->setValue(88);
```

86 WSDappDev

86.1 継承元

次のオブジェクトを継承しています。

86.2 概要

アプリケーションインスタンスです。一つのアプリケーションに一つの必ず WSDappDev インスタンスが存在します。グローバルインスタンス関数 WSGIappDev() でアクセスします。

86.3 機能

- main(int argc,char** argv) の argc,argv の保持
- アプリケーション名称 (プロジェクト名称) の保持
- ディスプレイの縦横の幅の取得

86.4 注意事項

86.5 関数一覧

- WSDappDev* WSGIappDev(); :WSDappDev
- int getArgc() :WSDappDev
- char* getArgv() :WSDappDev
- char* getInstanceName() :WSDappDev
- WSCushort getWidth() :WSDappDev
- WSCushort getHeight() :WSDappDev

86.6 関数仕様

86.6.1 WSGIappDev 関数の説明

書式 WSDappDev* WSGIappDev()
機能 アプリケーションにひとつ存在するアプリケーションクラスのグローバルインスタンスを取得します。
処理概要
引数 なし。
復帰値 アプリケーションインスタンスへのポインタ
注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。
サンプル
```
//main() に引き渡される、argc、argv の取得の例
int argc = WSGIappDev()->getArgc();
char** argv = WSGIappDev()->getArgv();
```

86.6.2 getArgc 関数の説明

書式 int getArgc()
機能 main(int argc,char** argv) で渡されて来る argc を返します。
処理概要
引数 なし。

復帰値 argc
注意事項 なし。
サンプル WSGIappDev() を参照してください。

86.6.3 getArgv 関数の説明

書式 char** getArgv();
機能 main(int argc,char** argv) で渡されて来る argv を返します。
処理概要
引数 なし。
復帰値 argv
注意事項 返される値を開放してはいけません。
サンプル WSGIappDev() を参照してください。

86.6.4 getInstanceName 関数の説明

書式 char* getInstanceName()
機能 プロジェクト名称を返します。
処理概要
引数 なし。
復帰値 プロジェクト名称
注意事項 返される値を開放してはいけません。なお、コマンド名称を取得したい場合は、getArgv() で取得されるものの配列要素 0 番を御参照ください。
サンプル

```
//プロジェクト名称を取得します。
char* iname = WSGIappDev()->getInstanceName();
```

86.6.5 getWidth() 関数の説明

書式 unsigned short getWidth()
機能 ディスプレイの横幅をドット単位で返します。
処理概要
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル

```
//ディスプレイの横幅を取得します。
WSCushort dw = WSGIappDev()->getWidth();
```

86.6.6 getHeight() 関数の説明

書式 unsigned short getHeight()
機能 ディスプレイの縦幅をドット単位で返します。
処理概要
引数 なし。
復帰値 なし。
注意事項 なし。
サンプル

```
//ディスプレイの縦幅を取得します。
WSCushort dh = WSGIappDev()->getHeight();
```

87 WSDcolor

87.1 継承元

次のオブジェクトを継承しています。

87.2 概要

色クラスです。色名称と３原色とピクセル値を保持します。

87.3 機能

- 色名称の保持
- ３原色の保持
- ピクセル値の保持

87.4 注意事項

87.5 関数一覧

- char* getColorName() :WSDcolor
- long getRGB(long*,long*,long*) :WSDcolor
- long setColorName(char* name) :WSDcolor

87.6 関数仕様

87.6.1 getColorName 関数の説明

書式 char* getColorName()

機能 色名称を取得します。

処理概要

引数 なし。

復帰値 色名称

注意事項

サンプル

```
WSDcolor* color = WSGIappColorSet()->getColor("#ffffff");
char* cname = color->getColorName();// #ffffff が返されます。
```

87.6.2 getRGB 関数の説明

書式 long getRGB(long* r,long* g,long* b)

機能 色の RGB 値を取得します。

処理概要

引数

(out)long* r	赤 (0 〜 0xff)
(out)long* g	緑 (0 〜 0xff)
(out)long* b	青 (0 〜 0xff)

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
WSDcolor* color = WSGIappColorSet()->getColor("#ffffff");
long r,g,b;
color->getRGB(\&r,\&g,\&b); // r=0xff、g=0xff、b=0xff が返されます。
```

87.6.3 setColorName 関数の説明

書式 long setColorName(char* cname)

機能 色名称を設定します。

処理概要

引数

(out)char* cname	色名称

色名称は、WSCcolorSet クラスの getColor() メソッドの説明を御覧ください。

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

```
color->setColorName("#ff0000"); //保持している色情報が指定したものに変ります。
```

88 WSDdev

88.1 継承元

次のオブジェクトを継承しています。

88.2 概要

ウィンドウシステム依存の資源を管理するクラスです。

88.3 機能

- ウィンドウ資源を保持します。
- ウィンドウへの描画を行います。
- ウィンドウの操作を行います。
- ウィンドウシステムからイベントの発生を受け付けます。

88.4 注意事項

88.5 関数一覧

- void setScalePtr(double*) :WSDdev
- void setXOffsetPtr(short*) :WSDdev
- void setYOffsetPtr(short*) :WSDdev
- double getScale() :WSDdev
- short getXOffset() :WSDdev
- short getYOffset() :WSDdev
- long setForeColor(short) :WSDdev
- long setBackColor(short) :WSDdev
- long setLineWidth(short) :WSDdev
- long setLineDashType(char) :WSDdev
- long setHatchPattern(char) :WSDdev
- long setRegion(short x,short y,WSCushort w,WSCushort h) :WSDdev
- long beginDraw(short x,short y,WSCushort w,WSCushort h,WSCbool absolute = False,WSCbool scaling = True) :WSDdev
- long endDraw() :WSDdev
- long drawArc(short x,short y,unsinged short w,unsinged short h,short a1,short a2) :WSDdev
- long drawFillArc(short x,short y,unsinged short w,unsinged short h,short a1,short a2,char kind) :WSDdev
- long drawLine(short x,short y,short x2,short y2) :WSDdev
- long drawLines(WSCpoint*,long num) :WSDdev
- long drawRect(short x,short y,unsinged short w,unsiged short h) :WSDdev
- long drawFillRect(short x,short y,unsinged short w,unsiged short h) :WSDdev
- long drawRects(WSCrect*,long num) :WSDdev
- long drawFillRects(WSCrect*,long num) :WSDdev
- long drawPoly(WSCpoint*,long num) :WSDdev
- long drawFillPoly(WSCpoint*,long num) :WSDdev
- long drawImage(short,short,WSCushort,WSCushort,WSDimage*,char) :WSDdev
- long drawStretchedImage(short,short,WSCushort,WSCushort,WSDimage*) :WSDdev
- long drawString(long,long,WSCulong,WSCulong,char,char,char*,long) :WSDdev
- long drawFillString(long,long,WSCulong,WSCulong,char,char,char*,long) :WSDdev
- long getDeviceResource() :WSDdev
- long getWindowResource() :WSDdev
- long getContextResource() :WSDdev
- long getSpecialResource() :WSDdev
- WSCbool getReady() :WSDdev

88.6 関数仕様

88.6.1 setScalePtr 関数の説明

書式 void setScalePtr(double* ptr)
機能 表示倍率を格納した変数へのポインタを指定します。
処理概要 表示倍率を格納した変数へのポインタが設定されると、その倍率で描画します。
引数

(in)double* ptr	倍率を格納した変数へのポインタ

復帰値 なし。
注意事項 ポインタの設定を解除したい場合は、NULL を指定します。
サンプル

```
dev->setScalePtr(\&scale);//double 型である scale のポインタを渡します。
```

88.6.2 setXOffsetPtr 関数の説明

書式 void setXOffset(short* ptr)
機能 表示位置Xのオフセットを格納した変数へのポインタを指定します。
処理概要 表示位置Xのオフセットを格納した変数へのポインタが設定されると、オフセット値を加算した座標で描画します。
引数

(in)short* ptr	オフセット値を格納した変数へのポインタ

復帰値 なし。
注意事項 ポインタの設定を解除したい場合は、NULL を指定します。
サンプル

```
dev->setXOffsetPtr(\&xoffset);//short 型である xoffset のポインタを渡します。
```

88.6.3 setYOffsetPtr 関数の説明

書式 void setYOffset(short* ptr)
機能 表示位置Yのオフセットを格納した変数へのポインタを指定します。
処理概要 表示位置Yのオフセットを格納した変数へのポインタが設定されると、オフセット値を加算した座標で描画します。
引数

