XVT C++
Quick Reference

XVT DEVELOPMENT SOLUTION FOR C++ QUICK REFERENCE

CONTENTS

XVT-Power++

Classes

Class: CApplication

Superclass: CBoss

CApplication(void);
CApplication(const CApplication& theApplication);
virtual **~CApplication**(void);
virtual void **Go**(int argc,
char *argv[],
short theMenuBarId,
short theAboutBoxId,
char *theBaseName,
char *theApplicationName,
char *theTaskTitle);
virtual void **StartUp**(void);
virtual void **ShutDown**(void);
BOOLEAN **IsMultiByte**() const;
virtual void **DoCommand**(long theCommand,
void* theData = NULL);
virtual void **DoMenuCommand**(MENU_TAG theMenuItem,
BOOLEAN isShiftKey,
BOOLEAN isControlKey);
virtual void **ChangeFont**(const CFont &theFont);
virtual void **SetUpMenus**(CMenuBar* theMenuBar);
virtual void **UpdateMenus**(CMenuBar* theMenuBar);
virtual void **DoAboutBox**(void);
virtual BOOLEAN **DoClose**(void);
virtual BOOLEAN **DoOpen**(void);
virtual BOOLEAN **DoNew**(void);
virtual BOOLEAN **DoPageSetUp**(void);
virtual BOOLEAN **DoPrint**(void);
virtual BOOLEAN **DoSave**(void);
virtual BOOLEAN **DoSaveAs**(void);
virtual void **DoKey**(const CKey&);
virtual CDocument* **FindDocument**(int theId) const;
virtual void **CloseAll**(void);
virtual int **GetNumDocuments**(void);
virtual RWOrdered* **GetDocuments**(void) const;
virtual const RWOrdered* **GetSubObjects**(void) const;
virtual const CEnvironment* **GetEnvironment**(void) const;
virtual void **DoSetEnvironment**(const CEnvironment& theNewEnvironment,
BOOLEAN isUpdate = FALSE);
virtual void **SetEnvironment**(const CEnvironment& theNewEnvironment);
virtual PRINT_RCD* **GetPrintRecord**(void) const;
virtual void **DoTimer**(long theTimerId);

virtual void **DoUser**(long theUserId,
    void* theData);
static BOOLEAN **IsInstantiated**(void);
BOOLEAN **IApplication**(CGlobalUser *theGlobalUser);
virtual void **AddDocument**(CDocument *theDocument);
virtual void **RemoveDocument**(CDocument *theDocument);
static void **SetInstantiated**(BOOLEAN isInstantiated);
virtual void **InstallFactories**(CFactoryMgr*);

Class: CApplicationFactory

Superclass: None

virtual CTaskDoc* **ConstructTaskDoc**(CApplication*,
    long theId);
virtual CTaskWin* **ConstructTaskWin**(CDocument*,
    WINDOW theXVTWindow);
virtual CPrintMgr* **ConstructPrintMgr**();
virtual CDesktop* **ConstructDesktop**(CApplication*);
virtual CResourceMgr* **ConstructResourceMgr**();
virtual CControllerMgr* **ConstructControllerMgr**();
virtual ~**CApplicationFactory**();
CApplicationFactory();

Class: CApplicationFactoryDefault

Superclass: CApplicationFactory

CApplicationFactoryDefault();
virtual CTaskDoc* **ConstructTaskDoc**(CApplication*,
    long theId);
virtual CTaskWin* **ConstructTaskWin**(CDocument*,
    WINDOW theXVTWindow);
virtual CPrintMgr* **ConstructPrintMgr**();
virtual CDesktop* **ConstructDesktop**(CApplication*);
virtual CResourceMgr* **ConstructResourceMgr**();
virtual CControllerMgr* **ConstructControllerMgr**();

Class: CArc

Superclass: CShape

CArc(CSubview* theEnclosure,
    const CPoint& theCenter,
    UNITS theHRadius,
    UNITS theVRadius,
    double theStartAngle = 0,
    double theEndAngle = 360);
CArc(CSubview* theEnclosure,
    const CRect& theRegion,
    double theStartAngle = 0,
    double theEndAngle = 360);
CArc(const CArc& theArc);
CArc& **operator** = (const CArc& theArc);
virtual **~CArc**(void);

```
BOOLEAN IArc(double theStartAngle = 0,
        double theEndAngle = 360,
        BOOLEAN isFilled = FALSE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void Draw(const CRect& theClippingRegion);
virtual void SetFilled(BOOLEAN isFilled);
virtual BOOLEAN IsFilled(void);
virtual void SetAngles(double theStartingAngle,
        double theEndingAngle);
virtual double GetStartingAngle(void);
virtual double GetEndingAngle(void);
virtual void Size(const CRect& theNewSize);
void SetDrawingPoints(const CPoint& theStartingPoint,
        const CPoint& theEndingPoint);
```

Class: CAttachment

Superclass: CDragSource

```
CAttachment(CView *theView);
CAttachment(const CAttachment& theAttachment);
CAttachment& operator = (const CAttachment& theAttachment);
virtual ~CAttachment(void);
virtual void IAttachment(BOOLEAN isDeletedUponClose = FALSE);
virtual const RWSortedVector* GetAllFitSizes(void);
virtual CRect GetBestSize(void);
virtual CRect GetFitSize(UNITS theDesiredWidth,
        UNITS theDesiredHeight);
CRect GetFitSizeFromHeight(UNITS theMaxHeight,
        UNITS theMaxWidth = 0L);
CRect GetFitSizeFromWidth(UNITS theMaxWidth,
        UNITS theMaxHeight = 0L);
CRect GetSizeToFitAttachmentPoint(CAttachmentFrame* theAttachmentFrame,
        CAttachmentFrame::AttachmentPoint theAttachmentPoint);
void Hide(void);
void Show(void);
virtual void SetSize(const CRect &theRect);
virtual CRect GetSize(void);
virtual void Attach(CAttachmentFrame* theFrame,
        CAttachmentFrame::AttachmentPoint theAttachmentPoint);
virtual void Detach(BOOLEAN hasTitleBar = TRUE);
BOOLEAN IsDetachable(void) const;
void SetDetachable(BOOLEAN canDetach);
virtual void Popup(CWindow* theWindow,
        BOOLEAN hasTitleBar = FALSE);
void PopupAtPoint(CWindow *theWindow,
        CPoint aTopLeft,
        BOOLEAN hasTitleBar = TRUE);
AttachmentState GetState(void) const;
virtual BOOLEAN IsDraggable(void) const;
void SetDragging(BOOLEAN isDraggable);
void SetTitle(const CStringRW& theNewTitle);
const CStringRW GetTitle(void) const;
CRect GetBorderSize(const CRect &theClientRect,
        BOOLEAN hasTitleBar);
CAttachmentFrame* GetAttachmentFrame(void);
```

```
CView* GetView(void) const;
virtual BOOLEAN DoDown(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoUp(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoMove(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoDouble(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual void DoDrag(long theDragCommand,
        void* theDragData,
        CWindow *theTrapWindow,
        CURSOR theDragCursor);
virtual BOOLEAN TearOff(void);
virtual CRect GetDesiredSize(void);
CPoint GetDragOffset(void);
void SetDragOffset(CPoint *theDragOffset = NULL);
void SetAttachmentPoint(CAttachmentFrame::AttachmentPoint theAttachmentPoint);
CAttachmentFrame::AttachmentPoint GetAttachmentPoint(void)const;
BOOLEAN CanAttachTo(const CWindow *theWindow);
void Redraw(void)const;
virtual void InternalAttach(CAttachmentFrame *theFrame,
        CAttachmentFrame::AttachmentPoint theAttachmentPoint);
virtual CAttachmentWindow* ConstructInternalWindow(CWindow *theEnclosure,
        CAttachment *theAttachment,
        BOOLEAN hasTitleBar);
virtual CAttachmentWindow* ConstructExternalWindow(CAttachment *theAttachment,
        BOOLEAN hasTitleBar);
```

Class: CAttachmentFrame

Superclass: CDragSink, CSubview

```
CAttachmentFrame(CWindow* theEnclosingWindow,
        AttachmentPoint theAttachmentPoints = ATTACH_TOP);
CAttachmentFrame(CSubview* theEnclosure,
        const CRect& theRegion,
        AttachmentPoint theAttachmentPoints = ATTACH_TOP);
CAttachmentFrame(const CAttachmentFrame& theAttachmentFrame);
CAttachmentFrame& operator = (const CAttachmentFrame& theAttachmentFrame);
virtual ~CAttachmentFrame(void);
virtual void AddAttachment(CAttachment* theAttachment);
virtual void RemoveAttachment(CAttachment* theAttachment,
        BOOLEAN willInvalidateAndArrange = TRUE);
virtual void SetAttachmentPoints(AttachmentPoint theAttachmentPoints);
virtual AttachmentPoint GetAttachmentPoints() const;
virtual void ArrangeAttachments(void);
virtual void SetAttachmentEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
```

```
virtual void SetAttachmentFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
void SetClientSubview(CSubview *theClientSubview);
CSubview* GetClientSubview(void) const;
void HasOpaqueClientSubview(BOOLEAN theClientSubviewDrawsBackground);
void Suspend(void);
void Resume(void);
BOOLEAN IsSuspended(void) const;
virtual BOOLEAN IsInSink(CPoint theLocation);
virtual void DoDrop(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual void DoEnter(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual void DoLeave(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual void DoDrag(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual CView* GetOwner();
virtual void Draw(const CRect& theClippingRegion);
virtual void DoSize(const CRect& theNewSize);
virtual void SetEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetGlue(GLUETYPE theGlue);
virtual void DoSetDragging(BOOLEAN isDraggable);
virtual void DoSetSizing(BOOLEAN isSizable);
void GetAttachmentSizes(RWOrdered &theList
        AttachmentPoint theAttachmentPoint);
void SetAttachmentSizes(RWOrdered &theList,
        CPoint theOffset,
        AttachmentPoint theAttachmentPoint);
void RemoveWhiteSpace(AttachmentPoint theAttachmentPoint);
void SizeClientSubviewAndInvalidate(const CRect &theNewFrame);
```

Class: CAttachmentWindow

Superclass: CWindow

CAttachmentWindow(CWindow *theEnclosingWindow,
CAttachment *theAttachment,
BOOLEAN hasTitleBar);
CAttachmentWindow(CAttachment *theAttachment,
BOOLEAN hasTitleBar);
~CAttachmentWindow();
CWindow* **GetCommandWindow**(void);
void **ShowTitle**(void);
void **HideTitle**(void);
BOOLEAN **HasTitle**(void);
void **DoCommand**(long theCommand,
void* theData = NULL);
void **DoMenuCommand**(MENU_TAG theMenuItem,
BOOLEAN isShiftKey,
BOOLEAN isControlKey);
CMenuBar* **GetMenuBar**(void);
BOOLEAN **Close**(void);
void **Size**(const CRect &theNewSize);
void **SizeWindow**(int theWidth,
int theHeight);
void **Draw**(const CRect& theClippingRegion);
void **DoActivateWindow**(void);
void **DoDeactivateWindow**(void);
void **DoMouseDouble**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
void **DoMouseDown**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
void **DoMouseUp**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
void **DoMouseMove**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
virtual void **DoKey**(const CKey&);
void **Initialize**(void);
CRect **GetClientRect**(void);
virtual void **DrawCloseBox**(void);
virtual void **DrawTitleBar**(const CRect &theClipRect);
static CRect **GetBorderSize**(const CRect &theClientRect,
BOOLEAN hasTitleBar,
CUnits* theUnits);
void **CalculateSize**(CRect &theNewSize);
CRect **GetDesiredSize**(void);

Class: CBoss

Superclass: CNotifier

```
virtual void DoCommand(long theCommand,
        void* theData = NULL);
virtual void DoMenuCommand(MENU_TAG theMenuItem,
        BOOLEAN isShiftKey,
        BOOLEAN isControlKey);
virtual void ChangeFont(const CFont &theFont);
virtual void BroadcastCommand(long theCommand,
        void* theData = NULL,
        BOOLEAN isDeep = FALSE);
virtual void BroadcastMenuCommand(MENU_TAG theTag,
        BOOLEAN isShift,
        BOOLEAN isCtrl,
        BOOLEAN isDeep = FALSE);
virtual void BroadcastTimer(long theTimerId,
        BOOLEAN isDeep = FALSE);
virtual void BroadcastUser(long theUserId,
        void* theData
        BOOLEAN isDeep = FALSE);
BOOLEAN IsBroadcasting() const;
virtual void DoTimer(long theTimerId);
virtual void DoUser(long theUserId,
        void* theData);
virtual CUnits* GetUnits(void) const;
virtual void SetUnits(CUnits* theCoordinateUnits);
static CUnits* GetCreationUnits(void);
static void SetCreationUnits(CUnits* theUnits);
virtual const RWOrdered* GetSubObjects(void)
        const = NULL;
virtual long GetId(void) const;
virtual void SetId(long theId = PWRIdBase);
CBoss (void);
CBoss (const CBoss& theBoss);
CBoss& operator = (const CBoss& theBoss);
virtual ~CBoss(void);
BOOLEAN IBoss(CGlobalUser *theGlobalUser);
virtual void UpdateUnits(CUnits* theUnits);
BOOLEAN IBoss(CGlobalClassLib *theGlobalClassLib);
```

Class: CBounds

Superclass: None

```
CBounds();
CBounds(long theTop,
        long theLeft,
        long theBottom,
        long theRight);
long Top() const;
long Left() const;
long Bottom() const;
long Right() const;
void Top(long theTop);
void Left(long theLeft);
void Bottom(long theBottom);
```

```
void Right(long theRight);
long Height() const;
long Width() const;
CCell TopLeft() const;
CCell TopRight() const;
CCell BottomLeft() const;
CCell BottomRight() const;
BOOLEAN Contains(const CCell& theBounds) const;
BOOLEAN IsEqual(const CBounds& theBounds) const;
int operator == (const CBounds& theBounds) const;
BOOLEAN IsEmpty() const;
CBounds Intersection(const CBounds& theBounds) const;
CBounds Union(const CBounds& theBounds) const;
CBounds Union(const CCell& theCell) const;
const CBounds& operator += (const CCell& theCell);
const CBounds& operator -= (const CCell& theCell);
```

Class: CBrush

Superclass: None

```
CBrush();
CBrush(COLOR theColor,
       PAT_STYLE thePattern);
CBrush(CBRUSH theBrush);
CBrush(const CBrush &theBrush);
CBrush & operator = (const CBrush &theBrush);
~CBrush();
operator CBRUSH () const;
CBrush & operator = (CBRUSH theBrush);
void IBrush(COLOR theColor,
       PAT_STYLE thePattern);
void SetColor(COLOR theColor);
void SetPattern(PAT_STYLE thePattern);
COLOR GetColor(void) const;
PAT_STYLE GetPattern(void) const;
```

Class: CButton

Superclass: CSubview

```
CButton(CSubview *theEnclosure,
       const CRect &theSize,
       BOOLEAN isTogglable = FALSE);
CButton(CSubview *theEnclosure,
       const CPoint &theTopLeft,
       // autosized to theText
       const CStringRW &theText
       BOOLEAN isTogglable = FALSE);
CButton(CSubview *theEnclosure,
       const CPoint &theTopLeft,
       // autosized to theImage
       const CImage &theImage,
       BOOLEAN isTogglable = FALSE);
CButton(const CButton &theButton);
CButton & operator = (const CButton &theButton);
virtual ~CButton();
```

```
void IButton(COLOR theDarkBorder = COLOR_BLACK,
        COLOR theLightShadowColor = COLOR_WHITE,
        COLOR theDarkShadowColor = COLOR_GRAY,
        COLOR theFlatColor = COLOR_LTGRAY);
virtual void MouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseDouble(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual BOOLEAN IsInButton(CPoint theLocation);
virtual void DoIn(void);
virtual void DoOut(void);
virtual void DoUp(void);
virtual void DoDown(void);
virtual void DoDraw(const CRect &theClippingRegion = MAXRect);
virtual void Draw(const CRect &theClippingRegion);
virtual void AddSubview(const CView* theSubview);
virtual void Size(const CRect& theNewSize);
virtual void Key(int theKey,
        BOOLEAN isShiftKey,
        BOOLEAN isControlKey);
virtual BOOLEAN ClassCanGetKeyFocus(void) const;
virtual CView * FindEventTarget(const CPoint &theLoc) const;
virtual CView * FindDeepSubview(const CPoint &theLoc) const;
virtual void SetTogglable(BOOLEAN isTogglable);
virtual BOOLEAN IsTogglable(void) const;
virtual ToggleState GetToggleState(void) const;
virtual void SetToggleState(ToggleState theState);
virtual void SetEnabled(BOOLEAN isEnabled,
        BOOLEAN isUpdated);
virtual void SizeToFit(void);
void SetCommands(long theUpCommand = NULLcmd,
        long theDownCommand = NULLcmd,
        long theInCommand = NULLcmd,
        long theOutCommand = NULLcmd);
void SetDownCommand(long theCommand);
long GetDownCommand(void) const;
void SetUpCommand(long theCommand);
long GetUpCommand(void) const;
void SetInCommand(long theCommand);
long GetInCommand(void) const;
void SetOutCommand(long theCommand);
long GetOutCommand(void) const;
virtual int GetBorderPixels(void) const;
virtual int GetTopLeftBorderPixels(void) const;
```

virtual void **DoCommand**(long theCommand,
    void* theData = NULL);
virtual void **DoUpdateModel**(long theControllerId,
    long theCommand,
    const CModel* theModel);
void **CopyInternals**(const CButton &theButton);
void **SubviewsDoSetOrigin**(int theHOffset,
    int theVOffset);
virtual BOOLEAN **NeedsDoOut**(void);
virtual vo id **InternalDoDraw**(const CRect &theClippingRegion = MAXRect);
virtual void **IndicateIn**(void);
virtual void **DrawBorder**(void);
virtual void **DrawDisabled**(void);
virtual void **DrawUp**(void);
virtual void **DrawDown**(void);
virtual void **DrawPressed**(void);
virtual void **DrawUnknown**(void);
virtual void **DoUpAction**(void);
virtual void **DoDownAction**(void);
virtual void **DoInAction**(void);
virtual void **DoOutAction**(void);
virtual void **InternalUp**(void);
virtual void **InternalDown**(void);

Class: CCell

Superclass: None

CCell();
CCell(const CCell& theCell);
CCell(long theH,
    long theV);
long **V**() const;
long **H**() const;
void **V**(long theV);
void **H**(long theH);
BOOLEAN **IsEqual**(const CCell& theCell) const;
int **operator ==** (const CCell& theCell) const;
int **operator !=** (const CCell& theCell) const;
CCell& **operator +=** (const CCell& theCell);
CCell& **operator -=** (const CCell& theCell);
CCell **operator** + (const CCell& theCell) const;
CCell **operator** - (const CCell& theCell) const;

Class: CCircle

Superclass: COval

CCircle(CSubview* theEnclosure,
    const CPoint& theCenterPoint,
    UNITS theRadius);
CCircle(const CCircle& theCircle);
CCircle& **operator =** (const CCircle& theCircle);
virtual **~CCircle**();
BOOLEAN **ICircle**(BOOLEAN isVisible,
    GLUETYPE Glue);
virtual void **Size**(const CRect& aRect);