(in)short* ptr	オフセット値を格納した変数へのポインタ

復帰値 なし。
注意事項 ポインタの設定を解除したい場合は、NULL を指定します。
サンプル

```
dev->setYOffsetPtr(\&yoffset);//short 型である yoffset のポインタを渡します。
```

88.6.4 getScale 関数の説明

書式 double getScale()
機能 表示倍率を取得します。
処理概要 表示倍率を格納した変数へのポインタが設定されている場合、その値を返します。設定されていない場合は、1 が返されます。
引数 なし。
復帰値 表示倍率
注意事項 なし。
サンプル

```
//WSDdev インスタンスに設定されているスケールオフセットを取得します。
double scale = dev->getScale();
```

88.6.5 getXOffset 関数の説明

書式 short getXOffset()
機能 表示位置Xのオフセット
処理概要 表示位置Xのオフセットを格納したポインタが設定されている場合は、その値が返されます。設定されていない場合、0 が返されます。
引数 なし。

復帰値 表示位置Xのオフセット
注意事項 なし。
サンプル
```
//WSDdev インスタンスに設定されている X オフセットを取得します。
short xoffset = dev->getXOffset();
```

88.6.6 getYOffset 関数の説明

書式 short getYOffset()
機能 表示位置Yのオフセット
処理概要 表示位置Yのオフセットを格納したポインタが設定されている場合は、その値が返されます。設定されていない場合、0 が返されます。
引数 なし。
復帰値 表示位置Yのオフセット
注意事項 なし。
サンプル
```
//WSDdev インスタンスに設定されている Y オフセットを取得します。
short yoffset = dev->getYOffset();
```

88.6.7 setForeColor 関数の説明

書式 long setForeColor(short cno)
機能 描画色を色番号で指定します。
処理概要 drawLine 等のメソッドで用いる描画色を指定します。
引数

(in)short cno	色番号

復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項
サンプル
```
//実際の WSCvline::draw() の例です。
long WSCvline::draw(){
  if (getVisible() == False){
    return WS_NO_ERR;
  }
  WSDdev* dev = getowndev();
  if (dev == NULL){
    return WS_ERR;
  }
  short x = _x;
  short y = _y;
  WSCushort w = _w;
  WSCushort h = _h;

  WSCbool absolute = getAbsoluteDraw();
  if (absolute == True){
  }else
  if (dev->isExposed(x, y, w, h) == False){
    return WS_NO_ERR;
  }
  WSCbool blinkfore = WSGIappBlink()->getBlinkFore(_blink_rate);
  if (_bl_fl == True \&\&
      blinkfore == True \&\&
      _tw_fl == False){
    return WS_NO_ERR;
  }

  long err = dev->beginDraw(x, y, w, h, absolute);
  if (err != WS_NO_ERR){
    return WS_NO_ERR;
  }
```

```
  WSCbase::update();
  setAbsoluteDraw(False);
  WSCushort line_w = _lw;
  if (line_w != 0) {
    if (_bl_fl == True \&\&
        blinkfore == True \&\&
        _tw_fl == True){
      dev->setForeColor(_blink_color);
    } else {
      dev->setForeColor(_fore_color);
    }

    if (line_w == 1) {
      dev->setLineWidth((WSCushort)0);
    } else {
      dev->setLineWidth((WSCushort)line_w);
    }
    dev->setLineDashType(_lt);
    if (_line_num != 0) {
      WSCpoint* pt = new WSCpoint [_line_num] ;
      short     i;
      for (i=0; i<_line_num; i++) {
        pt [i] .x = (short)(_line_x [i] );
        pt [i] .y = (short)(_line_y [i] );
      }
      dev->drawLines(pt, _line_num);
      delete pt;
    }
  }
  dev->endDraw();
  return WS_NO_ERR;
}
```

88.6.8 setBackColor 関数の説明

書式 long setBacKColor(short cno)

機能 背景色を指定を色番号で指定します。

処理概要 drawFillRect 等のメソッドで用いる背景色を指定します。

引数

(in)short cno	色番号

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項

サンプル

```
//背景色を設定します。
short cno = WSGIappColorSet()->getColorNo("#000000");
dev->setBackColor(cno);
```

88.6.9 setLineWidth 関数の説明

書式 long setLineWidth(short linewidth)

機能 描画する線の太さを指定します。

処理概要 drawLine 等のメソッドで用いる線幅を指定します。

引数

(in)short linewidth	線幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

88.6.10 setLineDashType 関数の説明

書式 long setLineDashType(char no)

機能 描画する線の点線属性を指定します。

処理概要 drawLine 等のメソッドで用いる線の点線属性を指定します。

引数

(in)char no	点線属性

点線属性に指定できる値には、次のようなものがあります。

属性値	意味
0	実線
1	鎖線
2	長鎖線
3	一点鎖線
4	二点鎖線
5	長一点鎖線
6	長二点鎖線
7	細点線

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項

サンプル setForeColor() を参照してください。

88.6.11 setHatchPattern 関数の説明

書式 long setHatchPattern(char no)

機能 塗りつぶし属性を指定します。

処理概要 drawRect 等のメソッドで使用される塗りつぶし属性を指定します。

引数

(in)char no	塗りつぶし属性

塗りつぶし属性に指定できる値には、次のようなものがあります。

属性値	意味
0	ベタ塗
1	左下斜線
2	右下斜線
3	縦線
4	横線
5	斜め交差
6	縦横交差
7	ドット

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//ハッチパターンを設定します。
dev->setHatchPattern(7);
```

88.6.12 setRegion 関数の説明

書式 long setRegion(short x,short y,unsigned short w,unsigned short h);

機能 描画する領域を指定します。

処理概要 指定した領域以外は描画の対象から外します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//描画する領域を設定します。
dev->setRegion(10,10,100,100);
```

88.6.13 beginDraw 関数の説明

書式 long beginDraw(short x,short y,WSCushort w,WSCushort h, WSCbool absolute = False,WSCbool scaling = True)

機能 指定した領域の描画を開始します。

処理概要 座標 x,y、横幅 w、縦幅 h を矩形領域の描画を開始します。absolute を True とすると露出イベントが発生していなくても描画します。また、scaling が True の場合、スケールオフセットを有効にして描画します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅
(in)WSCbool absolute	強制描画フラグ
(in)WSCbool scaling	スケールフラグ

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

88.6.14 drawArc 関数の説明

書式 long drawArc(short x,short y,unsigned short w, unsigned short h,short a1,short a2);

機能 円弧、楕円弧を描画します。

処理概要 横直径 w、縦直径 h を持つ円または楕円の、X 軸右水平を 0 度とした反時計回りで、円弧開始角度、終了角度を指定します。このとき、64 倍した値を指定するのに注意してください。これは、1/64 度の精度確保するためです。完全な円、楕円、または円弧を描く場合は、それぞれ 0 * 64、360 * 64 を指定します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅
(in)short a1	円弧の開始角度
(in)short a2	円弧の終了角度

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//円弧の描画を行います。
dev->drawArc(50,200,50,50,60*64,270*64);
```

88.6.15 drawFillArc 関数の説明

書式 long drawFillArc(short x,short y,unsigned short w, unsigned short h,short a1,short a2,char kind);

機能 塗りつぶし円弧、楕円弧を描画します。

処理概要 横直径 w、縦直径 h を持つ円または楕円の、X 軸右水平を 0 度とした反時計回りで、円弧開始角度、終了角度を指定します。このとき、64 倍した値を指定するのに注意してください。これは、1/64 度の精度確保するためです。完全な円、または円弧を描く場合は、それぞれ 0 * 64、360 * 64 を指定します。円弧の種別を指定することによって、扇型の円弧、弓型の円弧を描画することができます。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	領域の横幅
(in)unsigned short h	領域の縦幅
(in)short a1	円弧の開始角度
(in)short a2	円弧の終了角度
(in)char kind	塗りつぶし円弧の種別

塗りつぶし種別に指定できる値には、次のようなものがあります。

属性値	意味
0	扇型
1	弓型

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//パイ型円弧の描画を行います。
dev->drawFillArc(150,200,100,50,60*64,270*64, 0 /*PIE*/ );
//弓型円弧の描画を行います。
dev->drawFillArc(250,200,100,50,60*64,270*64, 1 /*CHORD*/ );
```

88.6.16 drawLine 関数の説明

書式 long drawLine(short x1,short y1,short x2,short y2);

機能 線を描画します。

処理概要 指定した座標 x1,y1 から x2,y2 まで線を描画します。

引数

(in)short x1	線の開始座標X
(in)short y1	線の開始座標Y
(in)short x2	線の終了座標X
(in)short y2	線の終了座標Y

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//線を描画します。
dev->drawLine(50,50,100,100);
```

88.6.17 drawLines 関数の説明

書式 long drawLines(WSCpoint* pt,long num);

機能 折れ線を描画します。

処理概要 指定した座標 X1,Y1 から Xn,Yn まで折れ線を描画します。

引数

(in)WSCpoint* pt	端点座標の配列
(in)long num	端点の数

WSCpoint は、メンバ x,y を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル setForeColor() を参照してください。

88.6.18 drawRect 関数の説明

書式 long drawRect(short x,short y,unsigned short w,unsigned short h);

機能 矩形を描画します。

処理概要 左上角を x,y とした横幅 w、縦幅 h の矩形を描画します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	矩形の横幅
(in)unsigned short h	矩形の縦幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//矩形を描画します。
dev->drawRect(150,50,100,100);
```

88.6.19 drawFillRect 関数の説明

書式 long drawFillRect(short x,short y,unsigned short w,unsigned short h);

機能 塗りつぶし矩形を描画します。

処理概要 左上角を x,y とした横幅 w、縦幅 h の塗りつぶし矩形を描画します。

引数

(in)short x	X座標
(in)short y	Y座標
(in)unsigned short w	矩形の横幅
(in)unsigned short h	矩形の縦幅

復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル
```
//塗りつぶし矩形を描画します。
dev->drawFillRect(150,50,100,100);
```

88.6.20 drawRects 関数の説明

書式 long drawRects(WSCrect* pt,long num);
機能 矩形を複数個描画します。
処理概要 WSCrect 配列で指定された複数の矩形を描画します。
引数

(in)WSCrect* rect	矩形の配列
(in)long num	矩形の数

WSCrect は、メンバ x,y,width,height を持ちます。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル
```
//矩形を描画します。
WSCrect rect[3];
rect[0].setRect(300,300,10,10);
rect[1].setRect(320,300,10,10);
rect[2].setRect(340,300,10,10);
dev->drawRects(rect,3);
```

88.6.21 drawFillRects 関数の説明

書式 long drawFillRects(WSCrect* pt,long num);
機能 塗りつぶし矩形を複数個描画します。
処理概要 WSCrect 配列で指定された複数の塗りつぶし矩形を描画します。
引数

(in)WSCrect* rect	矩形の配列
(in)long num	端点の数

WSCrect は、メンバ x,y,width,height を持ちます。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
```
//矩形を描画します。
WSCrect rect[3];
rect[0].setRect(300,300,10,10);
rect[1].setRect(320,300,10,10);
rect[2].setRect(340,300,10,10);
dev->drawFillRects(rect,3);
```

88.6.22 drawPoly 関数の説明

書式 long drawPoly(WSCpoint* pt,long num);
機能 多角形を描画します。
処理概要 指定した座標 X1,Y1 から Xn,Yn までを結ぶ多角形を描画します。
引数

(in)WSCpoint* pt	端点座標の配列
(in)long num	端点の数

WSCpoint は、メンバ x,y を持ちます。
復帰値 WS_NO_ERR:成功、WS_ERR:失敗
注意事項 なし。
サンプル
```
WSCpoint pt[3];
pt[0].x = 300;
pt[0].y = 300;
pt[1].x = 330;
pt[1].y = 330;
```

```
pt［2］.x = 300;
pt［2］.y = 330;
dev->setForeColor("#00ffff");
dev->drawPoly(pt,3);
```

88.6.23 drawFillPoly 関数の説明

書式 long drawFillPoly(WSCpoint* pt,long num);

機能 塗りつぶし多角形を描画します。

処理概要 指定した座標 X1,Y1 から Xn,Yn までを結ぶ塗りつぶし多角形を描画します。

引数

(in)WSCpoint* pt	端点座標の配列
(in)long num	端点の数

WSCpoint は、メンバ x,y を持ちます。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCpoint pt［3］;
pt［0］.x = 300;
pt［0］.y = 300;
pt［1］.x = 330;
pt［1］.y = 330;
pt［2］.x = 300;
pt［2］.y = 330;
dev->setForeColor("#00ffff");
dev->drawFillPoly(pt,3);
```

88.6.24 drawGradation 関数の説明

書式 long drawGradation(long type,short col1,short col2, short col3,short x,short y,WSCushort w,WSCushort h, WSCuchar grad_margin);

機能 グラデーションされた矩形を描画します。

処理概要 指定された３色を使用して、色１～色２～色３へとグラデーションされた矩形を描画します。

引数

(in)long type	グラデーションの種別
(in)short col1	色１
(in)short col2	色２
(in)short col3	色３
(in)short x	座標X
(in)short y	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)WSCuchar grad_margin	色２の幅

グラデーションされた矩形領域を指定し、グラデーションのマージンには、色２の幅をドット指定します。グラデーション種別に指定できる値には、次のようなものがあります。

グラデーション種別	意味
WS_GR_LT_RB	左上→右下
WS_GR_RT_LB	右上→左下
WS_GR_LB_RT	左下→右上
WS_GR_RB_LT	右下→左上
WS_GR_T_B	上→下
WS_GR_B_T	下→上
WS_GR_L_R	左→右
WS_GR_R_L	右→左

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
short cno1 = WSGIappColorSet()->getColorNo("#888888");
short cno2 = WSGIappColorSet()->getColorNo("#aaaaaa");
short cno3 = WSGIappColorSet()->getColorNo("#000000");
dev->drawGradation(WS_GR_T_B,cno1,cno2,cno3,10,10,100,50,10);
```

88.6.25 drawImage 関数の説明

書式 long drawImage(short x,short y,WSCushort w,WSCushort h, WSDimage* img,char align);

機能 画像を表示します。

処理概要 指定された画像構造体を指定矩形内に描画します。

引数

(in)short x	座標X
(in)short y	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)WSDimage* img	表示させたい画像
(in)char align	アラインメント

アラインメントに指定できる値には、次のようなものがあります。

属性値	意味
WS_LEFT	左寄
WS_RIGHT	右寄
WS_CENTER	中央
WS_TOP	上寄
WS_BOTTOM	下寄
WS_LEFT_TOP	左上寄
WS_LEFT_BOTTOM	左下寄
WS_RIGHT_BOTTOM	右下寄
WS_RIGHT_TOP	右上寄

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCimage* image = WSGIappImageSet()->getImage("001.jpg");
dev->drawImage(10,10,100,100,image,WS_CENTER);
```

88.6.26 drawStretchedImage 関数の説明

書式 long drawStretchedImage(short x,short y, WSCushort w,WSCushort h, WSDimage* img);

機能 画像を伸縮して表示します。

処理概要 指定された画像を、指定された矩形に伸縮して表示します。

引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)WSDimage* img	表示させたい画像

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
WSCimage* image = WSGIappImageSet()->getImage("001.jpg");
dev->drawStretchedImage(10,10,100,100,image);
```

88.6.27 drawString 関数の説明

書式 long drawString(short x,short y,WSCushort w,WSCushort h, char font_no,char align,char* string, long encoding = WS_EN_DEFAULT);

機能 指定された文字列を指定された矩形内に表示します。

処理概要 指定された文字列をエンコーディングに従がって指定された矩形内に表示します。

引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)char font_no	フォント番号
(in)char align	アラインメント
(in)char* string	表示させたい文字列
(in)long encoding	文字列のエンコーディング

フォント番号には、0番からのフォント番号を指定します。アラインメントに指定できる値には、次のような

ものがあります。

属性値	意味
WS_LEFT	左寄
WS_RIGHT	右寄
WS_CENTER	中央
WS_TOP	上寄
WS_BOTTOM	下寄
WS_LEFT_TOP	左上寄
WS_LEFT_BOTTOM	左下寄
WS_RIGHT_BOTTOM	右下寄
WS_RIGHT_TOP	右上寄