Class: CClipboard

Superclass: None

CClipboard(CBMode theMode = CB_WRITE);
~CClipboard( );
CClipboardTextIStream * **GetTextIStream**( );
CClipboardTextOStream * **GetTextOStream**( );
CClipboardAppIIStream * **GetAppIIStream**(const char * theName);
CClipboardAppIOStream * **GetAppIOStream**(const char * theName);
CClipboardPictIStream * **GetPictIStream**( );
CClipboardPictOStream * **GetPictOStream**( );
static BOOLEAN **HasFormat**(CB_FORMAT format,
char * theName = NULL);
void **Flush**( );
CBMode **GetMode**( );

Class: CClipboardAppIIStream

Superclass: RWbistream

virtual int **eof**();
virtual int **fail**();
virtual int **bad**();
virtual int **good**();
virtual int **rdstate**();
virtual void **clear**(int v = 0);
virtual int **get**();
virtual RWvistream& **get**(char&);
virtual RWvistream& **get**(char*, size_t);
virtual RWvistream& **get**(double*, size_t);
virtual RWvistream& **get**(float*, size_t);
virtual RWvistream& **get**(int*, size_t);
virtual RWvistream& **get**(unsigned int*, size_t);
virtual RWvistream& **get**(long*, size_t);
virtual RWvistream& **get**(unsigned long*, size_t);
virtual RWvistream& **get**(short*, size_t);
virtual RWvistream& **get**(unsigned short*, size_t);
virtual RWvistream& **getString**(char* s, size_t maxlen);
virtual RWvistream& **operator**(char&);
virtual RWvistream& **operator**(double&);
virtual RWvistream& **operator**(float&);
virtual RWvistream& **operator**(int&);
virtual RWvistream& **operator**(long&);
virtual RWvistream& **operator**(short&);
virtual RWvistream& **operator**(unsigned int&);
virtual RWvistream& **operator**(unsigned long&);
virtual RWvistream& **operator**(unsigned short&);

Class: CClipboardApplOStream

Superclass: RWbostream

virtual **~RWbostream**();
virtual int **eof**();
virtual int **fail**();
virtual int **bad**();
virtual int **good**();
virtual int **rdstate**();
virtual void **clear**(int v = 0);
virtual RWvostream& **putString**(const char*, size_t);
virtual RWvostream& **operator<<**(const char*);
virtual RWvostream& **operator<<**(char);
virtual RWvostream& **operator<<**(double);
virtual RWvostream& **operator<<**(float);
virtual RWvostream& **operator<<**(int);
virtual RWvostream& **operator<<**(unsigned int);
virtual RWvostream& **operator<<**(long);
virtual RWvostream& **operator<<**(unsigned long);
virtual RWvostream& **operator<<**(short);
virtual RWvostream& **operator<<**(unsigned short);
virtual RWvostream& **flush**();
virtual RWvostream& **put**(char);
virtual RWvostream& **put**(const char* p, size_t N);
virtual RWvostream& **put**(const short*, size_t);
virtual RWvostream& **put**(const unsigned short*, size_t);
virtual RWvostream& **put**(const int*, size_t);
virtual RWvostream& **put**(const unsigned int*, size_t);
virtual RWvostream& **put**(const long*, size_t);
virtual RWvostream& **put**(const unsigned long*, size_t);
virtual RWvostream& **put**(const float*, size_t);
virtual RWvostream& **put**(const double*, size_t);

Class: CClipboardPictIStream

Superclass: None

CClipboardPictIStream& **operator**(PICTURE &thePict);

Class: CClipboardPictOStream

Superclass: None

CClipboardPictOStream& **operator<<**(PICTURE thePict);

Class: CClipboardTextIStream

Superclass: istream

void **isfx**() { unlockbuf(); unlock(); }
istream& **operator**(istream& (__cdecl * _f)(istream&));
istream& **operator**(ios& (__cdecl * _f)(ios&));
istream& **operator**(char *);
istream& **operator**(unsigned char *);
istream& **operator**(signed char *);
istream& **operator**(char &);

istream& **operator**(unsigned char &);
istream& **operator**(signed char &);
istream& **operator**(short &);
istream& **operator**(unsigned short &);
istream& **operator**(int &);
istream& **operator**(unsigned int &);
istream& **operator**(long &);
istream& **operator**(unsigned long &);
istream& **operator**(float &);
istream& **operator**(double &);
istream& **operator**(long double &);
istream& **operator**(streambuf*);
int **get**();
istream& **get**(char *, int, char ='\n');
istream& **get**(unsigned char *, int,char ='\n');
istream& **get**(signed char *, int, char ='\n');
istream& **get**(char &);
istream& **get**(unsigned char &);
istream& **get**(signed char &);
istream& **get**(streambuf&, char ='\n');
istream& **getline**(char *, int, char ='\n');
istream& **getline**(unsigned char *, int, char ='\n');
istream& **getline**(signed char *, int, char ='\n');
istream& **ignore**(int =1, int =EOF);
istream& **read**(char *, int);
istream& **read**(unsigned char *, int);
istream& **read**(signed char *, int);
int **gcount**() const;
int **peek**();
istream& **putback**(char);
int **sync**();
istream& **seekg**(streampos);
istream& **seekg**(streamoff, ios::seek_dir);
streampos **tellg**();
void **eatwhite**();

Class: CClipboardTextOStream

Superclass: ostream

ostream& **flush**();
int **opfx**();
void **osfx**();
ostream& **operator<<**(ostream& (__cdecl * _f)(ostream&));
ostream& **operator<<**(ios& (__cdecl * _f)(ios&));
ostream& **operator<<**(const char *);
ostream& **operator<<**(const unsigned char *);
ostream& **operator<<**(const signed char *);
ostream& **operator<<**(char);
ostream& **operator<<**(unsigned char);
ostream& **operator<<**(signed char);
ostream& **operator<<**(short);
ostream& **operator<<**(unsigned short);
ostream& **operator<<**(int);
ostream& **operator<<**(unsigned int);

ostream& **operator<<**(long);
ostream& **operator<<**(unsigned long);
ostream& **operator<<**(float);
ostream& **operator<<**(double);
ostream& **operator<<**(long double);
ostream& **operator<<**(const void *);
ostream& **operator<<**(streambuf*);
ostream& **put**(char);
ostream& **put**(unsigned char);
ostream& **put**(signed char);
ostream& **write**(const char *, int);
ostream& **write**(const unsigned char *, int);
ostream& **write**(const signed char *, int);
ostream& **seekp**(streampos);
ostream& **seekp**(streamoff, ios::seek_dir);
streampos **tellp**();

Class: CColor

CColor();
CColor(COLOR theXVTColor, const CStringRW& theColorName);
CColor(COLOR theXVTColor);
CColor(const CColor& theColor);
CColor& **operator =** (const CColor& theColor);
~CColor();
operator COLOR () const;
void **IColor**(COLOR theXVTColor, const CStringRW& theColorName);
void **SetColor**(COLOR theXVTColor);
void **SetColorName**(const CStringRW& theColorName);
COLOR **GetColor**(void) const;
const CStringRW **GetColorName**(void) const;(CModel* theModel);

Class: CColorSet

CColorSet();
CColorSet(const CColorSet& theColorSet);
CColorSet& **operator =** (const CColorSet& theColorSet);
~CColorSet();
void **SetColor**(size_t theIndex, const CColor& theColor);
void **ClearColor**(size_t theIndex);
const CColor **GetColor**(size_t theIndex) const;

Class: CController

Superclass: CNotifier

(CModel* theModel);
virtual **~CController**();
virtual BOOLEAN **SetModel**(const CNotifier* theProvider,
CModel* theModel);
virtual const CModel* **GetModel**() const;
virtual BOOLEAN **DoChange**(const CNotifier* theProvider,
long theCommand,
const CModel* theModel);
virtual BOOLEAN **AddDependent**(CNotifier* theDependent);
virtual BOOLEAN **RemoveDependent**(CNotifier* theDependent);

```
virtual const RWOrdered* GetDependents() const;
virtual BOOLEAN IsDependent(CNotifier* theDependent) const;
virtual BOOLEAN AddProvider(CNotifier* theProvider);
virtual BOOLEAN RemoveProvider(CNotifier* theProvider);
virtual const RWOrdered* GetProviders() const;
virtual BOOLEAN IsProvider(CNotifier* theProvider) const;
 virtual const CNotifier* GetCurrentProvider() const;
long GetId();
CController(const CController& theController);
CController& operator = (const CController& theController);
```

Class: CControllerMgr

Superclass: CNotifier

```
CControllerMgr();
virtual ~CControllerMgr();
long Insert(CController *theController);
BOOLEAN Remove(long theControllerId,
      BOOLEAN isDelete = FALSE);
CController* Find(long theControllerId);
const RWOrdered* GetControllers() const;
CControllerMgr(const CControllerMgr& theControllerMgr);
CControllerMgr& operator = (const CControllerMgr& theControllerMgr);
```

Class: CCourse

Superclass: None

```
CCourse(int theDirection,
      Reach = kNear);
CCourse(const CCourse&);
CCourse& operator = (const CCourse&);
BOOLEAN operator == (const CCourse& theCourse) const;
void SetDirection(int);
int GetDirection() const;
Reach GetReach() const;
void SetReach(Reach);
CCourse BorderRollOver() const;
```

Class: CDesktop

Superclass: CNotifier

```
CDesktop(CApplication *theApplication);
CDesktop(const CDesktop& theDesktop);
CDesktop& operator = (const CDesktop& theDesktop);
virtual ~CDesktop(void);
virtual void SetFrontWindow(CWindow *theWindow);
virtual CWindow* GetFrontWindow(void) const;
virtual int GetNumWindows(void) const;
virtual void PlaceWindow(CWindow *theWindow);
BOOLEAN FindWindow(CWindow *theWindow) const;
virtual const RWOrdered GetWindows(void) const;
void AddDialog(CDialog* theDialog);
void RemoveDialog(CDialog* theDialog);
virtual BOOLEAN DoClose(void);
```

CView* **GetKeyFocus**(void) const;
void **Closing**(CWindow* win);
virtual void **AddWindow**(CWindow *theWindow);
virtual void **RemoveWindow**(CWindow *theWindow);
void **SetActiveWindow**(CWindow *theWindow);

Class: CDocument

Superclass: CBoss

CDocument(CApplication* theapplication,
long theid = PWRIdBase);
CDocument(const CDocument& theDocument);
CDocument& **operator** = (const CDocument& theDocument);
virtual **~CDocument**(void);
BOOLEAN **NeedsSaving**(void) const;
void **SetSave**(BOOLEAN needsSaving);
virtual BOOLEAN **IsEnvironmentShared**(void) const;
virtual const CEnvironment* **GetEnvironment**(void) const;
virtual void **SetEnvironment**(const CEnvironment& theNewEnvironment,
BOOLEAN isUpdate = FALSE);
virtual void **DoSetEnvironment**(const CEnvironment& theEnvironment,
BOOLEAN isUpdate = FALSE);
virtual void **ChangeFont**(const CFont &theFont);
virtual PRINT_RCD* **GetPrintRecord**(void) const;
virtual void **BuildWindow**(void) = 0;
virtual void **CloseAll**(void);
virtual CWindow* **FindWindow**(long theId = PWRIdBase) const;
int **GetNumWindows**(void);
const RWOrdered* **GetWindows**(void) const;
virtual void **DoCommand**(long theCommand,
void* theData = NULL);
virtual void **DoKey**(const CKey&);
virtual void **DoTimer**(long theTimerId);
virtual void **DoUser**(long theUserId,
void* theData);
virtual BOOLEAN **DoClose**(void);
virtual BOOLEAN **DoPageSetUp**(void);
virtual BOOLEAN **DoPrint**(void);
virtual BOOLEAN **DoOpen**(void);
virtual BOOLEAN **DoSave**(void);
virtual BOOLEAN **DoSaveAs**(void);
virtual BOOLEAN **DoNew**(void);
virtual void **DoMenuCommand**(MENU_TAG theMenuItem,
BOOLEAN isShiftKey,
BOOLEAN isControlKey);
virtual void **UpdateMenus**(CMenuBar* theMenuBar);
virtual const RWOrdered* **GetSubObjects**(void) const;
BOOLEAN **IDocument**();
virtual BOOLEAN **CloseWindow**(CWindow *theWindow);
virtual void **RemoveWindow**(CWindow *theWindow);
virtual void **AddWindow**(CWindow * theWindow);

Class: CDragSink

Superclass: None

```
virtual ~CDragSink();
virtual BOOLEAN IsInSink(CPoint theLocation) = 0;
virtual void DoDrop(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData) = 0;
virtual void DoEnter(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData) = 0;
```

```
virtual void DoLeave(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData) = 0;
virtual void DoDrag(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData) = 0;
virtual CView* GetOwner(void) = 0;
CDragSink(void);
```

Class: CDragSource

Superclass: CMouseHandler

```
CDragSource();
CDragSource(const CDragSource& theDragSource);
CDragSource& operator = (const CDragSource& theDragSource);
virtual ~CDragSource();
virtual void RegisterSink(CDragSink *theDragSink,
        BOOLEAN isInFront = TRUE);
virtual void UnregisterSink(CDragSink *theDragSink);
virtual void DoDrag(long theDragCommand,
        void* theDragData,
        CWindow *theTrapWindow,
        CURSOR theDragCursor = CURSOR_ARROW);
void SetSinkOrder(OrderPolicy aPolicy);
OrderPolicy GetSinkOrder() const;
virtual BOOLEAN DoUp(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoDown(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoDouble(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoMove(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN UsesGlobalCoords(void);
virtual void ShuffleSinks(void);
```

Class: CDrawingContext

Superclass: CStackable

```
CDrawingContext(CView *theView);
CDrawingContext(CWindow *theWindow,
        const CEnvironment &theEnvironment);
virtual ~CDrawingContext();
void DrawLine(const CPoint &theStartPoint,
        const CPoint &theEndPoint,
        BOOLEAN theBegArrow = FALSE,
        BOOLEAN theEndArrow = FALSE);
void DrawLine(const CPoint &theEndPoint,
        BOOLEAN theBegArrow = FALSE,
        BOOLEAN theEndArrow = FALSE);
void DrawIcon(int theIconId, const CPoint &theTopLeft);
void DrawArc(const CRect &theFrame,
        const CPoint &theStartPnt,
        const CPoint &theEndPnt);
void DrawPie(const CRect &theFrame,
        const CPoint &theStartPnt,
        const CPoint &theEndPnt);
void DrawPolygon(const RWOrdered &thePoints,
        BOOLEAN thePolyIsFilled = TRUE);
void DrawText(const CStringRW &theText,
        const CPoint &theBaseLine);
void DrawPicture(PICTURE thePict,
        const CRect &theViewRegion);
void DrawRect(const CRect &theFrame,
        short theWidthRounded = 0,
        // w & h in pixels
        short theHeightRounded = 0);
void DrawPixmap(XVT_PIXMAP thePixmap,
        const CRect &theSourceRegion,
        const CRect &theDestRegion);
void DrawImage(XVT_IMAGE theImage,
        const CRect &theSourceRegion,
        const CRect &theDestRegion);
void DrawOval(const CRect &theFrame);
void DrawOutsetFrame(const CRect &theOuterFrame,
        int theBorderWidth,
        /* in pixels */
        COLOR theUpShadow,
        COLOR theDownShadow);
void DrawInsetFrame(const CRect &theOuterFrame,
        int theBorderWidth,/* in pixels */
        COLOR theUpShadow,
        COLOR theDownShadow);
void DrawOutsetText(const CStringRW& theText,
        const CPoint& theBaseline,
        COLOR theUpShadow = COLOR_WHITE,
        COLOR theDownShadow = COLOR_GRAY);
void DrawInsetText(const CStringRW& theText,
        const CPoint& theBaseline,
        COLOR theUpShadow = COLOR_WHITE,
        COLOR theDownShadow = COLOR_GRAY);
void SetLinePos(const CPoint &theLoc);
void SetEnv(const CEnvironment &theEnvironment);
CEnvironment & GetEnv(void);
```

```
CEnvironment & operator () (void);
void SetClip(const CRect &theClipFrame = MAXRect);
CRect GetClip(void);
void SetCaretPos(const CPoint &theLoc);
void SetCaretVisible(BOOLEAN isVisible = TRUE);
void SetCaretSize(int theWidth,
        int theHeight);
void ScrollRect(const CRect &theRegion,
        int theHDist,
        int theVDist);
void InvalidateRect(const CRect &theRegion = MAXRect);
BOOLEAN IsUpdateNeeded(const CRect &theRegion);
void QueueInvalidate( const CRect& theRect );
void FlushInvalidate();
void ClearInvalidate();
void SetLocalRegionedQueue( BOOLEAN isRegioned );
static BOOLEAN IsRegionedQueue();
static void SetRegionedQueue( BOOLEAN isRegioned );
void MakeCurrent(void);
virtual void DoUpdate(void);
CDrawingContext(const CDrawingContext &);
CDrawingContext& operator = (const CDrawingContext &);
```