エンコーディングに指定できる値には、次のようなものがあります。省略すると WS_EN_DEFAULT が指定されます。

属性値	意味
WS_EN_DEFAULT	現在の設定を指定 (省略時の値)
WS_EN_LOCALE	現在の LANG 環境変数の設定を指定
WS_EN_NONE	設定を行わない
WS_EN_ISO8859_1	ISO8859(1) を指定
WS_EN_ISO8859_2	ISO8859(2) を指定
WS_EN_ISO8859_3	ISO8859(3) を指定
WS_EN_ISO8859_4	ISO8859(4) を指定
WS_EN_ISO8859_5	ISO8859(5) を指定
WS_EN_ISO8859_6	ISO8859(6) を指定
WS_EN_ISO8859_7	ISO8859(7) を指定
WS_EN_ISO8859_8	ISO8859(8) を指定
WS_EN_ISO8859_9	ISO8859(9) を指定
WS_EN_ISO8859_10	ISO8859(10) を指定
WS_EN_ISO8859_11	ISO8859(11) を指定
WS_EN_ISO8859_12	ISO8859(12) を指定
WS_EN_ISO8859_13	ISO8859(13) を指定
WS_EN_ISO8859_14	ISO8859(14) を指定
WS_EN_ISO8859_15	ISO8859(15) を指定
WS_EN_UTF8	UTF8 を指定
WS_EN_KOI8R	KOI8R を指定
WS_EN_EUCJP	EUCJP を指定
WS_EN_SJIS	SJIS を指定
WS_EN_EUCKR	EUCKR を指定
WS_EN_EUCCN	EUCCN を指定
WS_EN_BIG5	BIG5 を指定

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//文字列の描画
dev->drawString(100,300,100,30,0,WS_CENTER,"ABCDE1234");
```

88.6.28 drawFillString 関数の説明

書式 long drawFillString(short x,short y,WSCushort w,WSCushort h, char font_no,char align,char* string, long encoding = WS_EN_DEFAULT);

機能 指定された文字列を指定された矩形内に背景色付き表示します。

処理概要 指定された文字列をエンコーディングに従がって指定された矩形内に表示します。

引数

(in)short x	座標X
(in)short x	座標Y
(in)WSCushort w	横幅
(in)WSCushort h	縦幅
(in)char font_no	フォント番号
(in)char align	アラインメント
(in)char* string	表示させたい文字列
(in)long encoding	文字列のエンコーディング

引数の詳細は、drawString の項を参照ください。

復帰値 WS_NO_ERR:成功、WS_ERR:失敗

注意事項 なし。

サンプル

```
//文字列の描画
dev->drawFillString(100,300,100,30,0,WS_CENTER,"ABCDE1234");
```

88.6.29 getDeviceResource 関数の説明

書式 long getDeviceResource()

機能 ウィンドウシステムの資源を取得します。

処理概要

引数 なし。

復帰値 X11 では、Display* を返します。WIN32 では、未使用です。

注意事項 なし。

サンプル

```
//X11 では Display* です。
long val = dev->getDeviceResource();
```

88.6.30 getWindowResource 関数の説明

書式 long getWindowResource()

機能 ウィンドウ資源を取得します。

処理概要

引数 なし。

復帰値 X11 では、Window または Pixmap 、WIN32 では、HWND を返します。

注意事項 なし。

サンプル

```
//X11 では Window/Pixmap、WIN32 では、HWND です。
long val = dev->getWindowResource();
```

88.6.31 getContextResource 関数の説明

書式 long getContextResource()

機能 コンテキスト資源を取得します。

処理概要

引数 なし。

復帰値 X11 では、GC 、WIN32 では、HDC や MDC を返します。

注意事項 なし。

サンプル

```
//X11 では GC、WIN32 では、HDC/MDC です。
long val = dev->getContextResource();
```

88.6.32 getSpecialResource 関数の説明

書式 long getSpecialResource()

機能 ウィンドウ資源を取得します。

処理概要

引数 なし。

復帰値 X11 では、Widget を返します。WIN32 では未使用です。

注意事項 なし。
サンプル
```
//X11 では Widget です。
long val = dev->getSpecialResource();
```

88.6.33 getReady 関数の説明

書式 long getReady()
機能 描画可能状態か否かを取得します。
処理概要
引数 なし。
復帰値 True = 描画可能、False = 不可能。
注意事項 なし。
サンプル
```
//描画が行えるか否かを取得します。
WSCbool ready = dev->getReady();
```

89 WSDenv

89.1 継承元

次のオブジェクトを継承しています。

89.2 概要

環境変数を取得します。

89.3 機能

- 環境変数の取得
- 環境変数混じりの文字列の変換

89.4 注意事項

89.5 関数一覧

- WSDenv* WSGIappEnvironment(); :WSDenv
- char* getEnv(char*) :WSDenv
- char* getPlaneString(char*) :WSDenv

89.6 関数仕様

89.6.1 WSGIappEnvironment 関数の説明

書式 WSDenv* WSGIappEnvironment()
機能 アプリケーションにひとつ存在する環境変数オブジェクトのグローバルインスタンスを取得します。
処理概要
引数 なし。
復帰値 環境変数インスタンスへのポインタ
注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。
サンプル
```
//環境変数の値を取得するのに利用します。
char* value = WSGIappEnvironment()->getEnv("WSDIR");
```

89.6.2 getEnv 関数の説明

書式 char* getEnv(char* str);
機能 環境変数名を指定すると、その変数に設定されている値が取得できます。値が無い場合は、空文か返されます。
処理概要
引数

(in)char* str	環境変数名称

復帰値 環境変数の値
注意事項 返された値を開放してはいけません。
サンプル WSGIappEnvironment() を参照してください。

89.6.3 getPlaneString 関数の説明

書式 char* getPlaneString(char* str)
機能 環境変数混じりの文字列を与えると、環境変数を展開した文字列を返します。
処理概要
引数

(in)char* str	環境変数混じり文

与える文字列において、例えば環境変数が WSDIR の場合、
$(WSDIR)/bin/wsbuilder
の様に、$(...) で環境変数を指定します。
復帰値 環境変数の展開された文字列
注意事項 返される文字列を開放してはいけません。
サンプル

```
//環境変数混じりの文字列を展開します。
char* value = WSGIappEnvironment()->getPlaneString("\$(WSDIR)/include");
```

90 WSDfont

90.1 継承元

次のオブジェクトを継承しています。

90.2 概要

フォントクラスです。フォント名称と、フォントの高さ、幅を管理します。

90.3 機能

- フォント名称の保持
- フォントの高さ、幅の保持

90.4 注意事項

90.5 関数一覧

- char* getFontName() :WSDfont
- long getFontHeight() :WSDfont
- long getStringWidth(WSCstring*) :WSDfont
- long getStringHeight(WSCstring*) :WSDfont
- long getStringWidthUCS2(WSCushort*) :WSDfont
- long getStringHeightUCS2(WSCushort*) :WSDfont

90.6 関数仕様

90.6.1 getFontName 関数の説明

書式 char* getFontName()
機能 フォント名称を取得します。
処理概要
引数 なし。
復帰値 フォント名称
注意事項
サンプル

```
//フォント名称を取得します。
WSDfont* font = WSGIappFontSet()->getDefaultFont();
char* fname = font->getFontName();
```

90.6.2 getFontHeight 関数の説明

書式 long getFontHeight()
機能 フォントの高さを取得します。
処理概要
引数 なし。
復帰値 フォントの高さ
注意事項
サンプル

```
//フォントの高さを取得します。
WSDfont* font = WSGIappFontSet()->getDefaultFont();
long fheight = font->getFontHeight();
```

90.6.3 getStringWidth 関数の説明

書式 long getStringWidth(WSCstring* str)
機能 指定された文字列の幅を取得します。
処理概要
引数

(out)WSCstring* str	文字列

復帰値 文字列の幅
注意事項
サンプル

```
//文字列の横幅を取得します。
WSDfont* font = WSGIappFontSet()->getDefaultFont();
WSCstring str;
str = "abcde";
long swidth = font->getStringWidth(\&str);
```

90.6.4 getStringHeight 関数の説明

書式 long getStringHeight(WSCstring* str)
機能 指定された文字列の高さを取得します。
処理概要
引数

(out)WSCstring* str	文字列

復帰値 文字列の高さ
注意事項
サンプル

```
//文字列の縦幅を取得します。
WSDfont* font = WSGIappFontSet()->getDefaultFont();
WSCstring str;
str = "abcde";
long sheight = font->getStringHeight(\&str);
```

90.6.5 getStringWidthUCS2 関数の説明

書式 long getStringWidthUCS2(WSCushort* str)
機能 指定された文字列の幅を取得します。
処理概要
引数

(out)WSCushort* str	UCS2 文字列

復帰値 文字列の幅
注意事項
サンプル

```
//UCS2 文字列の横幅を取得します。
WSDfont* font = WSGIappFontSet()->getDefaultFont();
WSCushort* str = WSGFgetUCS2("abcde",WS_EN_DEFAULT);
long swidth = font->getStringWidthUCS2(str);
delete str;
```

90.6.6 getStringHeightUCS2 関数の説明

書式 long getStringHeightUCS2(WSCushort* str)
機能 指定された文字列の高さを取得します。
処理概要
引数

(out)WSCushort* str	UCS2 文字列

復帰値 文字列の高さ
注意事項
サンプル