Class: CEnvironment

Superclass: CNotifier

```
CEnvironment(COLOR theBackground = COLOR_INVALID,
        COLOR theForeground = COLOR_INVALID,
        COLOR theBrushColor = COLOR_WHITE,
        PAT_STYLEtheBrushPattern = PAT_SOLID,
        COLOR thePenColor = COLOR_BLACK,
        PAT_STYLE thePenPattern = PAT_SOLID,
        short thePenWidth = 1,
        const CFont& theFont = STDFont,
        DRAW_MODE theDrawingMode = M_COPY
        PEN_STYLE thePenStyle = P_SOLID
        BOOLEAN theTextIsOpaque = FALSE,
        COLOR theMonoBackground = COLOR_WHITE,
        COLOR theMonoForeground = COLOR_BLACK,
        COLOR theMonoBrushColor = COLOR_WHITE,
        COLOR theMonoPenColor = COLOR_BLACK);
CEnvironment(const CFont &theFont,
        const CBrush &theBrush,
        const CPen &thePen,
        DRAW_MODE theDrawingMode = M_COPY,
        COLOR theBackground = COLOR_INVALID,
        COLOR theForeground = COLOR_INVALID,
        BOOLEAN theTextIsOpaque = FALSE,
        COLOR theMonoBackground = COLOR_WHITE,
        COLOR theMonoForeground = COLOR_BLACK,
        COLOR theMonoBrushColor = COLOR_WHITE,
        COLOR theMonoPenColor = COLOR_BLACK);
CEnvironment(DRAW_CTOOLS theTools);
CEnvironment(const CEnvironment &theEnv);
CEnvironment & operator = (const CEnvironment &theEnv);
virtual ~Environment(void);
operator DRAW_CTOOLS () const;
```

```
int operator == (const CEnvironment &theEnv) const;
int operator != (const CEnvironment &theEnv) const;
BOOLEAN IEnvironment(COLOR theBackground = COLOR_INVALID,
        COLOR theForeground = COLOR_INVALID,
        COLOR theBrushColor = COLOR_WHITE,
        PAT_STYLE theBrushPattern = PAT_SOLID,
        COLOR thePenColor = COLOR_BLACK,
        PAT_STYLE thePenPattern = PAT_SOLID,
        short thePenWidth = 1,
        const CFont& theFont = STDFont,
        DRAW_MODE theDrawingMode = M_COPY
        PEN_STYLE thePenStyle = P_SOLID
        BOOLEAN theTextIsOpaque = FALSE,
        COLOR theMonoBackground = COLOR_WHITE,
        COLOR theMonoForeground = COLOR_BLACK,
        COLOR theMonoBrushColor = COLOR_WHITE,
        COLOR theMonoPenColor = COLOR_BLACK);
BOOLEAN IEnvironment(const CFont &theFont,
        const CBrush &theBrush,
        const CPen &thePen,
        DRAW_MODE theDrawingMode = M_COPY,
        COLOR theBackground = COLOR_INVALID,
        COLOR theForeground = COLOR_INVALID,
        BOOLEAN theTextIsOpaque = FALSE,
        COLOR theMonoBackground = COLOR_WHITE,
        COLOR theMonoForeground = COLOR_BLACK,
        COLOR theMonoBrushColor = COLOR_WHITE,
        COLOR theMonoPenColor = COLOR_BLACK);
virtual void SetColor(COLOR theBackground = COLOR_INVALID,
        COLOR theForeground = COLOR_INVALID,
        COLOR theBrushColor = COLOR_WHITE,
        COLOR thePenColor = COLOR_BLACK,
        COLOR theMonoBackground = COLOR_WHITE,
        COLOR theMonoForeground = COLOR_BLACK,
        COLOR theMonoBrushColor = COLOR_WHITE,
        COLOR theMonoPenColor = COLOR_BLACK);
virtual void SetBackgroundColor(COLOR theColor,
        COLOR theMonoColor = COLOR_WHITE);
void SetForegroundColor(COLOR theColor,
        COLOR theMonoColor = COLOR_BLACK);
virtual void SetBlendColor(COLOR theColor);
virtual void SetBorderColor(COLOR theColor);
virtual void SetHighlightColor(COLOR theColor);
virtual void SetSelectColor(COLOR theColor);
virtual void SetTroughColor(COLOR theColor);
COLOR GetBlendColor(void) const;
COLOR GetBorderColor(void) const;
COLOR GetHighlightColor(void) const;
COLOR GetSelectColor(void) const;
COLOR GetTroughColor(void) const;
COLOR GetForegroundColor(void) const;
COLOR GetBackgroundColor(void) const;
virtual void SetNativeViewEnv(const CNativeView& theNativeView) const;
virtual void INativeViewColors(WIN_TYPE theType);
virtual void SetFont(const CFont &theFont);
const CFont & GetFont(void) const;
virtual void SetPen(const CPen &thePen);
```

virtual void **SetPen**(COLOR theColor,
short thePenWidth,
PAT_STYLE thePenPattern
PEN_STYLE thePenStyle = P_SOLID);
virtual void **SetPenPattern**(PAT_STYLE thePenPattern);
virtual void **SetPenStyle**(PEN_STYLE thePenStyle);
virtual void **SetPenWidth**(short thePenWidth);
virtual void **SetPenColor**(COLOR theColor,
COLOR theMonoColor = COLOR_BLACK);
const CPen & **GetPen**(void) const;
PAT_STYLE **GetPenPattern**(void) const;
PEN_STYLE **GetPenStyle**(void) const;
short **GetPenWidth**(void) const;
COLOR **GetPenColor**(void) const;
virtual void **SetBrush**(const CBrush &theBrush);
virtual void **SetBrush**(COLOR theColor,
PAT_STYLE theBrushPattern);
virtual void **SetBrushPattern**(PAT_STYLE theBrushPattern);
virtual void **SetBrushColor**(COLOR theColor,
COLOR theMonoColor = COLOR_WHITE);
const CBrush & **GetBrush**(void) const;
PAT_STYLE **GetBrushPattern**(void) const;
COLOR **GetBrushColor**(void) const;
virtual void **SetDrawingMode**(DRAW_MODE theDrawMode);
virtual void **SetTextOpaque**(BOOLEAN theTextIsOpaque);
DRAW_MODE **GetDrawingMode**(void) const;
BOOLEAN **IsTextOpaque**(void) const;
virtual void **SetColorOn**(BOOLEAN isColorOn);
BOOLEAN **IsColorOn**(void) const;
void **SetDrawingEnv**(const CWindow& theWindow) const;
void **InvertColors**(void);

Class: CFaceNavigator

Superclass: CNavigator

CFaceNavigator(CSubview* theEnclosure = (CSubview*)0);
CFactoryMgr

Superclass: CNotifier

CFactoryMgr();
virtual ~**CFactoryMgr**();
void* **GetFactory**(int theFactoryStackId);
void* **GetNextFactory**(int theFactoryStackId,
void* thisFactory);
void **AddFactory**(int theFactoryStackId,
void* theNewFactory);
void **RemoveFactory**(int theFactoryStackId,
void* theOldFactory);

Class: CFaceWindow

Superclass: CWindow

CFaceWindow (CDocument* theDocument,
NNotebook* theNotebook);
CFaceWindow(const CFaceWindow& theFaceWindow);

~**CFaceWindow**();
BOOLEAN **IWindow**();
virtual void **CreateFace**(NNotebook* theNotebook,
short theTab,
short thePage);
virtual void **CreateFace**(NNotebook* theNotebook,
short theTab,
short thePage,
long theId);
virtual void **DoCommand**(long theCommand,
void* theData=NULL);
virtual **NNotebook* GetNotebook**(void);
virtual void **Draw**(const CRect& theClippingRegion);

Class: CFixedGrid

Superclass: CGrid

CFixedGrid(CSubview* theEnclosure,
const CRect& theRegion,
int theNumberOfRows,
int theNumberOfColumns);
CFixedGrid(CSubview* theEnclosure,
CPoint& theTopLeftCorner,
UNITS theRowHeight,
UNITS theColumnWidth,
int theNumberOfRows,
int theNumberOfColumns);
CFixedGrid(const CFixedGrid& theGrid);
CFixedGrid& **operator** = (const CFixedGrid& theGrid);
virtual **~CFixedGrid**(void);
BOOLEAN **IFixedGrid**(BOOLEAN isClipping = TRUE,
PLACEMENT thePlacement = TOPLEFT,
BOOLEAN isGridVisible = FALSE,
POLICY theSizingPolicy = ADJUSTCellSize,
BOOLEAN isVisible = TRUE,
GLUETYPE theGlue = NULLSTICKY);
virtual CRect **GetCellSize**(int theRow,
int theColumn) const;
virtual int **GetRow**(UNITS theVerticalPoint) const;
virtual int **GetCol**(UNITS theHorizontalPoint) const;
virtual void **SizeCell**(UNITS theNewCellWidth,
UNITS theNewCellHeight);
virtual void **SetNumCells**(int theNumberOfColumns,
int theNumberOfRows,
BOOLEAN theGridWillResize = TRUE);
virtual UNITS **GetWidth**(int col = 0) const;
virtual UNITS **GetHeight**(int row = 0) const;

Class: CFixedSplitter

Superclass: CSplitter

CFixedSplitter(CSubview* theEnclosure,
const CRect& theRect,
int theNumOfPanes);
virtual **~CFixedSplitter**();
virtual void **IFixedSplitter**();

virtual void **IFixedPane**(long theId,
CView* theView,
const CRect& theMinFrame = MAXRect,
const CRect& theMaxFrame = MAXRect) = 0;
virtual CRect **GetPaneCreationSize**(int theIndex) const = 0;
virtual CFixedPane* **CreateFixedPane**(CView* thePane,
long theId,
const CRect& theMinFrame,
const CRect& theMaxFrame);
virtual void **OnAdjustPanes**(CHorizontalSplitBar* theHorSplit,
UNITS theHorAxis,
CVerticalSplitBar* theVerSplit,
UNITS theVerAxis);
virtual void **DoNormalizePanes**(CHorizontalSplitBar* theHorSplit,
UNITS theHorAxis,
CVerticalSplitBar* theVerSplit,
UNITS theVerAxis);
virtual CFixedPane* **ConstructFixedPane**();

Class: CFixedSplitter::CFixedPane

Superclass: CSplitter::CPane

const CRect& **GetMinFrame**() const;
const CRect& **GetMaxFrame**() const;

Class: CHorizontalFixedSplitter

Superclass: CFixedSplitter

CHorizontalFixedSplitter(CSubview* theEnclosure,
const CRect& theRect,
int theNumOfPanes,
CURSOR theGrabCursor = CURSOR_ARROW,
CURSOR theDragCursor = CURSOR_ARROW);
virtual **~CHorizontalFixedSplitter**();
virtual void **IFixedPane**(long theId,
CView* theView,
const CRect& theMinFrame = MAXRect,
const CRect& theMaxFrame = MAXRect);
virtual CRect **GetPaneCreationSize**(int theIndex) const;

Class: CVerticalFixedSplitter

Superclass: CFixedSplitter

CVerticalFixedSplitter(CSubview* theEnclosure,
const CRect& theRect,
int theNumOfPanes,
CURSOR theGrabCursor = CURSOR_ARROW,
CURSOR theDragCursor = CURSOR_ARROW);
virtual **~CVerticalFixedSplitter**();
virtual void **IFixedPane**(long theId,
CView* theView,
const CRect& theMinFrame = MAXRect,
const CRect& theMaxFrame = MAXRect);
virtual CRect **GetPaneCreationSize**(int theIndex) const;

Class: CFloat

Superclass: None

CFloat(float theVal = 0.0);
CFloat(const CFloat &theFloat);
CFloat & **operator** = (const CFloat &theFloat);
~CFloat();
RWBoolean **operator** == (const CFloat &theFloat) const;
operator float() const;
float **Get**(void) const;
float **Set**(float theVal);

Class: CFloatRWC

Superclass: CFloat, CNotifier

CFloatRWC();
CFloatRWC(float theVal);
CFloatRWC(const CFloat &theFloat);
CFloatRWC & **operator** = (const CFloatRWC &theFloatRWC);
virtual **~CFloatRWC**();
virtual RWspace **binaryStoreSize**() const;
virtual int **compareTo**(const RWCollectable *c) const;
virtual unsigned **hash**() const;
virtual RWBoolean **isEqual**(const RWCollectable* c) const;
virtual void **restoreGuts**(RWvistream& theStream);
virtual void **restoreGuts**(RWFile& theFile);
virtual void **saveGuts**(RWvostream& theStream) const;
virtual void **saveGuts**(RWFile& theFile) const;

Class: CFont

Superclass: None

CFont();
CFont(int theResId);
CFont(XVT_FNTID theFontId);
CFont(const CStringRW &theNativeDesc);
CFont(const CStringRW &theFamily,
    XVT_FONT_STYLE_MASK theStyle,
    long theSize);
CFont(const CFont &theFont);
CFont & **operator** = (const CFont &theFont);
~CFont();
void **IFont**(const CStringRW &theFamily,
    XVT_FONT_STYLE_MASK theStyle,
    long theSize,
    const CStringRW &theNativeDesc = NULLString);
operator XVT_FNTID (void) const;
CFont & **operator** = (XVT_FNTID theFontId);
friend BOOLEAN **operator** == (const CFont &theAFont,
    const CFont &theBFont);
friend BOOLEAN **operator** != (const CFont &theAFont,
    const CFont &theBFont);

void **GetMetrics**(int *theOutLeading,
int *theOutAscent,
int *theOutDescent) const;
int **GetLeading**(void) const;
int **GetAscent**(void) const;
int **GetDescent**(void) const;
int **GetHeight**(void) const;
long **GetTextWidth**(const CStringRW &theStr) const;
CStringRW **GetMappedFamily**(void) const;
XVT_FONT_STYLE_MASK **GetMappedStyle**(void) const;
long **GetMappedSize**(void) const;
void **SetNativeDescription**(const CStringRW &theNativeDesc);
void **SetFamily**(const CStringRW &theFamily);
void **SetStyle**(XVT_FONT_STYLE_MASK theStyle);
void **SetSize**(long theSize);
CStringRW **GetNativeDescription**(void) const;
CStringRW **GetFamily**(void) const;
XVT_FONT_STYLE_MASK **GetStyle**(void) const;
long **GetSize**(void) const;
CStringRW **Serialize**(void) const;
void **Deserialize**(const CStringRW &theSerialStr);
void **Map**(CWindow *theCWindow);
void **MapUsingDefault**(void);
void **Unmap**(void);
BOOLEAN **IsMapped**(void) const;
BOOLEAN **IsPrint**(void) const;
BOOLEAN **IsScalable**(void) const;
static const CFont & **GetBestCtlFont**(void);

Class: CGlobalClassLib

Superclass: CNotifier

CDesktop* **GetDesktop**(void);
void **SetDesktop**(CDesktop* theNewDesktop);
CPrintMgr* **GetPrintMgr**(void);
void **SetPrintMgr**(CPrintMgr* theNewPrintManager);
CApplication* **GetApplication**(void);
CControllerMgr* **GetControllerMgr**(void);
CResourceMgr* **GetResourceMgr**(void);
CTaskWin* **GetTaskWin**(void);
CWindow* **GetHiddenWindow**(void);
BOOLEAN **IsTerminating**(void);
void **SetTerminate**(BOOLEAN isTerminating);
CFont & **GetSTDFont**(void);
CRect & **GetMAXRect**(void);
CMenuItem & **GetMENUSeparator**(void);
CRegion & **GetNULLCRegion**(void);
CStringRW & **GetNULLString**(void);
CView* **GetUpdateView**(void);
void **SetUpdateView**(CView* theView);
PWRID **GetId**(void);
CFactoryMgr* **GetFactoryMgr**();
CGlobalClassLib(CApplication *theApplication);
virtual **~CGlobalClassLib**(void);
void **CreateFactoryObjects**();

Class: CGlobalUser

Superclass: CNotifier

CGlobalUser(void);

Class: CGlue

Superclass: CNotifier

CGlue(CView* theOwner);
CGlue(const CGlue& theGlue);
CGlue& **operator** = (const CGlue& theGlue);
virtual **~CGlue**(void);
virtual void **SetGlue**(GLUETYPE theGlue);
virtual GLUETYPE **GetGlue**(void) const;
virtual void **SizeOwner**(void);

Class: CGrid

Superclass: CSubview

virtual **~CGrid**(void);
BOOLEAN **IGrid**(BOOLEAN isClipping = FALSE,
PLACEMENT thePlacement = TOPLEFT,
BOOLEAN isGridVisible = FALSE,
POLICY theSizingPolicy = ADJUSTCellSize,
BOOLEAN isVisible = TRUE,
GLUETYPE theGlue = NULLSTICKY);
virtual void **DoDraw**(const CRect& theClippingRegion = MAXRect);
virtual void **Draw**(const CRect& theClippingRegion);
virtual void **SetClipping**(BOOLEAN isClipping);
virtual BOOLEAN **IsClipping**(void);
virtual void **SetPlacement**(PLACEMENT thePlacement);
virtual PLACEMENT **GetPlacement**(void) const;
virtual void **ShowGrid**(void);
virtual void **HideGrid**(void);
virtual BOOLEAN **IsGridVisible**(void);
virtual BOOLEAN **DrawsVertical**(void) const;
virtual BOOLEAN **DrawsHorizontal**(void) const;
virtual void **SetVerticalDrawing**(BOOLEAN theSetting);
virtual void **SetHorizontalDrawing**(BOOLEAN theSetting);
virtual void **SetSnapping**(BOOLEAN IsSnapping);
virtual BOOLEAN **IsSnapping**(void) const;
virtual CView* **Remove**(int theRow,
int theColumn);
virtual BOOLEAN **Remove**(const CView* theView);
void **RemoveSubview**(const CView *theView);
virtual void **Insert**(CView* theView,
int theRow,
int theColumn);
virtual void **Insert**(CView* theView);
virtual CView* **Replace**(CView* theView,
int theRow,
int theColumn);
virtual CView* **Replace**(CView* theView);
virtual CView* **GetContents**(int theRow,
int theCol) const;
virtual CView* **GetContents**(CPoint theLocation) const;
virtual int **GetNumRows**(void) const;
virtual int **GetNumCols**(void) const;

```
virtual void GetPosition(const CView* theView,
        int *theRow,
        int *theCol) const;
virtual void GetPosition(const CRect& theView,
        int *theRow,
        int *theCol) const;
virtual CRect GetCellSize(int theRow,
        int theCol) const = NULL;
virtual int GetRow(UNITS theVerticalLocation)
        const = NULL;
virtual int GetCol(UNITS theHorizontalLocation)
        const = NULL;
virtual BOOLEAN Validate(int theRow,
        int theCol);
virtual void Size(const CRect& theNewSize);
virtual void SizeCell(UNITS theCellWidth,
        UNITS theCellHeight) = 0;
virtual void SetNumCells(int theNumberOfColumns,
        int theNumberOfRows,
        BOOLEAN theGridWillResize = TRUE);
virtual void AdjustCells(ADJUST theAdjustment);
virtual UNITS GetWidth(int theColumn = 0)
        const = NULL;
virtual UNITS GetHeight(int theRow = 0) const = NULL;
virtual POLICY GetSizingPolicy(void) const;
virtual void SetSizingPolicy(POLICY thePolicy);
virtual void SetPlacementOffset(const CPoint& theNewPlacementOffset);
virtual CPoint GetPlacementOffset(void);
virtual void MouseDouble(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual CView* FindEventTarget(const CPoint& theLocation) const;
virtual CView* FindHitView(const CPoint& theLocation) const;
void Suspend(void);
void Resume(void);
BOOLEAN IsSuspended(void) const;
virtual void PlaceView(CView* theView,
        int theRow,
        int theCol);
virtual void DoPlaceView(void);
virtual void DoClear(void);
CGrid(CSubview* theEnclosure,
        const CRect& theRegion);
CGrid(const CGrid& theGrid);
CGrid& operator = (const CGrid& theGrid);
```

```
void SetDragPoint(const CPoint& theLocation);
void ResetCell(CView* theView,
        int theNewRow = -1,
        int theNewCol = -1);
CView* GetHitItem(CPoint theLocation) const;
```

Class: CHorizontalWireFrame

Superclass: CWireFrame

```
CHorizontalWireFrame(CView *theOwner,
        CSubview* theGroupEnclosure = NULL);
virtual void MouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
```

Class: CImage

Superclass: None

```
CImage();
CImage(XVT_IMAGE_FORMAT theFormat,
        short theWidth,
        short theHeight);
CImage(const CStringRW &theFileName);
CImage(XVT_IOSTREAM theIOStream);
CImage(XVT_IMAGE theImageId);
CImage(int theResId);
CImage(XVT_PIXMAP thePixmap);
CImage(const CImage &theImage);
CImage & operator = (const CImage &theImage);
~CImage();
operator XVT_IMAGE (void) const;
CImage& operator = (XVT_IMAGE theImageId);
BOOLEAN Read(const CStringRW &theFile);
BOOLEAN ReadIOStream(XVT_IOSTREAM theIOStream);
BOOLEAN Write(const CStringRW &theFile = NULLString,
        PWR_IMAGE_FILETYPE theType = DEFAULT);
BOOLEAN WriteIOStream(XVT_IOSTREAM theIOStream = NULL,
        PWR_IMAGE_FILETYPE theType = DEFAULT);
void FillRect(COLOR theColor,
        const CRect &theFrame);
void Transfer(const CRect &theDestFrame,
        const CImage &theSourceImage,
        const CRect &theSourceFrame);
void TransferPixmap(const CRect &theDestFrame,
        XVT_PIXMAP thePixmap,
        const CRect &theSourceFrame);
void SetPixel(const CPoint &theLoc,
        COLOR theColor);
COLOR GetPixel(const CPoint &theLoc) const;
XVT_IMAGE_FORMAT GetFormat(void) const;
void GetDimensions(short *theWidth,
        short *theHeight) const;
short GetWidth(void) const;
short GetHeight(void) const;
COLOR GetCLUTEntry(short theIndex) const;
```

short **GetNumColors**(void) const;
void **GetResolution**(long *theHRes,
long *theVRes) const;
long **GetHRes**(void) const;
long **GetVRes**(void) const;
void **SetCLUTEntry**(short theIndex,
COLOR theColor);
void **SetNumColors**(short theNum);
void **SetResolution**(long theHRes,
long theVRes);
void **SetHRes**(long theHRes);
void **SetVRes**(long theVRes);