```
//UCS2 文字列の縦幅を取得します。
WSDfont* font = WSGIappFontSet()->getDefaultFont();
WSCushort* str = WSGFgetUCS2("abcde",WS_EN_DEFAULT);
long sheight = font->getStringHeightUCS2(str);
delete str;
```

91 WSDimage

91.1 継承元

次のオブジェクトを継承しています。

91.2 概要

画像クラスです。画像ファイル名称、画像ハンドルを保持します。

91.3 機能

- 画像ファイル名称の保持
- 画像ハンドルの保持

91.4 注意事項

91.5 関数一覧

- long destroyImage() :WSDimage
- long setImageName(char* iname) :WSDimage
- WSCushort getImageWidth() :WSDimage
- WSCushort getImageHeight() :WSDimage

91.6 関数仕様

91.6.1 destroyImage 関数の説明

書式 long destroyImage()
機能 読み込んだ画像ファイルを破棄します。
処理概要
引数 なし。
復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル

```
//画像ファイル 001.jpg のイメージインスタンスを取得。
WSDimage* image = WSGIappImageSet()->getImage("001.jpg");
//一度読み込んだ内容を破棄します。
image->destroyImage();
```

91.6.2 setImageName 関数の説明

書式 long setImageName(char* iname)
機能 画像ファイル名称を設定します。
処理概要
引数

(out)char* cname	画像ファイル名称

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル

```
//画像ファイル 001.jpg のイメージインスタンスを取得。
WSDimage* image = WSGIappImageSet()->getImage("001.jpg");
//イメージインスタンスの内容が、画像ファイル 001.jpg から
//画像ファイル 002.jpg になります。
image->setImageName("002.jpg");
```

91.6.3 getImageWidth 関数の説明

書式 long getImageWidth()
機能 画像の横幅を取得します。
処理概要
引数 なし。
復帰値 画像イメージの横幅
注意事項
サンプル

```
//画像ファイル 001.jpg のイメージインスタンスを取得。
WSDimage* image = WSGIappImageSet()->getImage("001.jpg");
long width = image->getImageWidth();
long height = image->getImageHeight();
```

91.6.4 getImageHeight 関数の説明

書式 long getImageHeight()
機能 画像の縦幅を取得します。
処理概要
引数 なし。
復帰値 画像イメージの縦幅
注意事項
サンプル getImageWidth() を参照してください。

92 WSDkeyboard

92.1 継承元

次のオブジェクトを継承しています。

92.2 概要

押されたキーの取得を行います。

92.3 機能

- 押されたキーの取得
- キーの種類の確認

92.4 注意事項

92.5 関数一覧

- WSDkeyboard* WSGIappKeyboard(); :WSDkeyboard
- char* getText(long encode = WS_EN_DEFAULT) :WSDkeyboard
- void setText(char* text,long encode = WS_EN_DEFAULT) :WSDkeyboard
- long getKey() :WSDkeyboard
- void setKey(long key) :WSDkeyboard
- WSCbool withShift(); :WSDkeyboard
- WSCbool withLock(); :WSDkeyboard
- WSCbool withCntl(); :WSDkeyboard
- WSCbool withAlt(); :WSDkeyboard
- WSCbool isCursorKey(); :WSDkeyboard
- WSCbool isFuncKey(); :WSDkeyboard
- void setSelectedString(char*,long encoding = WS_EN_DEFAULT) :WSDkeyboard
- char* getSelectedString(long encoding = WS_EN_DEFAULT) :WSDkeyboard
- void setGlobalKeyHook(WSCbool(*)(long,WSCbool)) :WSDkeyboard

92.6 関数仕様

92.6.1 WSGIappKeyboard 関数の説明

書式 WSDkeyboard* WSGIappKeyboard()

機能 アプリケーションにひとつ存在するキーボードのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 キーボードインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル

```
WSDkeyboard* keyboard = WSGIappKeyboard();
//入力のあったキーコードの取得
long keycode = keyboard->getKey();
//入力文字列の取得
WSCstring str = keyboard->getText();
//キーの種類の判定
WSCbool cursor = keyboard->isCursorKey();
WSCbool funckey = keyboard->isFuncKey();
//修飾キーの状態の取得
WSCbool shift = keyboard->withShift();
WSCbool ctrl = keyboard->withCntl();
WSCbool alt = keyboard->withAlt();
WSCbool lock = keyboard->withLock();
```

92.6.2 getText 関数の説明

書式 char* getText(long encode = WS_EN_DEFAULT)

機能 キーボード入力されたキーを文字列で取得します。FEP 経由で入力された日本語等はこの関数で取得します。

処理概要

引数 なし

復帰値 キー入力された文字列

注意事項 返された文字列を開放してはいけません。
また、キーボード入力イベント WSEV_KEY_PRESS 等が発生した後でないと、文字列の取得はできません。

サンプル WSGIappKeyboard() を参照してください。

92.6.3 setText 関数の説明

書式 void setText(char* text,long encode = WS_EN_DEFAULT)

機能 キーボード入力された文字列を置き換えます。FEP 経由で入力された日本語等を変更したい場合に変更後の文字列を設定します。

処理概要

引数

(in)char* str	変更後の文字列
(in)long encoding	文字列のエンコーディング

エンコーディングに指定できる値には、WSCstring クラスのコンストラクタを参照ください。エンコーディングは省略可能です。

復帰値 なし

注意事項 WSCvifield/WSCtextField 等で入力文字列を変更する場合は、キーボードフックイベント WSEV_KEY_HOOK で起動するイベントプロシージャ内で置換します。setText() 関数ではキーコードの変更はされませんので、キーコードを変更する場合は、setKey() を使用してください。

サンプル

```
WSDkeyboard* keyboard = WSGIappKeyboard();

//入力文字列の取得
WSCstring str = keyboard->getText();

//入力テキストの変更
if (!strcmp("a",(char*)str){
  keyboard->setText("A");
}
```

92.6.4 getKey 関数の説明

書式 long getKey()

機能 キーボード入力されたキーを返します。

処理概要

引数 なし

復帰値 キー値

注意事項 有効なキー値、キー値の定義は、include/WSkeysym.h を御参照ください。

サンプル WSGIappKeyboard() を参照してください。

92.6.5 setKey 関数の説明

書式 void setKey(long key)

機能 キーボード入力されたキーを変更します。

処理概要

引数

(in)long key	変更後のキーコード

復帰値 なし

注意事項 設定可能なキー値、キー値の定義は、include/WSkeysym.h を御参照ください。WSCvifield/WSCtextField 等で入力文字列を変更する場合は、キーボードフックイベント WSEV_KEY_HOOK で起動するイベントプロシージャ内で置換します。setKey() 関数では入力テキストの変更はされませんので入力テキストを変更する場合は、setText() を利用してください。

サンプル

```
WSDkeyboard* keyboard = WSGIappKeyboard();

//入力キーコードの取得
long  key = keyboard->getKey();

//入力キーコードの変更
if (key == WSK_a){
  keyboard->setKey(WSK_A);
}
```

92.6.6 withShift 関数の説明

書式 WSCbool withShift()
機能 シフトキーが同時に押されているかどうかを調べます。
処理概要
引数 なし。
復帰値 True = 押されている、False = 押されていない
注意事項
サンプル WSGIappKeyboard() を参照してください。

92.6.7 withLock 関数の説明

書式 WSCbool withLock()
機能 キャプスロックキーが同時に押されているかどうかを調べます。
処理概要
引数 なし。
復帰値 True = 押されている、False = 押されていない
注意事項
サンプル WSGIappKeyboard() を参照してください。

92.6.8 withCntl 関数の説明

書式 WSCbool withCntl()
機能 コントロールキーが同時に押されているかどうかを調べます。
処理概要
引数 なし。
復帰値 True = 押されている、False = 押されていない
注意事項
サンプル WSGIappKeyboard() を参照してください。

92.6.9 withAlt 関数の説明

書式 WSCbool withAlt()
機能 オルトキーが同時に押されているかどうかを調べます。
処理概要
引数 なし。
復帰値 True = 押されている、False = 押されていない
注意事項
サンプル WSGIappKeyboard() を参照してください。

92.6.10 isCursorKey 関数の説明

書式 WSCbool isCursorKey()
機能 カーソル関連のキーかどうかを調べます。
処理概要 カーソルキーには、上下左右矢印キー、PageUp、PageDown、Begin、End、Next、Home などがあります。
引数 なし。
復帰値 True = カーソルキー、False = 非カーソルキー
注意事項
サンプル WSGIappKeyboard() を参照してください。

92.6.11 isFuncKey 関数の説明

書式 WSCbool isFuncKey()
機能 ファンクションキーかどうかを調べます。
処理概要 ファンクションキーに該当するものは、F1〜F12 です。
引数 なし。
復帰値 True = ファンクションキー、False = 非ファンクションキー
注意事項
サンプル WSGIappKeyboard() を参照してください。

92.6.12 setSelectedString 関数の説明

書式 void setSelectedString(char* str,long encoding = WS_EN_DEFAULT)
機能 セレクション文字列を設定します。
処理概要 セレクションデータとして文字列を設定します。
引数

(in)char* str	セレクションに設定する文字列
(in)long encoding	文字列のエンコーディング

エンコーディングに指定できる値には、WSCstring クラスのコンストラクタを参照ください。省略すると WS_EN_DEFAULT が指定されます。

復帰値 なし。
注意事項
サンプル