Class: CKey

Superclass: None

CKey();
CKey(const CKey&);
CKey(XVT_WCHAR theKey,
BOOLEAN isShifted = FALSE,
BOOLEAN isControl = FALSE,
BOOLEAN isAlt = FALSE,
BOOLEAN isMeta = FALSE);
CKey(const xvt_event_char&);
Ckey& **operator** = (const CKey&);
CKey(eVirtualKey,
XVT_WCHAR);
BOOLEAN **operator** == (const CKey&) const;
BOOLEAN **operator** != (const CKey&) const;
static void **SetDefaultSensitivity**(unsigned theSensitivity);
static unsigned **GetDefaultSensitivity**();
BOOLEAN **IsEqual**(const CKey&,
unsigned = GetDefaultSensitivity()) const;
XVT_WCHAR **GetChar**() const;
BOOLEAN **IsShifted**() const;
BOOLEAN **IsCtrl**() const;
BOOLEAN **IsAlt**() const;
BOOLEAN **IsMeta**() const;
BOOLEAN **IsVirtualKey**() const;
const xvt_event_char& **GetKeyEventChar**() const;
BOOLEAN **IsCharEqualSensitive**(const CKey&) const;
BOOLEAN **IsCharEqualInsensitive**(const CKey&) const;

Class: CLine

Superclass: CShape

CLine (CSubview* theEnclosure,
const CPoint& theStartingPoint,
const CPoint& theEndingPoint);
CLine(const CLine& theLine);
CLine& **operator** = (const CLine& theLine);
BOOLEAN **ILine**(BOOLEAN hasBeginningArrow = FALSE,
BOOLEAN hasEndingArrow = FALSE,
BOOLEAN isVisible = TRUE,
long theGlue = NULLSTICKY);

virtual void **Draw**(const CRect& theClippingRegion);
virtual void **Size**(const CRect& theNewSize);
virtual void **Size**(const CPoint& theStartingPoint,
const CPoint& theEndingPoint);
virtual void **SetEnvironment**(const CEnvironment& theNewEnvironment,
BOOLEAN isUpdate = FALSE);
static UNITS **GetDistanceFromLine**(const CPoint &theStart,
const CPoint &theEnd,
const CPoint &thePoint);
virtual BOOLEAN **HitTest**(const CPoint &theHitLoc) const;
virtual void **SetDragging**(BOOLEAN isDraggable);
virtual void **SetSizing**(BOOLEAN isSizable);
virtual void **SetArrows**(BOOLEAN hasBeginningArrow,
BOOLEAN hasEndingArrow);
virtual BOOLEAN **HasBeginningArrow**(void);
virtual BOOLEAN **HasEndingArrow**(void);
CPoint **GetStartPoint**(void) const;
CPoint **GetEndPoint**(void) const;
CRefct **GetInitialFrame**(CSubview* theEnclosure,
const CPoint& theStPnt,
const CPoint& theEndPnt);
void **PadFrameFromPoints**(void);
void **PadPointsFromFrame**(void);

Class: CListBox

Superclass: CScroller

CListBox(CSubview* theEnclosure,
const CRect& theRegion);
CListBox(CWindow* theScrollableWindow,
UNITS theVirtualWidth = 0,
UNITS theVirtualHeight = 0);
CListBox(const CListBox& theListBox);
CListBox& **operator** = (const CListBox& theListBox);
virtual **~CListBox**(void);
BOOLEAN **IListBox**(const CStringCollection* theItems = NULL,
BOOLEAN hasHorizontalScrollBar = FALSE,
BOOLEAN hasVerticalScrollBar = TRUE,
BOOLEAN isThumbTracking = TRUE,
long theSingleClickCommand = NULLcmd,
long theDoubleClickCommand = NULLcmd,
BOOLEAN isVisible = TRUE,
long theGlue = NULLSTICKY);
virtual void **InsertLine**(const CStringRW& theItem,
int thePosition,
BOOLEAN replace = TRUE);
virtual void **RemoveLine**(int thePosition);
virtual int **GetNumberOfLines**(void) const;
virtual int **GetSelectedLine**(void) const;
virtual void **SelectLine**(int theLine);
virtual void **DeselectLine**(void);
virtual CStringRW **GetLine**(int thePosition) const;
virtual int **GetLine**(const CStringRW& theName) const;
virtual void **ScrollUntilVisible**(int thePosition);
virtual BOOLEAN **IsHeightSelfAdjusted**(void);
virtual void **SelfAdjustHeight**(BOOLEAN isHeightSelfAdjusted);

```
virtual void Key(int theKey,
        BOOLEAN isShiftKey,
        BOOLEAN isControlKey);
virtual void VScroll(SCROLL_CONTROL theEventType,
        UNITS theThumbPosition);
virtual void Size(const CRect& theNewSize);
virtual void DoSize(const CRect& theNewSize);
virtual CView* FindEventTarget(const CPoint& theLocation) const;
virtual void MouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseDouble(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoSetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetSizing(BOOLEAN isSizable);
virtual void DoSetDragging(BOOLEAN isDraggable);
virtual void SetGlue(GLUETYPE theGlue);
virtual void SetSelectedView(CView* theSelectedView);
virtual void DoCommand(long theCommand,
        void* theData = NULL);
virtual void DoUpdateModel(long theControllerId,
        long theCommand,
        const CModel* theModel);
void SizeLines(UNITS theWidth,
        UNITS theHeight);
void ResetDimensions(void);
void ScrollLines(CPoint theLocation);
CView* GetSelectedItem(int theIncrement);
void TdiNotifyItems(long theCommand,
        RWOrdered* theViews);
```

Class: CMappedSplitter

Superclass: CSplitter

CMappedSplitter(CSubview* theEnclosure,
const CRect& theRect,
SplitHorBehavior theHorBehavior = MB_HORIZONTAL_MANY,
SplitVerBehavior theVerBehavior = MB_VERTICAL_MANY,
CURSOR theHorGrabCursor = CURSOR_ARROW,
CURSOR theHorDragCursor = CURSOR_ARROW,
CURSOR theVerGrabCursor = CURSOR_ARROW,
CURSOR theVerDragCursor = CURSOR_ARROW,
CURSOR theHVGrabCursor = CURSOR_ARROW,
CURSOR theHVDragCursor = CURSOR_ARROW,
BOOLEAN theHorScrollBars = TRUE,
BOOLEAN theVerScrollBars = TRUE,
UNITS theBoxSize = 7.0F,
COLOR theBoxDarkColor = COLOR_BLACK,
COLOR theBoxLightColor = COLOR_WHITE,
COLOR theBoxBodyColor = COLOR_LTGRAY,
COLOR theBoxHighlightColor = COLOR_GRAY);
virtual **~CMappedSplitter**();
virtual void **IMappedSplitter**();
virtual void **IMappedPane**(CMappedPane* thePane,
long theId,
CView* theView);
virtual void **SplitHorizontal**();
virtual void **SplitVertical**();
virtual BOOLEAN **IsBoxTracking**() const;
virtual BOOLEAN **TrackBoxBegin**(const CPoint& theLoc);
virtual BOOLEAN **TrackBoxMove**(const CPoint& theLoc);
virtual BOOLEAN **TrackBoxEnd**(const CPoint& theLoc);
virtual BOOLEAN **TrackAutoSplit**(const CPoint& theLoc);
virtual void **DrawHorSplitBox**();
virtual void **DrawVerSplitBox**();
virtual void **DoDraw**(const CRect &theRegion = MAXRect);
virtual void **ISplitBoxes**();
virtual void **OnAdjustPanes**(CHorizontalSplitBar* theHorSplit,
UNITS theHorAxis,
CVerticalSplitBar* theVerSplit,
UNITS theVerAxis);
virtual void **DoSplitMappedPane**(UNITS theOrigin,
SplitBoxTracking theOrientation);
virtual BOOLEAN **HorSplitBoxHitTest**(const CPoint& theGlobalHit);
virtual BOOLEAN **VerSplitBoxHitTest**(const CPoint& theGlobalHit);
virtual BOOLEAN **IsMinHorLocation**(UNITS theOrigin) const;
virtual BOOLEAN **IsMaxHorLocation**(UNITS theOrigin) const;
virtual BOOLEAN **IsMinVerLocation**(UNITS theOrigin) const;
virtual BOOLEAN **IsMaxVerLocation**(UNITS theOrigin) const;
virtual CRect **AdjustClientFrame**(const CRect& theFrame,
BOOLEAN hasHorScrollBar,
BOOLEAN hasVerScrollBar);
virtual CMappedPane* **CreateMappedPane**(const CRect& theClientFrame,
NScrollBar* theHorScrollBar,
NScrollBar* theVerScrollBar);
virtual void **DestroyMappedPane**(CMappedPane* thePane);
virtual NScrollBar* **CreateHorScrollBar**(const CRect& theClientFrame,
BOOLEAN hasAdjacantVerScrollBar = FALSE);

```
virtual NScrollBar* CreateVerScrollBar(const CRect& theClientFrame,
        BOOLEAN hasAdjacantHorScrollBar = FALSE);
virtual CSplitterMouseAgent* ConstructMouseAgent();
virtual CMappedPane* ConstructMappedPane();
virtual NScrollBar* ConstructScrollBar(const CRect& theRegion,
        DIRECTION theDirection);
```

Class: CMappedSplitter::CMappedPane

Superclass: CSplitter::CPane

```
NScrollBar* GetHorScrollBar();
NScrollBar* GetVerScrollBar();
CMappedPane();
virtual ~CMappedPane();
```

Class: CMappedSplitter::CMappedSplitterMouseAgent

Superclass: CSplitter::CSplitterMouseAgent

```
CMappedSplitterMouseAgent(CMappedSplitter* theSplitter,
        short theTrackButton = 0 /* left button */);
virtual ~CMappedSplitterMouseAgent();
virtual BOOLEAN DoDown(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoUp(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoMove(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoDouble(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
```

Class: CMenu

Superclass: CNotifier

```
CMenu(void);
CMenu(const CStringRW &theText,
        MENU_TAG theTag,
        short theMnemonic,
        long theFlags = ENABLED);
CMenu(const CMenu &theMenu);
CMenu &operator = (const CMenu &theMenu);
virtual ~CMenu();
void SetTitle(const CStringRW &theText);
const CStringRW& GetTitle(void) const;
void SetTag(MENU_TAG theTag);
MENU_TAG GetTag(void) const;
void SetFlags(long theFlags);
long GetFlags(void) const;
```

```
void SetMnemonic(short theMnemonic);
short GetMnemonic(void) const;
void SetEnabled(BOOLEAN isEnabled);
BOOLEAN IsEnabled(void) const;
void SetChecked(BOOLEAN isChecked);
BOOLEAN IsChecked(void) const;
BOOLEAN IsCheckable(void) const;
```

Class: CMenuBar

Superclass: CNotifier

```
CMenuBar(CWindow *theWindow,
        MENU_TAG theMenu,
        BOOLEAN isImport = TRUE);
CMenuBar(const CMenuBar &theMenubar);
const CMenuBar &operator = (const CMenuBar &theMenuBar);
virtual ~CMenuBar();
MENU_TAG GetTag(void) const;
CWindow* GetWindow(void) const;
void DoPopup(const CPoint& theGlobalLoc,
        MENU_TAG theSubmenuPopup,
        XVT_POPUP_ALIGNMENT theAlign = XVT_POPUP_LEFT_ALIGN);
void DoPopup(const CPoint& theGlobalLoc,
        MENU_TAG theSubmenuPopup,
        MENU_TAG theOverMenuItem);
BOOLEAN AppendSubmenu(const CSubmenu &theSubmenu);
BOOLEAN InsertSubmenu(MENU_TAG theInFrontOfTag,
        const CSubmenu &theSubmenu);
BOOLEAN RemoveSubmenu(MENU_TAG theTag);
BOOLEAN ReplaceSubmenu(MENU_TAG theTag,
        const CSubmenu &theSubmenu);
CSubmenu* FindSubmenu(MENU_TAG theTag);
const RWOrdered* GetSubmenus(void) const;
void DoUpdate(void);
void SetTitle(MENU_TAG theTag,
        const CStringRW &theText);
const CStringRW& GetTitle(MENU_TAG theTag) const;
void SetEnabled(MENU_TAG theTag,
        BOOLEAN isEnabled);
BOOLEAN IsEnabled(MENU_TAG theTag) const;
void SetChecked(MENU_TAG theTag,
        BOOLEAN isChecked);
BOOLEAN IsChecked(MENU_TAG theTag) const;
BOOLEAN IsCheckable(MENU_TAG theTag) const;
BOOLEAN IsInMenu(MENU_TAG theTag) const;
static BOOLEAN IsReferenceCounted(void);
static void SetReferenceCounting(BOOLEAN isRefCounted);
CMenuBar(void);
void DestructiveHandle(void);
CMenu *FindMenu(MENU_TAG theTag);
const CMenu* FindMenu(MENU_TAG theTag) const;
MENU_ITEM* BuildMenuDef(void);
void InitMenuDefEntry(MENU_ITEM *theXVTMenu,
        CMenu *theMenu);
void InitSubmenuDefEntry(MENU_ITEM *theXVTMenu,
        CSubmenu *theSubmenu);
```

```
void InternalPopup(const CPoint& theGlobalLoc,
        XVT_POPUP_ALIGNMENT theAlign,
        MENU_TAG theSubmenuPopup,
        MENU_TAG theOverMenuItem );
```

Class: CMenuButton

Superclass: CButton

```
CMenuButton(CSubview *theEnclosure,
        const CRect &theSize,
        MENU_TAG theMenuTag);
CMenuButton(CSubview *theEnclosure,
        const CPoint & theTopLeft,
        // autosized to theText
        MENU_TAG theMenuTag,
        const CStringRW &theText);
CMenuButton(CSubview *theEnclosure,
        const CPoint &theTopLeft,
        // autosized to theImage
        MENU_TAG theMenuTag,
        const CImage &theImage);
CMenuButton(CSubview *theEnclosure,
        MENU_TAG theMenuTag,
        const CImage &theImage);
CMenuButton(CSubview *theEnclosure,
        MENU_TAG theMenuTag,
        const CStringRW &theText);
CMenuButton(const CMenuButton &theButton);
CMenuButton & operator = (const CMenuButton &theButton);
virtual ~CMenuButton();
virtual void SetTogglable(BOOLEAN isTogglable);
void SetMenuTag(MENU_TAG theMenuTag);
MENU_TAG GetMenuTag(void) const;
void SyncWithMenu(void);
virtual void Enable(void);
virtual void DoUpAction(void);
void InitMenuButton(void);
```

Class: CMenuFactory

Superclass: None

```
virtual CSubmenu* ConstructCSubmenu(const CSubmenu&);
virtual CMenuItem* ConstructCMenuItem(const CMenuItem&);
virtual CMenuBar* ConstructCMenuBar(CWindow*,
        MENU_TAG,
        BOOLEAN isImport);
virtual ~CMenuFactory();
CMenuFactory();
```

Class: CMenuFactoryDefault

Superclass: CMenuFactory

CMenuFactoryDefault();
virtual CSubmenu* **ConstructCSubmenu**(const CSubmenu&);
virtual CMenuItem* **ConstructCMenuItem**(const CMenuItem&);
virtual CMenuBar* **ConstructCMenuBar**(CWindow*,
MENU_TAG,
BOOLEAN isImport);

Class: CMenuItem

Superclass: CMenu

CMenuItem(void);
CMenuItem(const CStringRW &theText,
MENU_TAG theTag,
short theMnemonic,
long theFlags = ENABLED);
CMenuItem(const CMenuItem &theMenuItem);
CMenuItem &**operator** = (const CMenuItem &theMenuItem);
virtual **~CMenuItem**();
BOOLEAN **IsSeparator**(void) const;

Class: CModel

Superclass: CNotifier

CModel() {}
virtual **~CModel**(){}
virtual BOOLEAN **Change**(long theCommand,
const CModel* theModel) = 0;

Class: CMouseHandler

Superclass: None

CMouseHandler& **operator** = (const CMouseHandler& theMouseHandler);
virtual **~CMouseHandler**();
virtual BOOLEAN **DoDown**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey) = 0;
virtual BOOLEAN **DoUp**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey) = 0;
virtual BOOLEAN **DoDouble**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey) = 0;
virtual BOOLEAN **DoMove**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey) = 0;

```
virtual BOOLEAN DoScroll(CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey,
        XVT_INT32& theScrollX,
        XVT_INT32& theScrollY,) = 0;
virtual BOOLEAN UsesGlobalCoords(void) = 0;
```

Class: CMouseManager

Superclass: None

```
CMouseManager(CMouseManager* the Parent = NULL);
CMouseManager(const CMouseManager& theMouseManager);
CMouseManager& operator = (const CMouseManager& theMouseManager);
virtual ~CMouseManager();
void RegisterMouseHandler(CMouseHandler *theMouseHandler,
        BOOLEAN isInFront = FALSE);
void UnRegisterMouseHandler(CMouseHandler *theMouseHandler);
virtual BOOLEAN DoDown(const CWindow *theWindow,
        CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoUp(const CWindow *theWindow,
        CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoDouble(const CWindow *theWindow,
        CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoMove(const CWindow *theWindow,
        CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey);
virtual BOOLEAN DoScroll(const CWindow *theWindow,
        CPoint& theLocation,
        short& theButton,
        BOOLEAN& isShiftKey,
        BOOLEAN& isControlKey,
        XVT_INT32& theScrollX,
        XVT_INT32& theScrollY);
CMouseManager* GetParent(void) const;
void SetParent(CMouseManager* theParent);
```

Class: CNativeList

Superclass: CNativeView

```
virtual BOOLEAN Insert(const CStringCollection& theStringList,
        int thePosition = -1);
virtual BOOLEAN Insert(const CStringRW& theString,
        int thePosition = -1);
virtual BOOLEAN Remove(int thePosition);
virtual BOOLEAN Clear(void);
```

```
void Suspend(void);
void Resume(void);
virtual CStringCollection GetAllItems(void);
virtual CStringRW GetItem(int thePosition,
        int theNumberOfCharacters = ITEMLENGTH) const;
virtual int GetNumItems(void);
virtual void DoCommand(long theCommand,
        void* theData = NULL);
virtual void DoUpdateModel(long theControllerId,
        long theCommand,
        const CModel* theModel);
virtual BOOLEAN ClassCanGetKeyFocus(void) const;
virtual void SetValidator(CValidator theValidator);
CNativeList(CSubview* theEnclosure,
        const CRect& theRegion,
        WIN_TYPE theControlType,
        const CStringCollection& theItems,
        const CStringRW& theTitle,
        long theControlAttributes);
CNativeList(CSubview* theEnclosure);
CNativeList(const CNativeList& theNativeList);
CNativeList& operator = (const CNativeList& theNativeList);
virtual ~CNativeList();
void TdiNotifyItems(long theCommand,
        const CStringCollection& theValue);
```