```
//セレクションデータとして、文字列を設定します。
WSGIappKeyboard()->setSelectedString("selected string...");
```

92.6.13 setSelectedString 関数の説明

書式 char* getSelectedString(long encoding = WS_EN_DEFAULT)
機能 設定されているセレクション文字列を取得します。
処理概要 セレクションデータとして格納されている文字列を取得します。
引数

(in)long encoding	文字列のエンコーディング

エンコーディングに指定できる値には、WSCstring クラスのコンストラクタを参照ください。省略すると WS_EN_DEFAULT が指定されます。

復帰値 セレクション文字列
注意事項
サンプル

```
//セレクションデータ文字列を取得します。
char* sstr = WSGIappKeyboard()->getSelectedString();
```

92.6.14 setGlobalKeyHook 関数の説明

書式 void setGlobalKeyHook(WSCbool(*proc)(long,WSCbool))
機能 グローバルキーフックハンドラーを設定します。
処理概要 グローバルキーフックは、アプリケーションに入力されたキーボードイベントをフックします。フックプロシージャで、False が返された場合、キーイベントは配信されることなく、捨てられます。

引数

(in)WSCbool (*proc(long,WSCbool)	フックプロシージャ

復帰値 なし。

注意事項

サンプル

```
//グローバルキーフックハンドラー関数
WSCbool keyhook(long key,WSCbool){
  if (key == WSKEY_return){
    return False; //キーイベントを捨てます。
  }
  return True; //キーを配信します。
}

void sample_proc(WSCbase* object){
  //グローバルキーフックの設定を行います。
  WSGIappKeyboard()->setGlobalKeyHook(keyhook);
}
```

93 WSDmessage

93.1 継承元

次のオブジェクトを継承しています。

93.2 概要

汎用メッセージです。プロセス間で比較的短いデータのメッセージのやりとりが行えます。

93.3 機能

- メッセージ受け取りプロシージャの設定と管理
- メッセージの送信

93.4 注意事項

93.5 関数一覧

- WSDmessage* getNewInstance(); :WSDmessage
- void setupMessage(char*,void (*proc)(char*,void*),void*) :WSDmessage
- int sendMessageEx(char* disp,char* message,char* data) :WSDmessage

93.6 関数仕様

93.6.1 getNewInstance 関数の説明

書式 WSDmessage* getNewInstance()

機能 新しいメッセージクラスのインスタンスを作成いたします。

処理概要 メッセージインスタンスは、一つのメッセージ識別文字列とメッセージ受け取り関数を設定できます。メッセージは、sendMessageEx でメッセージ識別文字列を指定して、メッセージデータを投げることができます。

引数 なし。

復帰値 メッセージインスタンスへのポインタ

注意事項

サンプル

```
//メッセージを受け取ります。
void message_proc(char* data,void* ptr){
  printf("receive data=%s\n",data);
}
```

```
//メッセージを受け取るためのセットアップを行います。
WSDmessage* message = NULL;
void init_proc(){
  message = WSDmessage::getNewInstance();
  message->setupMessage("message1",message_proc,NULL);
}

//メッセージを送ります。
void send_proc(char* send_data){
  WSDmessage::sendMessageEx(NULL,"message1",send_data);
}
```

93.6.2 setupMessage 関数の説明

書式 void setupMessage(char* index,void(*proc)(char*,void*),void*)

機能 メッセージ識別文字列と、メッセージハンドラーを設定します。

処理概要 指定されたメッセージ識別文字列でメッセージが送られた場合、このメッセージインスタンスは、そのメッセージを受け取り、メッセージハンドラーを実行します。

引数

(in)char* index	メッセージ識別文字列
(in)void (*proc)(char*,void*)	メッセージハンドラー関数

復帰値 なし。

注意事項 メッセージインスタンス１つにつき、メッセージ識別文字列は１つしか設定できません。setupMessage( ) を２度以上実行した場合は、最後に実行した setupMessage( ) の設定が有効になります。

サンプル getNewInstance() を参照してください。

93.6.3 sendMessageEx 関数の説明

書式 int sendMessageEx(char* disp,char* index,char* data)

機能 指定されたメッセージ識別文字列で、メッセージを送ります。

処理概要 disp パラメータは、X11 ウィンドウシステムの場合、ディスプレイ名を指定します。指定しない場合は、NULL でかまいません。また Windows では、指定されても無視されます。index で指定されるメッセージ識別文字列で、data で指定される文字列を送ります。

引数

(in)char* disp	ディスプレイ名
(in)char* message	メッセージ識別文字列
(in)char* data	送信する文字列データ

復帰値

注意事項

サンプル getNewInstance() を参照してください。

94 WSDmouse

94.1 継承元

次のオブジェクトを継承しています。

94.2 概要

マウスクラスです。マウスの状態と位置を管理します。

94.3 機能

- マウスの状態の保持
- マウスの位置の保持

94.4 注意事項

94.5 関数一覧

- WSDmouse* WSGIappMouse() :WSDmouse
- WSCbool getMousePosition(short* x,short* y,WSCbase* inst) :WSDmouse
- long getMouseStatus() :WSDmouse
- void setMousePosition(short x,short y) :WSDmouse
- long getMouseStatus() :WSDmouse
- long getTargetBtn() :WSDmouse
- void setMouseFocusClient(WSCbase*) :WSDmouse
- WSCbase* getMouseFocusClient() :WSDmouse

94.6 関数仕様

94.6.1 WSGIappMouse 関数の説明

書式 WSDmouse* WSGIappMouse();
機能 システムに一つ存在するマウスインスタンスを取得します。
処理概要
引数 なし。
復帰値 マウスインスタンス
注意事項 返されたインスタンスを開放してはいけません。
サンプル

```
//マウスを取得します。
WSDmouse* mouse = WSGIappMouse();
//マウスの位置を取得します。
short x,y;
mouse->getMousePosition(\&x,\&y);
```

94.6.2 getMousePosition 関数の説明

書式 WSCbool getMousePosition(short* x,short* y,WSCbase* inst);
機能 マウスの位置を取得します。
処理概要 inst を省略した場合は、スクリーン座標を取得します。inst を指定した場合は、そのインスタンスの座標系でマウスの位置を取得します。
引数

(in)short* x	X座標を格納するポインタ
(in)short* y	Y座標を格納するポインタ
(in)WSCbase* inst	インスタンス

復帰値 True = 成功、False = 失敗
注意事項
サンプル WSGIappMouse() を参照してください。

94.6.3 setMouseStatus 関数の説明

書式 long getMouseStatus()
機能 マウスの状態を設定します。
処理概要 ソフトウェア的にマウスの状態を設定します。
引数 なし。
復帰値 マウスの状態

WS_MOUSE_BTN1	ボタン 1 (左ボタン) が押されている
WS_MOUSE_BTN2	ボタン 2 (中ボタン) が押されている
WS_MOUSE_BTN3	ボタン 3 (右ボタン) が押されている
WS_MOUSE_BTN4	ボタン 4 (ホイール上) が押されている
WS_MOUSE_BTN5	ボタン 5 (ホイール下) が押されている
WS_MOUSE_SHIFT	シフトキーが押されている
WS_MOUSE_MOD	モディファイアキーが押されている
WS_MOUSE_CONTROL	コントロールキーが押されている
WS_MOUSE_LOCK	キャプスロックキーが押されている

注意事項

サンプル

```
//マウスを取得します。
WSDmouse* mouse = WSGIappMouse();
//マウスの状態を設定します。
//左ボタンが押された状態に設定します。
mouse->setMouseStatus(WS_MOUSE_BTN1);
```

94.6.4 setMousePosition 関数の説明

書式 void setMousePosition(short x,short y)

機能 マウスの位置を指定します。

処理概要

引数

short x	X座標
short y	Y座標

復帰値 なし。

注意事項

サンプル

```
//マウスを取得します。
WSDmouse* mouse = WSGIappMouse();
//マウスの位置を、100,100 に設定します。
mouse->setMousePosition(100,100);
```

94.6.5 getTargetBtn 関数の説明

書式 long getTargetBtn()

機能 最近に押されたマウスのボタンを取得します。

処理概要

引数 なし。

復帰値 ボタン識別子

WS_MOUSE_BTN1	ボタン 1 (左ボタン) が押されている
WS_MOUSE_BTN2	ボタン 2 (中ボタン) が押されている
WS_MOUSE_BTN3	ボタン 3 (右ボタン) が押されている
WS_MOUSE_BTN4	ボタン 4 (ホイール上) が押されている
WS_MOUSE_BTN5	ボタン 5 (ホイール下) が押されている
WS_MOUSE_SHIFT	シフトキーが押されている

注意事項 getMouseStatus() は、同時に押されているボタンの状態を返すのに対し、getTargetBtn() は一番最後に押されたボタンだけを返します。

サンプル

```
//マウスを取得します。
WSDmouse* mouse = WSGIappMouse();
//最近に押されたマウスのボタンを取得します。
long btn = mouse->getTargetBtn();
```

95 WSDmwindowDev

95.1 継承元

次のオブジェクトを継承しています。
WSDdev

95.2 概要

メモリ描画用クラスです。通常のウィンドウシステム資源クラス (WSDdev) とは異なりメモリ上で仮想的な資源です。メモリバッファ上に画像データを転送したり、画像を読みだしたり、画像データの操作を行うことができます。

95.3 機能

- メモリ上での画像データの操作
- 画像データのウィンドウへの転送
- ウィンドウからの画像データの転送

95.4 注意事項

95.5 関数一覧

- void getGeometry(WSCushort* w,WSCushort* h) :WSDmwindowDev
- long copyToWindow(WSDdev*,short x,short y,WSCushort w,WSCushort h,short dx,short dy) :WSDmwindowDev
- long copyFromWindow(WSDdev*,short x,short y,WSCushort w,WSCushort h,short dx,short dy) :WSDmwindowDev
- long copyToWindowWithMask(WSDdev*,short x,short y,WSCushort w,WSCushort h,short dx,short dy,WSDimage* mask) :WSDmwindowDev
- long copyToWindowWithMask(WSDdev*,short x,short y,WSCushort w,WSCushort h,short dx,short dy,WSDmwindowDev*) :WSDmwindowDev
- long initBuffer() :WSDmwindowDev
- long setBufferRGB(WSCushort x,WSCushort y,WSCuchar r,WSCuchar g,WSCuchar b) :WSDmwindowDev
- long getBufferRGB(WSCushort x,WSCushort y,WSCuchar* r,WSCuchar* g,WSCuchar* b) :WSDmwindowDev
- long putBufferToPixmap() :WSDmwindowDev

95.6 関数仕様

95.6.1 getGeometry 関数の説明

書式 void getGeometry(WSCushort* w,WSCushort* y)

機能 現在の描画領域の縦横幅を取得します。

処理概要

引数

(out)WSCushort* w	横幅
(out)WSCushort* h	縦幅

復帰値 なし。

注意事項

サンプル

```
WSDmwindowDev* mdev = WSDmwindowDev::getNewInstance();
mdev->createPixmap(200,200);
WSCushort w,h;
mdev->getGeometry(\&w,\&h) //200,200 が返されます。
```

95.6.2 copyToWindow 関数の説明

書式 long copyToWindow(WSDdev* dev,short x,short y,WSCushort w,WSCushort y,short dx,short dy)

機能 指定されたウィンドウシステム資源インスタンスの指定された領域の画像イメージをコピーして来ます。

処理概要

引数

(in)WSDdev* dev	ウィンドウシステム資源インスタンス
(in)short x	コピー元X座標
(in)short y	コピー元Y座標
(in)WSCushort w	コピー元領域の横幅
(in)WSCushort h	コピー元領域の縦幅
(in)short dx	コピー先X座標
(in)short dy	コピー先Y座標

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル 画像 001.jpg を段々浮き上がるように表示します。

```
#include <WSCcolorSet.h>
#include <WSCimageSet.h>
#include <WSCmainWindow.h>
extern WSCmainWindow* newwin000;

#include <WSDmwindowDev.h>
WSDmwindowDev* mdev = NULL;
WSDmwindowDev* mdev2 = NULL;

void btnep(WSCbase* object){
  WSDdev* dev = newwin000->getdev();
  if (mdev == NULL){
    mdev = WSDmwindowDev::getNewInstance();
    mdev2 = WSDmwindowDev::getNewInstance();
  }
  mdev->createPixmap(200,200);
  mdev->beginDraw(0,0,200,200);
  WSDimage* image = WSGIappImageSet()->getImage("001.jpg");
  mdev->drawStretchedImage(0,0,200,200,image);
  mdev->endDraw();

  mdev2->createPixmap(200,200);

  mdev->initBuffer();
  mdev2->initBuffer();

  long i,x,y;
  for(i=0;i<100; i++){
    for(x=0; x<200; x++){
      for(y=0; y<200; y++){
        WSCuchar r,g,b;
        mdev->getBufferRGB(x,y,\&r,\&g,\&b);
        r = (WSCushort)((double)(r*i)/100);
        g = (WSCushort)((double)(g*i)/100);
        b = (WSCushort)((double)(b*i)/100);
        mdev2->setBufferRGB(x,y,r,g,b);
      }
    }
    mdev2->putBufferToPixmap();
    mdev2->copyToWindow(dev,0,0,200,200,0,0);
  }
}
static WSCfunctionRegister  op("btnep",(void*)btnep);
```

95.6.3 copyFromWindow 関数の説明

書式 long copyFromWindow(WSDdev* dev,short x,short y,WSCushort w,WSCushort y,short dx,short dy)

機能 指定されたウィンドウシステム資源インスタンスへ指定された領域の画像イメージをコピーして来ます。

処理概要

引数

(in)WSDdev* dev	ウィンドウシステム資源インスタンス
(in)short x	コピー元X座標
(in)short y	コピー元Y座標
(in)WSCushort w	コピー元領域の横幅
(in)WSCushort h	コピー元領域の縦幅
(in)short dx	コピー先X座標
(in)short dy	コピー先Y座標

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
//newwin000 の表示内容をメモリ描画インスタンスにコピーします。
WSDdev* dev = newwin000->getdev();
mdev->copyFromWindow(dev,0,0,200,200,0,0);
```

95.6.4 copyToWindowWithMask 関数の説明

書式 long copyToWindowWithMask(WSDdev* dev,short x,short y,WSCushort w,WSCushort y,short dx,short dy,WSDimage* mask)

機能 指定されたウィンドウシステム資源インスタンスの指定された領域の画像イメージをコピーして来ます。

処理概要 指定された白黒でマスクで、イメージがマスクされコピーされます。

引数

(in)WSDdev* dev	ウィンドウシステム資源インスタンス
(in)short x	コピー元X座標
(in)short y	コピー元Y座標
(in)WSCushort w	コピー元領域の横幅
(in)WSCushort h	コピー元領域の縦幅
(in)short dx	コピー先X座標
(in)short dy	コピー先Y座標
(in)WSDimage* mask	マスク画像

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
//メモリ描画インスタンス mdev の内容を mask.jpg でマスクして
//newwin000 上にコピーします。
WSDdev* dev = newwin000->getdev();
WSDimage* mask = WSGIappImageSet()->getImage("mask.jpg");
mdev->copyFromWindow(dev,0,0,200,200,0,0,mask);
```

95.6.5 copyToWindowWithMask 関数の説明

書式 long copyToWindowWithMask(WSDdev* dev,short x,short y,WSCushort w,WSCushort y,short dx,short dy,WSDmwindowDev* mask)

機能 指定されたウィンドウシステム資源インスタンスの指定された領域の画像イメージをコピーして来ます。

処理概要 指定された白黒でマスクで、イメージがマスクされコピーされます。

引数

(in)WSDdev* dev	ウィンドウシステム資源インスタンス
(in)short x	コピー元X座標
(in)short y	コピー元Y座標
(in)WSCushort w	コピー元領域の横幅
(in)WSCushort h	コピー元領域の縦幅
(in)short dx	コピー先X座標
(in)short dy	コピー先Y座標
(in)WSDmwindowDev* mask	マスク画像

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項

サンプル