Class: CNativeSelectList

Superclass: CNativeList

```
virtual int GetNumSelectedItems(void);
virtual CStringCollection GetSelectedItems(void);
virtual CStringRW GetFirstSelectedItem(int theNumberOfCharacters = ITEMLENGTH) const;
virtual int GetSelectPosition(void);
virtual BOOLEAN IsItSelected(int thePosition);
virtual BOOLEAN SelectItem(int thePositionOfItem);
virtual BOOLEAN SelectItem(const CStringRW& theTitle);
virtual BOOLEAN DeselectItem(int thePositionOfItem);
virtual BOOLEAN DeselectItem(const CStringRW& theTitle);
virtual BOOLEAN SelectAll(void);
virtual BOOLEAN DeselectAll(void);
virtual void DoCommand(long theCommand,
        void* theData = NULL);
virtual void DoUpdateModel(long theControllerId,
        long theCommand,
        const CModel* theModel);
CNativeSelectList(CSubview* theEnclosure,
        const CRect& theRegion,
        WIN_TYPE theControlType,
        const CStringCollection& theItems,
        const CStringRW& theTitle,
        long theControlAttributes);
CNativeSelectList(CSubview* theEnclosure);
CNativeSelectList(const CNativeSelectList& theNativeList);
CNativeSelectList& operator = (const CNativeSelectList& theNativeList);
virtual ~CNativeSelectList();
void TdiNotifySelections(void);
```

Class: CNativeTextEdit

Superclass: CView

```
virtual ~CNativeTextEdit(void);
BOOLEAN INativeTextEdit(unsigned theAttributes = TX_BORDER,
        UNITS theRightMargin = 1000,
        int theCharactedLimit = 1000,
        const CStringRW theInitialText = NULLString,
        BOOLEAN isAutoSelected = FALSE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void SetText(const CStringRW& theText);
virtual BOOLEAN Append(const CStringRW& theText);
virtual CStringRW GetText(void) const;
virtual T_CNUM GetNCharInText(void) const;
virtual void SelectText(void);
virtual CStringRW GetSelectedText(void) const;
virtual T_CNUM GetNCharInSelection(void) const;
virtual BOOLEAN Clear(void);
virtual BOOLEAN IsEmpty(void) const;
virtual void SetAttribute(unsigned theAttribute,
        BOOLEAN isSet = TRUE);
virtual unsigned long GetAttributes(void) const;
virtual void SetLimit(int theCharacterLimit);
virtual int GetLimit(void) const;
virtual void SetMargin(UNITS theRightMargin);
virtual UNITS GetMargin(void) const;
virtual void Reset(void);
virtual void Suspend(void);
virtual void Resume(void);
virtual void DoHit(CONTROL_INFO theControlInfo);
virtual void SetFocusCommands(long theKeyFocusCommand,
        long theKeyFocusLostCommand);
virtual void SetTextCommand(long theTextCommand);
long GetTextCommand( void ) const;
long GetKeyFocusCommand(void) const;
long GetKeyFocusLostCommand(void) const;
virtual int Validate(int theKey,
        BOOLEAN isShiftKey,
        BOOLEAN isControlKey);
virtual void PostValidate();
void SetTabSize(int theSize);
int GetTabSize(void) const;
virtual void Key(int theKey,
        BOOLEAN isShiftKey,
        BOOLEAN isControlKey);
virtual void SetOrigin(const CPoint& theDelta);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void Size(const CRect& theNewSize);
virtual void Show(void);
virtual void Hide(void);
virtual void Activate(void);
virtual void Deactivate(void);
virtual void Draw(const CRect& theClippingRegion);
virtual void SetGlue(GLUETYPE theGlue);
```

```
virtual BOOLEAN ClassCanGetKeyFocus(void) const;
virtual void Disable(void);
virtual void SetEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void DoCommand(long theCommand,
        void* theData = NULL);
TXEDIT GetXVTTextEdit(void) const;
CNativeTextEdit(CSubview *theEnclosure,
        const CRect& theRegion,
        unsigned theAttributes = TX_BORDER,
        UNITS theRightMargin = 1000,
        int theCharacterLimit = 1000);
CNativeTextEdit(CSubview *theEnclosure);
CNativeTextEdit(const CNativeTextEdit& theTextEdit);
CNativeTextEdit& operator = (const CNativeTextEdit& theTextEdit);
void CreateTextEdit(void);
void GetTextInternal(T_PNUM theFirstParagraph,
        T_PNUM theLastParagraph,
        T_LNUM theFirstLine,
        T_LNUM theLastLine,
        T_CNUM theFirstChar,
        T_CNUM theLastChar,
        CStringRW theTextBuffer) const;
T_CNUM GetNCharInternal(T_PNUM theFirstParagraph,
        T_PNUM theLastParagraph,
        T_LNUM theFirstLine,
        T_LNUM theLastLine,
        T_CNUM theFirstChar,
        T_CNUM theLastChar) const;
void GetFullPar(T_PNUM theParagraph,
        CStringRW theText) const;
void GetLineInternal(T_PNUM theParagraph,
        T_LNUM theLine,
        CStringRW theText) const;
void Truncate(CStringRW& theText);
void ResetColors(void);
virtual void UpdateUnits(CUnits* theUnits);
virtual void PositionTextEdit(void);
virtual void DoUpdateModel(long theControllerId,
        long theCommand,
        const CModel* theModel);
void TdiProcessEvent(EVENT* e);
void GetPartLine(T_CNUM theFirstChar,
        T_CNUM theLastChar,
        T_PNUM theParagraph,
        T_LNUM theLineNumber,
        CStringRW theText) const;
void GetPartPar(T_CNUM theStartChar,
        T_CNUM theEndChar,
        T_PNUM theParagraph,
        T_LNUM theStartLine,
        T_LNUM theEndLine,
        CStringRW theText) const;
T_CNUM GetNCharInPartPar(T_CNUM theStartChar,
        T_CNUM theEndChar,
        T_PNUM theParagraph,
        T_LNUM theStartLine,
        T_LNUM theEndLine) const;
```

Class: CNativeView

Superclass: CView

```
virtual ~CNativeView(void);
BOOLEAN INativeView(const CStringRW theTitle = NULLString,
        BOOLEAN isEnabled = TRUE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void DoHit(CONTROL_INFO theControlInfo) = NULL;
virtual void Size(const CRect& theNewSize);
virtual void SetOrigin(const CPoint& diff);
virtual void Hide(void);
virtual void Show(void);
virtual void SetId(int theId);
virtual void SetEnclosure(CSubview *theEnclosure);
virtual void SetTitle(const CStringRW& theNewTitle);
virtual const CStringRW GetTitle(void);
virtual void SetGlue(GLUETYPE theGlue);
virtual void Enable(void);
virtual void Disable(void);
virtual void Activate(void);
virtual void Deactivate(void);
virtual void SetFont(const CFont& theFont,
        BOOLEAN isUpdate = FALSE);
virtual void SetEnvironment(const CEnvironment& theEnv,
        BOOLEAN isUpdate = FALSE);
virtual void SetSizing(BOOLEAN isSizable);
virtual void SetDragging(BOOLEAN isDraggable);
WINDOW GetXVTWindow(void) const;
WIN_TYPE GetXVTType(void) const;
long GetAttributes(void) const;
void SetAttributes(long theXVTAttributes);
static void EnableSurrogateClipping(void);
static void DisableSurrogateClipping(void);
CNativeView(CSubview* theEnclosure,
        const CRect& theRegion,
        WIN_TYPE theControlType,
        long theControlAttributes = NULL;
        const CStringRW& theTitle = NULLString);
CNativeView(WINDOW theCreatedControl,
        CSubview* theEnclosure,
        const CRect& theRegion,
        WIN_TYPE theControlType);
CNativeView(const CNativeView& theControl);
CNativeView& operator = (const CNativeView& theControl);
void CreateControl(void);
virtual void InitControl(void);
virtual void UpdateUnits(CUnits* theUnits);
void Close(void);
```

Class: CNavigator

Superclass: None

```
CNavigator(CSubview* theEnclosure = NULL);
virtual ~CNavigator();
virtual CCourse GetMovement(const CKey& theKey) const;
void DefineMovement(const CKey& theKey,
        CCourse theCourse);
void SetEndJump(EndJump);
EndJump GetEndJump() const;
void SetJumpInto(JumpInto);
JumpInto GetJumpInto() const;
int TabStopCount() const;
CTabStop* GetTabStopAt(int theIndex) const;
void RemoveTabStop(CTabStop* theTabStop);
void ClearTabStops();
void InsertTabStop(int theIndex ,
        CTabStop* theTabStop);
void AppendTabStop(CTabStop* theTabStop);
void InsertSubNavigator(int theIndex,
        CNavigator* theSubNavigator);
void AppendSubNavigator(CNavigator* theSubNavigator);
void MoveTabStop(int theIndex,
        CTabStop* theTabStop);
BOOLEAN AppendSubviews(const CSubview* theEnclosure);
CTabStop* FindTabStopHavingView(const CView* theView);
void AddHotKeyToView(const CKey& theHotKey,
        CView* theView);
void InitFocus();
BOOLEAN DoKey(int theKey,
        BOOLEAN itIsShifted,
        BOOLEAN itIsControl);
BOOLEAN DoKey(const CKey&);
void DeactivateCurrentTabStop();
virtual BOOLEAN CurrentTabStopSearch(CNavigator*& theCurrentNav,
        CTabStop*& theCurrentTabStop);
virtual BOOLEAN HotKeyTabStopSearch(const CKey& theHotKey,
        CNavigator*& foundNav,
        CTabStop*& foundTabStop);
virtual void DeactivateTabStop(CTabStop* theTabStop);
virtual void ActivateTabStop(CTabStop* theTabStop);
static CNavigatorManager* Manager();
virtual BOOLEAN Navigate(const CCourse& theCourse,
        const CKey& theKey,
        CTabStop*,
        CNavigator*& aNewNav,
        CTabStop*& aNewTabStop );
BOOLEAN Navigate(const CKey&,
        CTabStop*,
        CNavigator*& aNewNav,
        CTabStop*& aNewTabStop);
const CSubview* GetEnclosure() const;
void SetEnclosure(const CSubview* theEnclosure);
virtual CTabStop* ConstructNavTabStop(CNavigator*);
virtual CTabStop* ConstructViewTabStop(CView*);
virtual NRadioNavigator* ConstructRadioNavigator(CRadioGroup*);
```

```
static RWBoolean TestTrue(void*, void*);
static RWBoolean TabStopTestViewIs(const CTabStop*,
    const CView*);
static RWBoolean TabStopTestCanActivate(const CTabStop*,
    void*);
static RWBoolean TabStopTestIsActive(const CTabStop*,
    void*);
static RWBoolean TabStopTestXvtWindowIs(const CTabStop*,
    void*);
static RWBoolean TabStopHasHotKey(const CTabStop*,
    const CKey*);
virtual BOOLEAN TestTabStop(FindDirection,
    CTabStop* theTabStop,
    CNavigator*& foundNav,
    CTabStop*& foundTabStop,
    BOOLEAN UpdateCurrentTabStop = FALSE,
    RWtestGeneric theTabStopTest = 0,
    void* theTabStopTestData = NULL,
    RWtestGeneric theNavTest = 0,
    void* theNavTestData = NULL);
virtual BOOLEAN FindTabStop(FindDirection,
    FindEntryMethod,
    CTabStop* theStart,
    CNavigator*& foundNav,
    CTabStop*& foundTabStop,
    BOOLEAN UpdateCurrentTabStop = FALSE,
    RWtestGeneric theTabStopTest = 0,
    void* = NULL,
    RWtestGeneric theNavTest = 0,
    void* = NULL);
virtual CTabStop* PickTabStop(RWtestGeneric theTabStopTest = 0,
    void* = NULL,
    RWtestGeneric theNavTest = 0,
    void* = NULL);
BOOLEAN TraverseCurrentTabStops(CNavigator*& foundNav,
    CTabStop*& foundTabStop,
    RWtestGeneric theTabStopTest = 0,
    void* = NULL);
CTabStop* FindFocusable(FindDirection,
    FindEntryMethod,
    CTabStop* theTabStop);
```

Class: CNavigatorManager

Superclass: None

```
CNavigatorManager();
virtual ~CNavigatorManager();
BOOLEAN DoKey(const CKey&);
void Register(CNavigator*);
void Unregister(CNavigator*);
CNavigator* Lookup(WINDOW theWinHandle);
```

Class: CNotifier

Superclass: CObjectRWC

CNotifier();
virtual **~CNotifie**r();
CNotifier(const CNotifier& theNotifier);
CNotifier& **operator =** (const CNotifier& theNotifier);
virtual void **DoUpdateModel**(long theControllerId,
long theCommand,
const CModel* theModel);
virtual BOOLEAN **DoChangeModel**(long theControllerId,
const CNotifer* theProvider,
long theCommand,
const CModel* theModel);
virtual CTdiController* **GetPrimaryController**();
virtual void **SetPrimaryController**(CTdiController* controller);
const CTdiValue* **GetCurrentTdiValue**() const;
void **SetCurrentTdiValue**(const CTdiValue* theValue);
void **TdiRequest**(long theCommand,
CTdiValue* theData = NULL,
CNotifier* theProvider = NULL);
BOOLEAN **CreateDependentControllers**(void);
BOOLEAN **CreateProviderControllers**(void);
void **TdiNotify**(long theTDICommand,
const CTdiValue& theData);
void **TdiNotify**(long theTDICommand,
const CStringRW& theValue);
void **TdiNotify**(long theTDICommand,
BOOLEAN theValue);
void **TdiNotify**(long theTDICommand,
int theValue);
void **TdiNotify**(long theTDICommand,
UNITS theValue);
void **TdiNotify**(long theTDICommand,
long theCommand,
CTdiValue* theData);

Class: CObjectRWC

Superclass: RWCollectable

CObjectRWC();
CObjectRWC(const CObjectRWC& theObject);
CObjectRWC& **operator =** (const CObjectRWC& theObject);
virtual **~CObjectRWC**();
static CGlobalClassLib * **GetG**(void);
static CGlobalUser * **GetGU**(void);
long **GetResourceContainerId**(void) const;
long **GetResourceId**(void) const;
virtual RWSpace **binaryStoreSize**() const;
virtual void **restoreGuts**(RWvistream& theStream);
virtual void **restoreGuts**(RWFile& theFile);
virtual void **saveGuts**(RWvostream& theStream) const;
virtual void **saveGuts**(RWFile& theFile) const;
void **SetResourceContainerId**(long theResourceContainerId);
void **SetResourceId**(long theResourceId);

```
static void SetG(CGlobalClassLib *);
static void SetGU(CGlobalUser *);
```

Class: COval

Superclass: CShape

```
COval(CSubview* theEnclosure,
      const CPoint& theCenter,
      UNITS theHRadius,
      UNITS theVRadius);
COval(CSubview* theEnclosure,
      const CRect& theRegion);
COval(const COval& theOval);
COval& operator = (const COval& theOval);
virtual ~COval(void);
BOOLEAN IOval(BOOLEAN isVisible,
      GLUETYPE theGlue);
virtual void Draw(const CRect& theClippingRegion);
```

Class: CPasswordEdit

Superclass: NEditControl

```
CPasswordEdit(CSubview* theEnclosure,
      const CRect& theRegion, // As in NEditControl
      const CStringRW& theSubstitute = "*", // Password character
      int theMaxLength = 8, // Max length of password
      const CStringRW& theTitle = NULLString, // Prompt string
      long theAttributes = 0L); // As in NEditControl
CPasswordEdit(CSubview* theEnclosure,
      long theContainerId,
      long theId,
      const CStringRW& theSubstitute = "*", // Password character
      int theMaxLength = 8); // Max length of password
CPasswordEdit(const CPasswordEdit& theEditControl);
virtual ~CPasswordEdit();
CPasswordEdit& operator=(const CPasswordEdit& theEditControl);
virtual const CStringRW GetTitle( ) const;
```

Class: CPen

Superclass: None

```
CPen();
CPen(COLOR theColor,
      PAT_STYLE thePattern,
      PEN_STYLE theStyle = P_SOLID,
      short theWidth = 1);
CPen(CPEN thePen);
CPen(const CPen &theImage);
CPen & operator = (const CPen &thePen);
~CPen();
operator CPEN () const;
CPen & operator = (CPEN thePen);
void IPen(COLOR theColor,
      PAT_STYLE thePattern,
      PEN_STYLE theStyle = P_SOLID,
      short theWidth = 1);
```

void **SetColor**(COLOR theColor);
void **SetPattern**(PAT_STYLE thePattern);
void **SetStyle**(PEN_STYLE theStyle);
void **SetWidth**(short theWidth);
COLOR **GetColor**(void) const;
PAT_STYLE **GetPattern**(void) const;
PEN_STYLE **GetStyle**(void) const;
short **GetWidth**(void) const;

Class: CPicture

Superclass: CSubview

CPicture(CSubview *theEnclosure,
const CPoint &theOrigin,
const CImage &theImage);
CPicture(const CPicture &thePicture);
CPicture & **operator =** (const CPicture &thePicture);
virtual **~CPicture**();
void **IPicture**(const CImage &theImage);
virtual void **Draw**(const CRect &theClippingRegion);
virtual void **SetEnclosure**(CSubview* theEnclosure);
virtual XVT_PIXMAP **GetPixmapFromImage**(const CImage &theImage);

Class: CPlatformFactory

Superclass: None

virtual ~**CPlatformFactory**();
virtual CControlDelegate* **ConstructNativeControlDelegate**(CNativeView*);
virtual CAttachmentWindow* **ConstructAttachmentWindow**(CWindow *theEnclosure,
CAttachment *theAttachment,
BOOLEAN hasTitleBar);
CPlatformFactory();

Class: CPlatformFactoryDefault

Superclass: CPlatformFactory

CPlatformFactoryDefault();
virtual CControlDelegate* **ConstructNativeControlDelegate**(CNativeView*);
virtual CAttachmentWindow* **ConstructAttachmentWindow**(CWindow *theEnclosure,
CAttachment *theAttachment,
BOOLEAN hasTitleBar);

Class: CPoint

Superclass: None

CPoint(void);
CPoint(UNITS theHorizontalPos,
UNITS theVerticalPos);
CPoint(const CPoint &thePoint);
CPoint & **operator =** (const CPoint &thePoint);
~CPoint();
CPoint& **SetPoint**(UNITS theHorizontalPos,
UNITS theVerticalPos);
UNITS **H**(void) const;

```
void H(UNITS theHorizontalPos);
UNITS V(void) const;
void V(UNITS theVerticalPos);
CPoint(PNT point);
operator PNT(void) const;
CPoint& operator += (const CPoint& aCPoint);
CPoint& operator -= const CPoint& CPoint);
friend CPoint operator + (const CPoint& leftPoint,
        const CPoint& rightPoint);
friend CPoint operator - (const CPoint& leftPoint,
        const CPoint& rightPoint);
BOOLEAN operator == (const CPoint& aCPoint) const;
BOOLEAN operator != (const CPoint& aCPoint) const;
CPoint Translate(const CView* fromView,
        const CView* toView);
CPoint GetTranslated(const CView* fromLocalView,
        const CView* toView) const;
CPoint Globalize(const CView* fromView);
CPoint GetGlobal(const CView* fromView) const;
CPoint Localize(const CView* toView);
CPoint GetLocal(const CView* toView) const;
```

Class: CPointRWC

Superclass: CPoint, CNotifier

```
CPointRWC();
CPointRWC(UNITS theHorizontalPos,
        UNITS theVerticalPos);
CPointRWC(const CPoint &thePoint);
CPointRWC & operator = (const CPointRWC &thePointRWC);
virtual ~CPointRWC();
virtual RWspace binaryStoreSize() const;
virtual int compareTo(const RWCollectable *c) const;
virtual unsigned hash() const;
virtual RWBoolean isEqual(const RWCollectable* c) const;
virtual void restoreGuts(RWvistream& theStream);
virtual void restoreGuts(RWFile& theFile);
virtual void saveGuts(RWvostream& theStream) const;
virtual void saveGuts(RWFile& theFile) const;
```

Class: CPolygon

Superclass: CShape

```
CPolygon(CSubview* theEnclosure,
        const RWOrdered* theListOfPoints);
CPolygon(CSubview* theEnclosure,
        const CPoint* theArrayOfPoints,
        int theNumberOfPoints);
CPolygon(const CPolygon& thePolygon);
CPolygon& operator = (const CPolygon& thePolygon);
virtual ~CPolygon(void);
BOOLEAN IPolygon(BOOLEAN isFilled = FALSE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void SetFilled(BOOLEAN isFilled);
```

```
virtual BOOLEAN IsFilled(void) const;
virtual void Draw(const CRect& theClippingRegion);
virtual void Size(const CRect& theNewSize);
virtual void SetOrigin(const CPoint& theChange);
virtual void SetDragging(BOOLEAN isDraggable);
virtual void SetSizing(BOOLEAN isSizable);
RWOrdered GetPoints(void) const;
virtual BOOLEAN HitTest(const CPoint &theHitLoc) const;
CPolygon(CSubview* theEnclosure,
        const CRect& theRegion);
void AdjustInternalPoints(void);
CRect CreatePoints(const CPoint* theArrayOfPoints,
        int theNumberOfPoints);
CRect CreatePoints(const CIdOrderedList* theListOfPoints);
CRect BoundingRect();
```

Class: CPrintMgr

Superclass: CNotifier

```
CPrintMgr(void);
virtual ~CPrintMgr(void);
virtual void Insert(const CView* theView,
        CRect& theRegion,
        int thePage = LASTPage);
virtual int Remove(const CView* theView,
        int thePage = ALLPages);
virtual const RWOrdered GetQueue(int thePage = ALLPages) const;
virtual void ClearQueue(void);
virtual BOOLEAN DoPrint(PRINT_RCD* printRecord,
        const CStringRW& theTitle = NULLString);
static BOOLEAN PrintThread(long theData);
static WINDOW GetPrintWindow(void);
```

Class: CRadioGroup

Superclass: CSubview

```
CRadioGroup(CSubview *theEnclosure,
        const CPoint& theTopLeft,
        BOOLEAN hasBorder = FALSE);
CRadioGroup(CSubview *theEnclosure,
        long theContainerId,
        long theId);
CRadioGroup(const CRadioGroup& theGroup);
CRadioGroup& operator = (const CRadioGroup& theGroup);
virtual ~CRadioGroup(void);
BOOLEAN IRadioGroup(BOOLEAN hasBorder = FALSE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual long AddButton(const CPoint &aPoint,
        const CStringRW& theButtonTitle = NULLString,
        long theCommand = NULLcmd,
        long theAttributes = 0L,
        GLUETYPE theGlue = NULLSTICKY);
```

```
virtual long AddButtons(long theFirstResourceId,
        int theNumberOfButtons,
        int theFirstCommand,
        long theAttributes = 0L,
        int theSelectedButton = 0,
        UNITS theSeparation = 0,
        DIRECTION theDirection = VERTICAL);
virtual void RemoveButton(long theButtonId);
virtual long FindResButtonId(long theContainerId,
        long theId);
virtual void SetSelectedButton(long id);
virtual long GetSelectedButton(void) const;
virtual void Draw(const CRect& theClippingRegion);
virtual void SetBorderSpace(int thePixelSpace);
virtual int GetBorderSpace(void) const;
virtual void AddResourceButtonInternal(long theContainerId,
        long theId);
long AddButtonInternal(const CPoint &aPoint,
        const CStringRW& theButtonTitle = NULLString,
        long theCommand = NULLcmd,
        long theAttributes = 0L,
        GLUETYPE theGlue = NULLSTICKY);
void CopyButtons(const CRadioGroup &theRadioGroup);
```

Class: CRadioNavigator

Superclass: CNavigator

```
CRadioNavigator(CSubview* theEnclosure = (CSubview*)0);
```

Class: CRect

Superclass: None

```
CRect(void);
CRect(UNITS theLeft,
        UNITS theTop,
        UNITS theRight,
        UNITS theBottom);
CRect(const CPoint& theTopLeftCorner,
        const CPoint& theBottomRightCorner);
CRect(const CRect &theRect);
CRect & operator = (const CRect &theRect);
~CRect();
CRect& SetRect(UNITS theLeft,
        UNITS theTop,
        UNITS theRight,
        UNITS theBottom);
CRect& SetRect(CPoint theTopLeftCorner,
        CPoint theBottomRightCorner);
CRect& Inflate(UNITS theInflation);
CRect& Inflate(UNITS theHInflation,
        UNITS theVInflation);
UNITS Left(void) const;
UNITS Left(UNITS theNewLeft);
UNITS Top(void) const;
UNITS Top(UNITS theNewTop);
UNITS Right(void) const;
```

```
UNITS Right(UNITS theNewRight);
UNITS Bottom(void) const;
UNITS Bottom(UNITS theNewBottom);
UNITS Height(void) const;
UNITS Height(UNITS theNewHeight);
UNITS Width(void) const;
UNITS Width(UNITS theNewWidth);
BOOLEAN IsPointInRect(const CPoint& thePoint) const;
BOOLEAN IsEmpty(void) const;
CRect GetInflatedRect(UNITS theInflation) const;
CRect GetInflatedRect(UNITS theHInflation,
      UNITS theVInflation) const;
CRect(const RCT& rect);
operator RCT(void) const;
BOOLEAN operator == (const CRect& aCRect) const;
BOOLEAN operator != (const CRect& aCRect) const;
CRect& operator += (const CRect& aCRect);
friend CRect operator + (const CRect& leftRect,
      const CRect& rightRect);
CRect& operator -= (const CRect& aCRect);
friend CRect operator - (const CRect& leftRect,
      const CRect& rightRect);
CRect& operator -= (const CPoint& aPoint);
CRect& operator += (const CPoint& aPoint);
friend CRect operator + (const CRect& theRect,
      const CPoint& thePoint);
friend CRect operator + (const CPoint& thePoint,
      const CRect& theRect);
friend CRect operator - (const CRect& theRect,
      const CPoint& thePoint);
CRect Translate(const CView* fromView,
      const CView* toView);
CRect GetTranslated(const CView* fromLocalView,
      const CView* toView) const;
CRect Globalize(const CView* fromView);
CRect GetGlobal(const CView* fromView) const;
CRect Localize(const CView* toView);
CRect GetLocal(const CView* toView) const;
```

Class: CRectRWC

Superclass: CRect, CNotifier

```
CRectRWC();
CRectRWC(UNITS theLeft,
      UNITS theTop,
      UNITS theRight,
      UNITS theBottom);
CRectRWC(const CPoint& theTopLeftCorner,
      const CPoint& theBottomRightCorner);
CRectRWC(const CRect &theRect);
CRectRWC & operator = (const CRectRWC &theRect);
virtual ~CRectRWC();
virtual RWspace binaryStoreSize() const;
virtual int compareTo(const RWCollectable *c) const;
virtual unsigned hash() const;
virtual RWBoolean isEqual(const RWCollectable* c) const;
```

```
virtual void restoreGuts(RWvistream& theStream);
virtual void restoreGuts(RWFile& theFile);
virtual void saveGuts(RWvostream& theStream) const;
virtual void saveGuts(RWFile& theFile) const;
```

Class: CRectangle

Superclass: CShape

```
CRectangle(CSubview* theEnclosure,
    const CRect& theRegion);
CRectangle(const CRectangle& theRectangle);
CRectangle& operator = (const CRectangle& theRectangle);
virtual ~CRectangle(void);
BOOLEAN IRectangle(BOOLEAN hasRoundCorners = FALSE,
    UNITS theCornerWidth = 0,
    UNITS theCornerHeight = 0,
    BOOLEAN isVisible = TRUE,
    GLUETYPE theGlue = NULLSTICKY);
virtual void Draw(const CRect& theClippingRegion);
virtual void SetRoundedCorners(BOOLEAN isCornerRounded,
    UNITS theCornerWidth = SAME,
    UNITS theCornerHeight = SAME);
```

Class: CRegion

Superclass: None

```
CRegion();
CRegion(const CRegion& theRegion);
CRegion(const CBounds& theBounds);
const CRegion& operator = (const CRegion& theRegion);
virtual ~CRegion();
virtual void Add(const CBounds& theBounds);
virtual void Add(const CCell& theCell);
virtual void AddRow(long theRow);
virtual void AddColumn(long theColumn);
virtual void Clear();
virtual void Remove(const CBounds& theBounds);
virtual void Remove(const CCell& theCell);
virtual void RemoveRow(long theRow);
virtual void RemoveColumn(long theColumn);
virtual void InsertColumns(long theColumn,
    long theCount = 1);
virtual void InsertRows(long theRow,
    long theCount = 1);
virtual void DeleteColumns(long theColumn,
    long theCount = 1);
virtual void DeleteRows(long theRow,
    long theCount = 1);
virtual BOOLEAN Contains(const CCell& theCell) const;
virtual BOOLEAN ContainsRow(long theRow) const;
virtual BOOLEAN ContainsColumn(long theColumn) const;
virtual void Union(const CRegion& theRegion);
virtual void Subtract(const CRegion& theRegion);
virtual void Xor(const CRegion& theRegion);
virtual const CBounds& Bounds() const;
```

```
void SetBounds();
CRegionIterator(const CRegion& theRegion);
BOOLEAN Next(CBounds* theBounds);
CRegionCellIterator(const CRegion& theRegion);
BOOLEAN Next(CCell* theCell);
CRegionRowIterator(const CRegion& theRegion);
BOOLEAN Next(long* theRow);
CRegionColumnIterator(const CRegion& theRegion);
BOOLEAN Next(long* theColumn);
```

Class: CRegularPoly

Superclass: CPolygon

```
CRegularPoly(CSubview* theEnclosure,
        const CPoint& theCenterPoint,
        UNITS theRadius,
        int theNumberOfSides,
        float theStartingAngle = 0.0);
CRegularPoly(const CRegularPoly& thePolygon);
CRegularPoly& operator = (const CRegularPoly& thePolygon);
virtual ~CRegularPoly(void);
BOOLEAN IRegularPoly(int theNumberOfSides,
        BOOLEAN isInteriorDrawn = FALSE,
        BOOLEAN isVisible = TRUE,
        long theGlue = NULLSTICKY);
virtual void Size(const CRect& theNewSize);
virtual void Rotate(float theNumOfDegrees);
virtual float GetStartingAngle(void);
virtual void SetNumberOfSides(int theNumberOfSides);
virtual int GetNumberOfSides(void) const;
virtual void SetDragging(BOOLEAN isDraggable);
virtual void SetSizing(BOOLEAN isSizable);
void CalculatePoints(void);
```

Class: CResource

Superclass: CNotifier

```
CResource(PwrResourceType theType,
        long theId,
        long theEndId = 0L);
virtual ~CResource(void);
virtual BOOLEAN Hold(void) = 0;
virtual void Release(void) = 0;
virtual long GetId(void) const;
virtual void SetId(long theID);
virtual long GetEndId(void) const;
virtual void SetEndId(long theId);
virtual PwrResourceType GetType(void) const;
virtual void SetType(PwrResourceType theType);
virtual BOOLEAN IObject(long theId,
        CObjectRWC *theObject) = 0;
virtual long FirstItem(void) = 0;
virtual long NextItem(void) = 0;
virtual long GetItemType(long theId) = 0;
CResource(const CResource &theResource);
CResource & operator = (const CResource &theResource);
```

Class: CResourceFactory

Superclass: None

virtual NButton* **ConstructNButton**(CSubview*,
long theContainerId,
long theId);
virtual CRadioGroup* **ConstructCRadioGroup**(CSubview*,
long theContainerId,
long theId);
virtual NCheckBox* **ConstructNCheckBox**(CSubview*,
long theContainerId,
long theId);
virtual NScrollBar* **ConstructNScrollBar**(CSubview*,
long theContainerId,
long theId);
virtual NEditControl* **ConstructNEditControl**(CSubview*,
long theContainerId,
long theId);
virtual NText* **ConstructNText**(CSubview*,
long theContainerId,
long theId);
virtual NListBox* **ConstructNListBox**(CSubview*,
long theContainerId,
long theId);
virtual NIcon* **ConstructNIcon**(CSubview*,
long theContainerId,
long theId);
virtual NListButton* **ConstructNListButton**(CSubview*,
long theContainerId,
long theId);
virtual NListEdit* **ConstructNListEdit**(CSubview*,
long theContainerId,
long theId);
virtual NGroupBox* **ConstructNGroupBox**(CSubview*,
long theContainerId,
long theId);
virtual NScrollText* **ConstructNScrollText**(CSubview*,
long theContainerId,
long theId);
virtual NRadioButton* **ConstructNRadioButton**(CRadioGroup*,
long theContainerId,
long theId);
virtual CResourceWindow* **ConstructCResourceWindow**(long theId);
virtual CResourceMenu* **ConstructCResourceMenu**(long theId);
virtual ~**CResourceFactory**();
CResourceFactory();

Class: CResourceFactoryDefault

Superclass: CResourceFactory

CResourceFactoryDefault();
virtual NButton* **ConstructNButton**(CSubview*,
long theContainerId,
long theId);
virtual CRadioGroup* **ConstructCRadioGroup**(CSubview*,
long theContainerId,
long theId);
virtual NCheckBox* **ConstructNCheckBox**(CSubview*,
long theContainerId,
long theId);
virtual NScrollBar* **ConstructNScrollBar**(CSubview*,
long theContainerId,
long theId);
virtual NEditControl* **ConstructNEditControl**(CSubview*,
long theContainerId,
long theId);
virtual NText* **ConstructNText**(CSubview*,
long theContainerId,
long theId);
virtual NListBox* **ConstructNListBox**(CSubview*,
long theContainerId,
long theId);
virtual NIcon* **ConstructNIcon**(CSubview*,
long theContainerId,
long theId);
virtual NListButton* **ConstructNListButton**(CSubview*,
long theContainerId,
long theId);
virtual NListEdit* **ConstructNListEdit**(CSubview*,
long theContainerId,
long theId);
virtual NGroupBox* **ConstructNGroupBox**(CSubview*,
long theContainerId,
long theId);
virtual NScrollText* **ConstructNScrollText**(CSubview*,
long theContainerId,
long theId);
virtual NRadioButton* **ConstructNRadioButton**(CRadioGroup*,
long theContainerId,
long theId);
virtual CResourceWindow* **ConstructCResourceWindow**(long theId);
virtual CResourceMenu* **ConstructCResourceMenu**(long theId);

Class: CResourceItems

Superclass: CNotifier

CResourceItems(long theId);
virtual ~**CResourceItems**(void);
CResourceItems(const CResourceItems &theResource);
CResourceItems & **operator =** (const CResourceItems &theResource);
virtual long **First**(void);
virtual long **Next**(void);
virtual long **GetType**(long theId);
virtual BOOLEAN **IObject**(long theId,
CObjectRWC *theObject);

Class: CResourceMenu

Superclass: CResource

CResourceMenu(long theId);
virtual **~CResourceMenu**(void);
virtual BOOLEAN **Hold**(void);
virtual void **Release**(void);
virtual BOOLEAN **IObject**(long theId,
    CObjectRWC *theObject);
virtual long **FirstItem**(void);
virtual long **NextItem**(void);
virtual long **GetItemType**(long theId);
CResourceMenu(const CResourceMenu &theResource);
CResourceMenu & **operator =** (const CResourceMenu &theResource);
void **BuildSubmenu**(MENU_ITEM *theXVTMenu,
    CSubmenu *theSubMenu,
    CMenu *theMenu);
void **InitMenu**(MENU_ITEM* theXVTMenu,
    CMenu *theMenu);

Class: CResourceMgr

Superclass: CNotifier

CResourceMgr(void);
virtual **~CResourceMgr**(void);
long **Insert**(CResource* theResource);
BOOLEAN **Remove**(long theId,
    BOOLEAN IsDelete = FALSE);
CResource* **Find**(long theId);
const RWOrdered* **GetResources**(void) const;
CResourceMgr(const CResourceMgr& theResourceMgr);
CResourceMgr& **operator =** (const CResourceMgr& theResourceMgr);

Class: CResourceWindow

Superclass: CResource

CResourceWindow(long theId);
virtual **~CResourceWindow**(void);
virtual BOOLEAN **Hold**(void);
virtual void **Release**(void);
virtual BOOLEAN **IObject**(long theId,
    CObjectRWC *theObject);
virtual long **FirstItem**(void);
virtual long **NextItem**(void);
virtual long **GetItemType**(long theId);
void **ConvertWinDefUnits**(void);
WIN_DEF* **FindWinDefEntry**(long theId);
CResourceWindow(const CResourceWindow &theResource);
CResourceWindow & **operator =** (const CResourceWindow &theResource);

Class: CRevOrdIteratorRW

Superclass: RWIterator

CRevOrdIteratorRW(const RWOrdered &theOrdered);
virtual **~CRevOrdIteratorRW**();
virtual RWCollectable * **findPrev**(const RWCollectable *theItem);
virtual RWCollectable * **findNext**(const RWCollectable *theItem);
virtual RWCollectable * **key**() const;
virtual RWCollectable * **operator**()();
virtual void **reset**() { itsIndex = RW_NPOS; }
CRevOrdIteratorRW(const CRevOrdIteratorRW &theIterator);
CRevOrdIteratorRW & **operator** = (const CRevOrdIteratorRW &theIterator);

Class: CScroller

Superclass: CVirtualFrame

CScroller(CSubview* theEnclosure,
const CRect& theRegion,
UNITS theVirtualWidth = 0,
UNITS theVirtualHeight = 0);
CScroller(CWindow* theScrollableWindow,
UNITS theVirtualWidth = 0,
UNITS theVirtualHeight = 0);
CScroller(const CScroller& theScroller);
CScroller& **operator** = (const CScroller& theScroller);
virtual **~CScroller**(void);
BOOLEAN **IScroller**(BOOLEAN hasHorizontalScrollBar = TRUE,
BOOLEAN hasVerticalScrollBar = TRUE,
BOOLEAN isThumbTracking = FALSE,
UNITS theLineIncrement = 10,
UNITS thePageIncrement = 50,
BOOLEAN isVisible = TRUE,
long theGlue = NULLSTICKY);
virtual void **SetHIncrements**(UNITS theLineIncrement,
UNITS thePageIncrement);
virtual void **SetVIncrements**(UNITS theLineIncrement,
UNITS thePageIncrement);
virtual UNITS **GetHLineIncrement**(void) const;
virtual UNITS **GetVLineIncrement**(void) const;
virtual UNITS **GetHPageIncrement**(void) const;
virtual UNITS **GetVPageIncrement**(void) const;
virtual void **SetThumbtracking**(BOOLEAN isThumbTracking);
virtual BOOLEAN **IsThumbtracking**(void) const;
static BOOLEAN **RegisterForUpdate**(CView* theView);
virtual void **VScroll**(SCROLL_CONTROL theEventType,
UNITS theThumbPosition);
virtual void **HScroll**(SCROLL_CONTROL theEventType,
UNITS theThumbPosition);
virtual void **ScrollViews**)const CPoint& theNewOrigin);
virtual CPoint **AutoScroll**(UNITS horizontalChange,
UNITS verticalChange);
virtual void **Key**(int theKey,
BOOLEAN shift,
BOOLEAN control);
virtual void **AddSubview**(const CView* theSubview);
virtual CPoint **GetGlobalOrigin**(void) const;

```
virtual CRect GetClippedFrame(void) const;
virtual void Size(const CRect& theRect);
virtual void Draw(const CRect& theClipping);
virtual void DoDraw(const CRect& theClippingRegion = MAXRect);
virtual void PrintDraw(const CRect& theClipping = MAXRect);
virtual void SetOrigin(const CPoint& theDeltaPoint);
virtual void Show();
virtual void Hide();
virtual void Enable();
virtual void Disable();
virtual void SetEnvironment(
        const CEnvironment& theEnvironment,
        BOOLEAN isUpdate);
virtual void SetEnclosure(CSubview* theEnclosure);
virtual void AdjustScrollBars(void);
CRect GetInitialFrame(CSubview* theEnclosure,
        const CRect& theRegion,
        BOOLEAN hasHorizontalScrollBar,
        BOOLEAN hasVerticalScrollBar) const;
CRect GetInnerFrame(void) const;
static RWOrdered & GetUpdateViewList(void);
static void ProcessViewUpdate(void);
```