```
//メモリ描画インスタンス mdev の内容を
//メモリ描画インスタンス mdev2 でマスクして newwin000 上にコピーします。
WSDdev* dev = newwin000->getdev();
mdev->copyFromWindow(dev,0,0,200,200,0,0,mdev2);
```

95.6.6 initBuffer 関数の説明

書式 long initBuffer()

機能 WSDmindowDev 上に画像を入出力可能なバッファを割り当てます。

処理概要 setBufferRGB/getBufferRGB メンバー関数で画像データの操作するのに先だって呼び出す必要があります。

引数 なし。

復帰値 WS_NO_ERR= 正常、それ以外はエラー。

注意事項
サンプル copyToWindow() を参照してください。

95.6.7 setBufferRGB 関数の説明

書式 long setBufferRGB(WSCushort x,WSCushort y,WSCuchar r,WSCuchar g,WSCuchar b);
機能 指定された座標のドットを指定された RGB 値に設定します。
処理概要 setBufferRGB/getBufferRGB メンバー関数で画像データの操作するのに先だって呼び出す必要があります。
引数

(in)WSCushort x	X座標
(in)WSCushort y	Y座標
(in)WSCuchar r	赤
(in)WSCuchar g	緑
(in)WSCuchar b	青

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル copyToWindow() を参照してください。

95.6.8 getBufferRGB 関数の説明

書式 long getBufferRGB(WSCushort x,WSCushort y,WSCuchar* r,WSCuchar* g,WSCuchar* b);
機能 指定された座標のドットを指定された RGB 値を取得します。
処理概要 setBufferRGB/getBufferRGB メンバー関数で画像データの操作するのに先だって呼び出す必要があります。
引数

(in)WSCushort x	X座標
(in)WSCushort y	Y座標
(out)WSCuchar* r	赤の取得
(out)WSCuchar* g	緑の取得
(out)WSCuchar* b	青の取得

復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル copyToWindow() を参照してください。

95.6.9 putBufferToPixmap 関数の説明

書式 long putBufferToPixmap()
機能 操作された入出力可能なバッファの内容を、画像バッファに反映させます。
処理概要
引数 なし。
復帰値 WS_NO_ERR= 正常、それ以外はエラー。
注意事項
サンプル copyToWindow() を参照してください。

96 WSDtimer

96.1 継承元

次のオブジェクトを継承しています。

96.2 概要

汎用タイマーです。250ms 刻で起動します。タイマーにアクセスするには、グローバルインスタンス関数 WSGIappTimer() を使用します。

96.3 機能

- 指定されたプロシージャの一回起動
- 指定されたプロシージャの定周期起動

96.4 注意事項

96.5 関数一覧

- WSDtimer* WSGIappTimer(); :WSDtimer
- long addTriggerProc(WStimerProc,WSCuchar,void*) :WSDtimer
- long addTimerProc(WStimerProc,WSCuchar,void*) :WSDtimer
- void delTriggerProc(long); :WSDtimer
- void delTimerProc(long); :WSDtimer

96.6 関数仕様

96.6.1 WSGIappTimer 関数の説明

書式 WSDtimer* WSGIappTimer()

機能 アプリケーションにひとつ存在するタイマーのグローバルインスタンスを取得します。

処理概要

引数 なし。

復帰値 タイマーインスタンスへのポインタ

注意事項 この関数はグローバル関数であり、メンバ関数ではありませんのでご注意ください。また、返されたポインタを開放してはいけません。

サンプル addTriggerProc() を参照してください.

96.6.2 addTriggerProc 関数の説明

書式 long addTriggerProc(WStimerProc proc,WSCuchar rate,void* data)

機能 この関数でプロシージャを登録すると、指定された時間間隔後に一度だけプロシージャが起動されます。

処理概要

引数

(in)WStimerProc proc	起動するプロシージャ
(in)WSCuchar rate	時間間隔識別子
(in)void* data	プロシージャに引き渡されるデータ

rate で指定する時間間隔識別子

WS250MS	250 ms 後
WS500MS	500 ms 後
WS750MS	750 ms 後
WS1000MS	1000 ms 後
WS1250MS	1250 ms 後
WS1500MS	1500 ms 後
WS1750MS	1750 ms 後
WS2000MS	2000 ms 後
WS2250MS	2250 ms 後
WS2500MS	2500 ms 後
WS2750MS	2750 ms 後
中略	中略
WS19750MS	19750 ms 後
WS20000MS	20000 ms 後

復帰値 登録 ID

注意事項 WStimerProc は次の様に定義されています。
typedef void (*WStimerProc)(WSCuchar,void*);
下記の例では、例えば、イベントプロシージャsample_event_procedure() が WSCvbtn の様なもので起動された場合、一秒後に timer_proc() がタイマーから呼び出され Hello! と表示するものです。

サンプル

```
void timer_proc(WSCuchar clock,void* data){
//一回起動の場合は、clock パラメータは使用しません。
//data は、addTriggerProc() の第３パラメータが渡って来ます。
  WSCbase* object = (WSCbase*)data;
  object->setProperty(WSNlabelString,"Hello!");
}
void sample_event_procedure(WSCbase* object){
  //グローバルインスタンス関数 WSGIappTimer() で
  //タイマーインスタンスを取得します。
  WSDtimer* timer = WSGIappTimer();
  //１秒後に、timer_proc() プロシージャを起動させます。
  timer->addTriggerProc(timer_proc,WS1000MS,object);
}
```

96.6.3 addTimerProc 関数の説明

書式 long addTimerProc(WStimerProc proc,WSCuchar rate,void* data)

機能 この関数でプロシージャを登録すると、指定された時間間隔の定周期でプロシージャが起動されます。

処理概要

引数

(in)WStimerProc proc	起動するプロシージャ
(in)WSCuchar rate	時間間隔識別子
(in)void* data	プロシージャに引き渡されるデータ

rate で指定する時間間隔識別子

WS250MS	250 ms
WS500MS	500 ms
WS750MS	750 ms
WS1000MS	1000 ms
WS1250MS	1250 ms
WS1500MS	1500 ms
WS1750MS	1750 ms
WS2000MS	2000 ms
WS2250MS	2250 ms
WS2500MS	2500 ms
WS2750MS	2750 ms
中略	中略
WS19750MS	19750 ms
WS20000MS	20000 ms

復帰値 登録 ID

注意事項 WStimerProc は次の様に定義されています。
typedef void (*WStimerProc)(WSCuchar,void*);
下記の例では、例えば、イベントプロシージャsample_event_procedure() が WSCvbtn の様なもので起動された場合、一秒毎に timer_proc() がタイマーから呼び出され、数字がカウントアップされていきます。

サンプル

```
void timer_proc(WSCuchar clock,void* data){
//clock パラメータは 250MS 単位でカウントアップされます。
//時間経過を見るのに使います。
//data は、addTimerProc() の第３パラメータが渡って来ます。
static int cnt = 0;
  cnt++;
  WSCbase* object = (WSCbase*)data;
  object->setProperty(WSNlabelString,cnt);
}
void sample_event_procedure(WSCbase* object){
  //グローバルインスタンス関数 WSGIappTimer() で
  //タイマーインスタンスを取得します。
  WSDtimer* timer = WSGIappTimer();
  //１秒後に、timer_proc() プロシージャを起動させます。
  timer->addTimerProc(timer_proc,WS1000MS,object);
}
```

96.6.4 delTriggerProc 関数の説明

書式 void delTriggerProc(long id)

機能 addTriggerProc() メンバ関数の返り値である登録ＩＤを指定して、登録したタイマープロシージャを解除します。

処理概要

引数

(in)long id	登録ＩＤ

復帰値 なし。

注意事項 プロシージャが起動された後に delTriggerProc() が呼び出されても無視されます。

サンプル

```
long id = WSGIappTimer()->addTriggerProc(timer_proc,WS1000MS,object);
//タイマーを解除するには、この id で解除します。
long id = timer->delTriggerProc(id);
```

96.6.5 delTimerProc 関数の説明

書式 void delTimerProc(long id)

機能 addTimerProc() メンバ関数の返り値である登録ＩＤを指定して、登録したタイマープロシージャを解除して、定周期起動を止めます。

処理概要

引数

(in)long id	登録ＩＤ

復帰値 なし。

注意事項 無効な登録ＩＤは無視されます。

サンプル

```
long id = WSGIappTimer()->addTimerProc(timer_proc,WS1000MS,object);
//タイマーを解除するには、この id で解除します。
long id = timer->delTimerProc(id);
```