Class: CShape

Superclass: CSubview

```
CShape(CSubview *theEnclosure,
        const CRect& theRegion);
CShape(const CShape& theShape);
CShape& operator = (const CShape& theShape);
virtual ~CShape(void);
BOOLEAN IShape(BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void Draw(const CRect& theClippingRegion);
```

Class: CSketchPad

Superclass: CSubview

```
CSketchPad(CSubview *theEnclosure,
        const CRect& theRegion);
CSketchPad(const CSketchPad& theSketchPad);
CSketchPad& operator = (const CSketchPad& theSketchPad);
virtual ~CSketchPad(void);
BOOLEAN ISketchPad(SKETCHTYPE theType = RECTANGLE,
        BOOLEAN isVisible = TRUE,
        long theGlue = NULLSTICKY);
virtual CRect GetSketchedRegion() const;
virtual CPoint GetStartPoint(void) const;
virtual CPoint GetEndPoint() const;
virtual void GetPoints(CPoint *&aPntAry,
        int &aNumPoints);
virtual void SetSketchType(SKETCHTYPE theType = RECTANGLE);
virtual SKETCHTYPE GetSketchType(void) const;
virtual void SetSketchEverywhere(BOOLEAN isSketchEverywhere);
virtual BOOLEAN IsSketchEverywhere(void);
```

```
virtual void MouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual CView* FindEventTarget(const CPoint& theLocation) const;
virtual void Draw(const CRect& theClippingRegion);
virtual void Enable(void);
virtual void Disable(void);
```

Class: CSparseArray

Superclass: None

```
CSparseArray(void);
CSparseArray(const CSparseArray& theSparseArray);
CSparseArray& operator = (const CSparseArray& theSparseArray);
virtual ~CSparseArray(void);
virtual void Insert(RWCollectable *theItem,
        int theRow,
        int theCol);
virtual BOOLEAN Remove(RWCollectable *theItem);
virtual RWCollectable* Remove(int theRow,
        int theCol);
virtual void* GetContents(int theRow,
        int theCol) const;
virtual int GetNumRows(void) const;
virtual int GetNumCols(void) const;
virtual int GetNumEntries(void) const;
virtual void Clear(void);
void ShiftColumns(int theColumn,
        int theShift);
```

Class: CSparseArrayIterator

Superclass: None

```
CSparseArrayIterator(const CSparseArray *aSparseArray);
~CSparseArrayIterator(void);
RWCollectable* Next(int *theRow = NULL,
        int *theCol = NULL);
void Reset(void);
CSparseArrayIterator(const CSparseArrayIterator& theIterator);
CSparseArrayIterator& operator = (const CSparseArrayIterator& theIterator);
```

Class: CSparseColIterator

Superclass: None

```
CSparseColIterator(const CSparseArray* theSparseArray,
        int theCol);
~CSparseColIterator(void);
RWCollectable* Next(int* theRow = NULL);
void Reset(void);
CSparseColIterator(const CSparseColIterator& theIterator);
CSparseColIterator& operator = (const CSparseColIterator& theIterator);
```

Class: CSparseRowIterator

Superclass: None

CSparseRowIterator(const CSparseArray* theSparseArray,
int theRow);
~CSparseRowIterator(void);
RWCollectable* **Next**(int* theCol = NULL);
void **Reset**(void);
CSparseRowIterator(const CSparseRowIterator& theIterator);
CSparseRowIterator& **operator** = (const CSparseRowIterator& theIterator);

Class: CSplitBar

Superclass: None

virtual UNITS **GetSplitBarSize**() const;
virtual CRect **GetSplitBarFrame**(UNITS theAxisLoc = PWR_SPLITBAR_DEFAULT) const = 0;
virtual BOOLEAN **HitTest**(const CPoint& theLoc) const;
virtual UNITS **GetAxis**() const;
virtual void **SetAxis**(UNITS theAxisLoc,
BOOLEAN theUpdate = TRUE);
virtual CRect **GetRange**() const;
virtual void **SetRange**(const CRect& theRange);
virtual void **Draw**() = 0;
virtual void **UpdateDraw**();
virtual BOOLEAN **ShouldDrawIntersection**(const CSplitBar* theSplitBar);
virtual BOOLEAN **IsTracking**() const;\
virtual void **BeginTracking**(const CPoint& theLoc);
virtual void **MoveTracking**(const CPoint& theLoc);
virtual UNITS **EndTracking**();
virtual UNITS **GetAxisFromPoint**(const CPoint& theLoc) = 0;
virtual CPoint **GetAxisEdge**(const CPoint& theLoc) = 0;
CHorizontalSplitBar(CWindow* theDrawWin,
const CRect& theRange,
UNITS theCurVertLoc,
UNITS theSplitBarSize = 7.0F,
COLOR theLightCol = COLOR_WHITE,
COLOR theShadowCol = COLOR_GRAY,
COLOR theBarCol = COLOR_LTGRAY);
virtual **~CHorizontalSplitBar**();
virtual CRect **GetSplitBarFrame**(UNITS theAxisLoc = PWR_SPLITBAR_DEFAULT) const;
virtual void **Draw**();
virtual void **DrawIntersection**(const CVerticalSplitBar* theSplitBar);
virtual UNITS **GetAxisFromPoint**(const CPoint& theLoc) = 0;
virtual CPoint **GetAxisEdge**(const CPoint& theLoc) = 0;
CVerticalSplitBar(CWindow* theDrawWin,
const CRect& theRange,
UNITS theCurVertLoc,
UNITS theSplitBarSize = 7.0F,
COLOR theLightCol = COLOR_WHITE,
COLOR theShadowCol = COLOR_GRAY,
COLOR theBarCol = COLOR_LTGRAY);
virtual **~CVerticalSplitBar**();
virtual CRect **GetSplitBarFrame**(UNITS theAxisLoc = PWR_SPLITBAR_DEFAULT) const;
virtual void **Draw**();
virtual void **DrawIntersection**(const CVerticalSplitBar* theSplitBar);

```
virtual UNITS GetAxisFromPoint(const CPoint& theLoc) = 0;
virtual CPoint GetAxisEdge( const CPoint& theLoc ) = 0;
```

Class: CSplitter

Superclass: CSubview

```
CSplitter(CSubview* theEnclosure,
        const CRect& theRect,
        CURSOR theHorGrabCursor = CURSOR_ARROW,
        CURSOR theHorDragCursor = CURSOR_ARROW,
        CURSOR theVerGrabCursor = CURSOR_ARROW,
        CURSOR theVerDragCursor = CURSOR_ARROW,
        CURSOR theHVGrabCursor = CURSOR_ARROW,
        CURSOR theHVDragCursor = CURSOR_ARROW);
virtual ~CSplitter();
virtual void ISplitter();
virtual BOOLEAN IsSplitTracking() const;
virtual void TrackSplitBegin(const CPoint &theGlobalHit);
virtual void TrackSplitMove(const CPoint &theGlobalHit);
virtual void TrackSplitEnd();
const CPane* QueryPane(long thePaneId) const;
int GetNumberOfPanes() const;
virtual void DoDraw(const CRect &theRegion = MAXRect);
virtual void DoSize(const CRect &theRegion);
virtual void OnAdjustPanes(CHorizontalSplitBar* theHorSplit,
        UNITS theHorAxis,
        CVerticalSplitBar* theVerSplit,
        UNITS theVerAxis) = 0;
virtual void SetSplitCursor(CursorState theCursor);
virtual CHorizontalSplitBar* FindHorSplit(const CPoint &theGlobalHit);
virtual CVerticalSplitBar* FindVerSplit(const CPoint &theGlobalHit);
virtual void DoNormalizePanes(CHorizontalSplitBar* theHorSplit,
        UNITS theHorAxis,
        CVerticalSplitBar* theVerSplit,
        UNITS theVerAxis);
virtual void DoAdjustPanes(CHorizontalSplitBar* theHorSplit,
        UNITS theHorAxis,
        CVerticalSplitBar* theVerSplit,
        UNITS theVerAxis);
virtual void AdjustPane(CPane* thePane,
        CHorizontalSplitBar* theHorSplit,
        UNITS theHorAxis,
        CVerticalSplitBar* theVerSplit,
        UNITS theVerAxis);
virtual CHorizontalSplitBar* CreateHorSplitBar(const CRect& theRange,
        UNITS theAxisLoc);
virtual CVerticalSplitBar* CreateVerSplitBar(const CRect& theRange,
        UNITS theAxisLoc);
virtual CSplitterMouseAgent* ConstructMouseAgent();
virtual CHorizontalSplitBar* ConstructHorSplitBar(const CRect& theRange,
        UNITS theAxisLoc);
virtual CVerticalSplitBar* ConstructVerSplitBar(const CRect& theRange,
        UNITS theAxisLoc);
```

Class: CSplitter::CPane

Superclass: None

long **GetId**() const;
CRect **GetFrame**() const;
CView* **GetView**() const;
BOOLEAN **IsLeftBorder**() const;
BOOLEAN **IsRightBorder**() const;
BOOLEAN **IsTopBorder**() const;
BOOLEAN **IsBottomBorder**() const;
CPane();
virtual **~CPane**();
CRect **GetAdjustedFrame**() const;
BOOLEAN **IsAdjusted**() const;
void **ResetAdjusted**();
void **ApplyAdjustedFrame**();

Class: CSplitter::CSplitterMouseAgent

Superclass: CMouseHandler

CSplitterMouseAgent(CSplitter* theSplitter,
short theTrackButton = 0 /* left button */);
virtual **~CSplitterMouseAgent**();
virtual BOOLEAN **DoDown**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);
virtual BOOLEAN **DoUp**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);
virtual BOOLEAN **DoMove**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);
virtual BOOLEAN **DoDouble**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);
virtual BOOLEAN **UsesGlobalCoords**();

Class: CSquare

Superclass: CRectangle

CSquare(CSubview* theEnclosure,
const CPoint& theTopLeftPoint,
int theSideLength);
CSquare(const CSquare& theSquare);
CSquare& **operator** = (const CSquare& theSquare);
virtual **~CSquare**();
BOOLEAN **ISquare**(BOOLEAN hasRoundCorners = FALSE,
int theCornerWidth = 0,
int theCornerHeight = 0,
BOOLEAN isVisible = TRUE,
GLUETYPE theGlue = NULLSTICKY);
virtual void **Size**(const CRect& theNewSize);

Class: CStackable

Superclass: None

```
CStackable();
virtual ~CStackable();
BOOLEAN IsTop(void) const;
void MakeTop(void);
virtual void DoUpdate(void) = 0;
static RWGDlist(CStackable) & GetStack(void);
```

Class: CStatusBar

Superclass: CSubview

```
CStatusBar(CSubview *theEnclosure);
CStatusBar(CSubview *theEnclosure,
        const CRect &theFrame);
CStatusBar(const CStatusBar& theStatusBar);
CStatusBar& operator = (const CStatusBar& theStatusBar);
void IStatusBar(int theBorderWidth = SB_DefaultBorderWidth,
        int theFieldBorderWidth = SB_DefaultFieldBorderWidth,
        int theSeparatorWidth = SB_DefaultSeparatorWidth,
        int theHPadding = SB_DefaultHPadding,
        int theVPadding = SB_DefaultVPadding,
        COLOR theFlatColor = SB_DefaultFlatColor,
        COLOR theLightShadowColor = SB_DefaultLightShadowColor,
        COLOR theDarkShadowColor = SB_DefaultDarkShadowColor);
virtual ~CStatusBar();
virtual BOOLEAN AppendField(long theFieldId,
        const CStringRW &theInitialStatus,
        UNITS theWidth = SB_FillToWidth);
virtual BOOLEAN InsertField(unsigned theIndex,
        long theFieldId,
        const CStringRW &theInitialStatus,
        UNITS theWidth = SB_FillToWidth;
virtual void SetFieldStatus(long theFieldId,
        const CStringRW &theStatus,
        BOOLEAN isShown = TRUE);
virtual CStringRW GetFieldStatus(long theFieldId) const;
virtual BOOLEAN AppendField(long theFieldId,
        CView *theFieldView,
        // theFieldView is consumed
        UNITS theWidth = SB_FitToView,
        BOOLEAN isInset = TRUE);
virtual BOOLEAN InsertField(unsigned theIndex,
        long theFieldId,
        CView *theFieldView,
        // theFieldView is consumed
        UNITS theWidth = SB_FitToView,
        BOOLEAN isInset = TRUE);
virtual long AddField(CView *theFieldView,
        // theFieldView is consumed
        UNITS theWidth = SB_FitToView,
        BOOLEAN isInset = TRUE);
virtual CView *GetField(long theFieldId) const;
virtual void RemoveField(long theFieldId);
virtual void ShowField(long theFieldId);
virtual void HideField(long theFieldId);
```

```
virtual void ShowAllFields(void);
virtual void HideAllFields(void);
virtual void RemoveAllFields(BOOLEAN itDeletesSubviews = TRUE);
virtual void SizeField(long theFieldId,
        UNITS theSize = SB_FitToView);
virtual unsigned GetFieldIndex(long theFieldId) const;
virtual long GetFieldId(unsigned theIndex) const;
virtual unsigned GetNumFields(void) const;
virtual UNITS GetFieldWidth(long theFieldId) const;
virtual BOOLEAN IsFieldInset(long theFieldId) const;
virtual BOOLEAN IsFieldVisible(long theFieldId) const;
virtual void ArrangeSubviews(void);
void GetSpacingValues(int *theBorderWidth,
        int *theFieldBorderWidth,
        int *theSeparatorWidth,
        int *theHPadding,
        int *theVPadding) const;
BOOLEAN IsAutoSized(void) const;
void SetAutoSized(BOOLEAN isAutoSized);
virtual void RemoveSubview(const CView* theSubview);
virtual void DoShow(void);
virtual void DoSize(const CRect& theNewSize);
virtual void Draw(const CRect& theClippingRegion);
virtual void DoDraw(const CRect& theClippingRegion = MAXRect);
virtual void DoSetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void SetSizing(BOOLEAN isSizable):
virtual void SetDragging( BOOLEAN isDraggable );
void Copy(const CStatusBar& theStatBar);
void SetFieldIndices(void);
CStatusField *GetStatusField(long theId) const;
```

Class: CStatusBarAttachment

Superclass: CAttachment

```
CStatusBarAttachment(CAttachmentFrame *theAttachmentFrame,
        CStatusBar *theStatusBar);
CStatusBarAttachment(const CStatusBarAttachment &theAttachment);
CStatusBarAttachment& operator = (const CStatusBarAttachment &theAttachment);
~CStatusBarAttachment(void);
CStatusBar* GetStatusBar(void) const;
const RWSortedVector* GetAllFitSizes(void);
CRect GetBestSize(void);
void Attach(CAttachmentFrame *theFrame,
        CAttachmentFrame::AttachmentPoint theAttachmentPoint);
void Detach(BOOLEAN hasTitleBar = TRUE);
void Popup(CWindow *theWindow,
        BOOLEAN hasTitleBar = FALSE);
BOOLEAN IsDraggable(void) const;
```

Class: CStringCollection

Superclass: None

CStringCollection(int theFirstId,
int theLastId);
CStringCollection(const RWOrdered &theCollection);
CStringCollection(SLIST theSlist);
CStringCollection(const CStringCollection &theCollection);
CStringCollection & **operator** = (const CStringCollection &theCollection);
~CStringCollection();
operator SLIST() const;
void **clearAndDestroy**();
RWBag **asBag**(void) const;
RWSet **asSet**(void) const;
RWOrdered **asOrderedCollection**(void) const;
RWBinaryTree **asSortedCollection**(void) const;

Class: CStringCollectionRWC

Superclass: CStringCollection, CNotifier

CStringCollectionRWC();
CStringCollectionRWC(int theFirstID,
int theLastID);
CStringCollectionRWC(const RWOrdered &theCollection);
CStringCollectionRWC(SLIST theSLIST);
CStringCollectionRWC(const CStringCollection &theStringCollection);
CStringCollection & **operator =** (const CStringCollectionRWC &theStringCollectionRWC);
virtual **~CStringCollectionRWC**();
virtual RWspace **binaryStoreSize**() const;
virtual int **compareTo**(const RWCollectable *c) const;
virtual unsigned **hash**() const;
virtual RWBoolean **isEqual**(const RWCollectable* c) const;
virtual void **restoreGuts**(RWvistream& theStream);
virtual void **restoreGuts**(RWFile& theFile);
virtual void **saveGuts**(RWvostream& theStream) const;
virtual void **saveGuts**(RWFile& theFile) const;

Class: CStringRW

Superclass: RWCString

CStringRW();
CStringRW(const char *theString);
CStringRW(const char *theString,
size_t theSize);
CStringRW(RWSize_T theCapacity);
CStringRW(const RWCString& theString);
CStringRW(const RWCSubString& theString);
CStringRW(int theRID);
~CStringRW();
int **collate**(const char* theString) const;
int **collate**(const CStringRW& theString) const;
int **compareTo**(const char *theString,
caseCompare cmp = exact) const;

```
int compareTo(const CStringRW& theString,
        caseCompare cmp = exact) const;
RWBoolean contains(const char* pat,
        caseCompare cmp = exact) const;
RWBoolean contains(const CStringRW& pat,
        caseCompare cmp = exact) const;
size_t first(char theChar) const;
size_t first(const char *theString) const;
size_t first(const CStringRW& theString) const;
size_t index(const char *pat,
        size_t i = 0,
        caseCompare cmp = exact) const;
size_t index(const CStringRW& theString,
        size_t i = 0,
        caseCompare cmp = exact) const;
size_t index(const char *pat,
        size_t patlen,
        size_t i,
        caseCompare cmp) const;
size_t index(const CStringRW& theString,
        size_t patlen,
        size_t i,
        caseCompare cmp) const;
size_t index(const RWCRegexp& pat,
        size_t i = 0) const;
size_t index(const RWCRegexp& pat,
        size_t *ext,
        size_t i = 0) const;
size_t last(char theChar) const;
size_t last(const char *theString) const;
size_t last(const CStringRW& theString) const;
size_t mbLength() const;
        // multibyte length, or
        // RW_NPOS on error
void toLower();
        // Change self to lower-case
void toUpper();
        // Change self to upper-case
int asWideChar(XVT_WCHAR& theWideChar,
        size_t theIndex = 0);
int asWideString(XVT_WCHAR *theWideCharString,
        size_t length);
static CStringRW fromWideChar(XVT_WCHAR theWideChar);
static CStringRW fromWideString(XVT_WCHAR *theWideCharString);
```

Class: CStringRWC

Superclass: CStringRWW, CNotifier

CStringRWC();
CStringRWC(const char *theString);
CStringRWC(const char *theString,
size_t theSize);
CStringRWC(RWSize_T theCapacity);
CStringRWC(const CStringRW& theString);
CStringRWC(const RWCSubString& theString);
CStringRWC(int theRID);
CStringRWC & **operator** = (const CStringRWC &theStringRWC);
virtual **~CStringRWC**();
virtual RWspace **binaryStoreSize**() const;
virtual int **compareTo**(const RWCollectable *c) const;
virtual unsigned **hash**() const;
virtual RWBoolean **isEqual**(const RWCollectable* c) const;
virtual void **restoreGuts**(RWvistream& theStream);
virtual void **restoreGuts**(RWFile& theFile);
virtual void **saveGuts**(RWvostream& theStream) const;
virtual void **saveGuts**(RWFile& theFile) const;

Class: CSubmenu

Superclass: CMenu

CSubmenu(void);
CSubmenu(const CStringRW &theText,
MENU_TAG theTag,
short theMnemonic,
long theFlags = ENABLED);
CSubmenu(const CSubmenu &theSubmenu);
CSubmenu &**operator** = (const CSubmenu &theSubmenu);
virtual **~CSubmenu**();
BOOLEAN **Append**(const CMenu &theMenu);
BOOLEAN **Insert**(MENU_TAG theInFrontOfTag,
const CMenu &theMenu);
BOOLEAN **Replace**(MENU_TAG theTag,
const CMenu &theMenu);
BOOLEAN **Remove**(MENU_TAG theTag);
CMenu* **Find**(MENU_TAG theTag);
const RWOrdered* **GetMenus**(void) const;
void **DestructiveHandle**(void);

Class: CSubview

Superclass: CView

CSubview(CSubview* theEnclosure,
const CRect& theRegion);
CSubview(const CSubview& theSubview);
CSubview& **operator** = (const CSubview& theSubview);
virtual **~CSubview**(void);
virtual void **DoShow**(void);
virtual void **DoHide**(void);
virtual void **DoActivate**(void);

```
virtual void DoDeactivate(void);
virtual void DoEnable(void);
virtual void DoDisable(void);
virtual void DoMouseDouble(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoDraw(const CRect& theClippingRegion = MAXRect);
virtual void DoPrintDraw(const CRect& theClippingRegion);
virtual void DoSetOrigin(const CPoint& theDeltaPoint);
virtual void SetEnvironment(const CEnvironment&
        theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetEnvironment(const CEnvironment&
        theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetGlue(GLUETYPE theGlue);
virtual void DoSetDragging(BOOLEAN isDraggable);
virtual void DoSetSizing(BOOLEAN isSizable);
virtual void SetSelectedView(CView* theSelectedView);
virtual CView* GetSelectedView(void) const;
virtual void AddSelectedView(CView* theView);
virtual void RemoveSelectedView(CView* theView);
virtual const RWOrdered* GetSelectedViews(void);
virtual void PlaceTopSubview(const CView* theView);
virtual void PlaceBottomSubview(const CView* theView);
virtual CView* FindSubview(const CPoint& theLocation) const;
virtual CView* FindSubview(int theViewId) const;
virtual CView* FindDeepSubview(const CPoint& theLocation) const;
virtual void FindSubviews(const CPoint& theLocation,
        RWOrdered* theList) const;
virtual const RWOrdered* GetSubviews(void) const;
int GetNumViews(void) const;
virtual const RWOrdered* GetSubObjects(void) const;
virtual CView* FindResourceView(long theContainerId,
        long theId) const;
virtual void SetKeyFocus(CView *theFocusedView);
CView* GetKeyFocus(void) const;
virtual void AddSubview(const CView* theSubview);
virtual void RemoveSubview(const CView* thesubview);
virtual void DoKey(const CKey&);
```

```
virtual void DoSize(const CRect& theNewSize);
virtual CPoint AutoScroll(UNITS theHorizontalChange,
        UNITS theVerticalChange);
CView* FindEventTarget(const CPoint& theLocation) const;
virtual void SetEnclosure(CSubview* theEnclosure);
BOOLEAN ISubview(BOOLEAN isVisible,
        GLUETYPE theGlue);
```

Class: CSwitchBoard

Superclass: CNotifier

```
static long EventHandler(WINDOW win,
        EVENT *ep);
static long DialogHandler(WINDOW win,
        EVENT *ep);
static BOOLEAN IsUpdateEvent(void);
CSwitchBoard(CApplication* theApplication);
CSwitchBoard(const CSwitchBoard& theSwitchBoard);
virtual ~CSwitchBoard(void);
```

Class: CTable

Superclass: CSubview

```
CTable(
        CSubview* theEnclosure,
        const CRect& theRegion,
        const CFont& theTableFont = STDFont,
        UNITS theDefaultRowHeight = kNoHeight,
        UNITS theDefaultColumnWidth = kNoWidth,
        long theChunkRows = 128,
        long theChunkColumns = 32);
virtual ~CTable();
BOOLEAN ITable(
        SelectionPolicy theSelectionPolicy = MultipleRow,
        BOOLEAN hasHorizontalScrollBar = TRUE,
        BOOLEAN hasVerticalScrollBar = TRUE,
        BOOLEAN hasColumnLabels = TRUE,
        BOOLEAN hasRowLabels = TRUE,
        BOOLEAN isThumbTracking = FALSE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void VScroll(SCROLL_CONTROL theEventType,
        UNITS thePos);
virtual void HScroll(SCROLL_CONTROL theEventType,
        UNITS thePos);
virtual void DoKey(const CKey& theKey);
virtual void DoCommand(long theCommand,
        void* theData = NULL);
virtual void SetTable(
        CTableAttributes& theAttributes,
        BOOLEAN thePropagate = FALSE );
virtual const CTableAttributes* GetTableAttributes() const;
virtual void SetCellOrigin(const CCell& theCell);
virtual CCell GetCellOrigin() const;
virtual void SetCellBounds(const CBounds& theBounds);
virtual CBounds GetCellBounds() const;
virtual CRect GetLocalInterior() const;
```

```
virtual void SetThumbTracking(BOOLEAN isThumbTracking);
virtual void MakeRowVisible(long theRow);
virtual void MakeColumnVisible(long theColumn);
virtual void MakeCellVisible(const CCell& theCell);
virtual void InsertColumns(long theColumn,
        long theCount = 1);
virtual void DeleteColumns(long theColumn,
        long theCount = 1);
virtual void InsertRows(long theRow,
        long theCount = 1);
virtual void DeleteRows(long theRow,
        long theCount = 1);
virtual void SetSelectionPolicy(
        SelectionPolicy theSelectionPolicy);
virtual SelectionPolicy GetSelectionPolicy() const;
virtual void SelectCells(
        const CBounds& theBounds,
        BOOLEAN isSelected = TRUE);
virtual void SelectRow(
        long theRow,
        BOOLEAN isSelected = TRUE);
virtual void SelectColumn(
        long theColumn,
        BOOLEAN isSelected = TRUE);
virtual void SelectCell(const CCell& theCell,
        BOOLEAN isSelected = TRUE);
virtual void SetSelectedRegion(
        const CRegion& theRegion);
virtual const CRegion& GetSelectedRegion() const;
virtual void DeselectAll();
virtual BOOLEAN IsRowSelected(long theRow) const;
virtual BOOLEAN IsColumnSelected(long theColumn) const;
virtual BOOLEAN IsCellSelected(
        const CCell& theCell) const;
virtual CRect ToScreen(const CBounds& theCells) const;
virtual CPoint ToScreen(const CCell& theCell) const;
virtual CBounds ToCell(CRect thePoints) const;
virtual CCell ToCell(CPoint thePoint) const;
virtual void SetDefaultRowHeight(UNITS theHeight);
virtual void SetDefaultColumnWidth(UNITS theWidth);
virtual UNITS GetDefaultRowHeight() const;
virtual UNITS GetDefaultColumnWidth() const;
virtual void SetRow(long theRow,
        UNITS theHeight);
virtual void SetRow(
        long theRow,
        CTableAttributes& theAttributes,
        BOOLEAN thePropagate = FALSE);
virtual void SetRow(long theRow,
        const CStringRW& theTitle);
virtual const CTableAttributes* GetRowAttributes(
        long theRow ) const;
virtual void SetColumn(long theColumn,
        UNITS theWidth);
```

```
virtual void SetColumn(
        long theColumn,
        CTableAttributes& theAttributes,
        BOOLEAN thePropagate = FALSE);
virtual void SetColumn(
        long theColumn,
        const CStringRW& theTitle);
virtual const CTableAttributes* GetColumnAttributes(
        long theColumn) const;
virtual void SetCell(
        const CCell& theCell,
        CTableAttributes& theAttributes);
virtual const CTableAttributes* GetCellAttributes(
        const CCell& theCell) const;
virtual void SetFocus(const CCell* theCell);
virtual const CCell* GetFocus() const;
virtual BOOLEAN ClassCanGetKeyFocus() const;
virtual void SetEventStyle(EventStyle theStyle);
virtual EventStyle GetEventStyle() const;
CTableView* GetRowLabels();
CTableView* GetColumnLabels();
CTableView* GetTableView();
const CTableView* GetRowLabels() const;
const CTableView* GetColumnLabels() const;
const CTableView* GetTableView() const;
void SetSource(CTableSource* theSource);
CTableSource* GetSource();
const CTableSource* GetSource() const;
virtual UNITS LabelWidth(const CFont& theFont);
virtual UNITS LabelHeight(const CFont& theFont);
virtual CMouseHandler* CreateMouseHandler();
virtual CRowLabels* CreateRowLabels(BOOLEAN hasOtherBar);
virtual CColumnLabels* CreateColumnLabels(BOOLEAN hasOtherBar);
virtual CTableView* CreateMainTable();
virtual NScrollBar* CreateScrollBar(
        const CRect& theRect,
        DIRECTION theDirection,
        BOOLEAN isVisible
CTable(const CTable& theTable) :
        CSubview(theTable) { }
CTable& operator=(const CTable& /* theTable */) ;
```

Class: CTableAttributes

Superclass: None

CTableAttributes();
CTableAttributes(const CTableAttributes& theAttributes);
~CTableAttributes();
CTableAttributes& **operator =** (const CTableAttributes& theAttributes);
int **operator ==** (const CTableAttributes& theAttributes) const;
void **Combine**(const CTableAttributes& theAttributes);
void **Font**(const CFont& theFont);
void **Foreground**(COLOR theColor);
void **Background**(COLOR theColor);
void **Justification**(TextJustification theJustification);
void **TopBorder**(Border theBorder);
void **LeftBorder**(Border theBorder);
void **BottomBorder**(Border theBorder);
void **RightBorder**(Border theBorder);
void **Interpreter**(CTableInterpreter* theInterpreter);
void **ReadOnly**(BOOLEAN isReadOnly);
void **Validator**(CValidator theValidator);
void **Mask**(unsigned int theMask);
void **SetBorderColor**(COLOR theColor,
BorderLocation theLocation = BorderAll);
void **SetBorderStyle**(BorderStyle theStyle,
BorderLocation theLocation = BorderAll);
void **UnFont**();
void **UnForeground**();
void **UnBackground**();
void **UnJustification**();
void **UnTopBorder**();
void **UnLeftBorder**();
void **UnBottomBorder**();
void **UnRightBorder**();
void **UnInterpreter**();
void **UnReadOnly**();
void **UnValidator**();
const CFont& **Font**() const;
COLOR **Foreground**() const;
COLOR **Background**() const;
TextJustification **Justification**() const;
Border **TopBorder**() const;
Border **LeftBorder**() const;
Border **BottomBorder**() const;
Border **RightBorder**() const;
CTableInterpreter* **Interpreter**() const;
BOOLEAN **ReadOnly**() const;
CValidator **Validator**() const;
unsigned int **Mask**() const;

Class: CTableCheckBoxInterpreter

Superclass: CTableViewInterpreter[xref]

CTableCheckBoxInterpreter(CTableView* theTable,
const CStringRW& theTitle);
void **OnString**(const CStringRW& theString);
const CStringRW& **OnString**() const;
void **OffString**(const CStringRW& theString);
const CStringRW& **OffString**() const;
virtual CCellView* **CreateCellView**(CTableView* theTable,
const CTableInterpreterParameters& theParameters,
const CTdiValue* theData);
virtual **~CTableCheckBoxInterpreter**() { }

Class: CTableEvent

Superclass: None

CTableEvent(CTable* theTable,
Type theType,
BOOLEAN theNotification = FALSE);
CTableEvent(CTable* theTable,
Type theType,
long theFirst,
long theCount,
BOOLEAN theNotification = FALSE);
CTableEvent(CTable* theTable,
Type theType,
long theRowColumn,
UNITS theSize,
BOOLEAN theNotification = FALSE);
CTableEvent(CTable* theTable,
Type theType,
const CCell* theCell,
BOOLEAN theNotification = FALSE);
CTableEvent(CTable* theTable,
Type theType,
const CCell* theCell,
const CKey* theKey,
BOOLEAN theNotification = FALSE);
CTableEvent(CTable* theTable,
Type theType,
const CRegion* theRegion,
BOOLEAN theSelect,
BOOLEAN theNotification = FALSE);
CTable* **GetTable**();
Type **GetType**() const;
BOOLEAN **IsPermissionEvent**() const;
void **Deny**();
BOOLEAN **IsDenied**() const;

Class: CTableListButtonInterpreter

Superclass: CTableViewInterpreter

CTableListButtonInterpreter(CTableView* theTable,
UNITS theHeight);
void **AddString**(const CStringRW& theString);
virtual CCellView* **CreateCellView**(CTableView* theTable,
const CTableInterpreterParameters& theParameters,
const CTdiValue* theData);
virtual **~CTableListButtonInterpreter**();

Class: CTableMouseHandler

Superclass: CMouseHandler

```
// The part of the table we're currently tracking
enum TablePart
{
        None,
        ColumnLabels,
        ColumnLabelsBorder,
        RowLabels,
        RowLabelsBorder,
        Table,
        FocusCell
};
CTableMouseHandler( CTable* theTable, CURSOR theSplitCursor = CURSOR_CROSS );
CTableMouseHandler(const CTableMouseHandler& theMouseHandler);
CTableMouseHandler& operator=(const CTableMouseHandler& theMouseHandler);
virtual ~CTableMouseHandler();
virtual BOOLEAN DoDown(
        CPoint&   theLocation,
        short&    theButton,
        BOOLEAN&  isShiftKey,
        BOOLEAN&  isControlKey);
virtual BOOLEAN DoUp(
        CPoint&   theLocation,
        short&    theButton,
        BOOLEAN&  isShiftKey,
        BOOLEAN&  isControlKey);
virtual BOOLEAN DoDouble(
        CPoint&   theLocation,
        short&    theButton,
        BOOLEAN&  isShiftKey,
        BOOLEAN&  isControlKey);
virtual BOOLEAN DoMove(
        CPoint&   theLocation,
        short&    theButton,
        BOOLEAN&  isShiftKey,
        BOOLEAN&  isControlKey);
virtual BOOLEAN UsesGlobalCoords();
```

Class: CTablePictureInterpreter

Superclass: CTableViewInterpreter

```
CTablePictureInterpreter(CTableView* theTable);
void AddImage(const CImage& theImage,
        const CStringRW& theKey);
void AddDefaultImage(const CImage& theImage);
void ResizeToFitCell(BOOLEAN theResizeToFit);
BOOLEAN ResizeToFitCell() const;
virtual CCellView* CreateCellView(CTableView* theTable,
        const CTableInterpreterParameters& theParameters,
        const CTdiValue* theData);
virtual ~CTablePictureInterpreter();
XVT_PIXMAP FindImage(const CStringRW& theKey) const;
```

```
virtual XVT_PIXMAP GetPixmapFromImage(const CWindow& theWindow,
        const CImage &theImage);
```

Class: CTableTdiSource

Superclass: CTableSource[xref], CNotifier

```
CTableTdiSource(CTable* theTable);
CTableTdiSource(const CTableTdiSource& theTableSource);
CTableTdiSource& operator = (const CTableTdiSource& theTableSource);
virtual ~CTableTdiSource();
virtual void SetCacheSize( int theRowsInCache);
virtual void Prime(const CBounds& theCell);
virtual const CTdiValue* GetTableData(const CCell& theCell);
virtual BOOLEAN PutTableData(const CCell& theCell,
        const CTdiValue* theData);
virtual void DoUpdateModel(long theControllerId,
        long theCommand,
        const CModel* theModel);
void RequestNextRow();
void RequestPreviousRow();
void RequestRow(long theRow);
void RowToCache(const CTdiValue* theRowValue,
        long theRow);
CTdiValue* CacheToRow(long theRow) const;
void DeleteOldestRow();
```

Class: CTableTextInterpreter

Superclass: CTableInterpreter

```
CTableTextInterpreter(CTableView* theTable,
        unsigned int theAttributes = TX_BORDER,
        UNITS theRightMargin = 1000,
        int theCharacterLimit = 1000,
        long theCommand NULLcmd,
        BOOLEAN singleLin = TRUE,
        BOOLEAN notifyAllChanges = FALSE);
unsigned int GetAttributes() const
        { return itsAttributes; }
UNITS GetRightMargin() const
        { return itsRightMargin; }
int GetCharacterLimit() const
        { return itsCharacterLimit; }
long GetCommand() const
        { return itsCommand; }
void SetAttributes(unsigned int theAttributes)
        { itsAttributes = theAttributes; }
void SetRightMargin(UNITS theRightMargin)
        { itsRightMargin = theRightMargin; }
void SetCharacterLimit(int theCharacterLimit)
        { itsCharacterLimit = theCharacterLimit; }
void SetCommand(long theCommand)
        { itsCommand = theCommand; }
virtual void Draw(const CTableInterpreterParameters& theParameters);
virtual BOOLEAN CanFocus(const CTableInterpreterParameters& theParameters);
virtual void FocusIn(const CTableInterpreterParameters& theParameters);
virtual void FocusOut(const CTableInterpreterParameters& theParameters);
```

```
virtual BOOLEAN Key(const CTableInterpreterParameters& theParameters,
        const CKey& theKey);
virtual void Scroll(const CPoint& theOffset);
virtual ~CTableTextInterpreter() { }
UNITS TextHeight(const CFont& theFont,
        const CStringRW& theString);
```

Class: CTableView

Superclass: CSubview

```
CTableView(
        CSubview* theEnclosure,
        const CRect& theRegion,
        const CFont& theTableFont = STDFont,
        UNITS theDefaultRowHeight = kNoHeight,
        UNITS theDefaultColumnWidth = kNoWidth,
        long theChunkRows = 128,
        long theChunkColumns = 32);
virtual ~CTableView();
virtual BOOLEAN ITableView(
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY,
        const CFont& theFont = STDFont);
virtual void Draw(const CRect& theClip);
virtual void Size(const CRect& theNewSize);
virtual void VScroll(SCROLL_CONTROL theEventType,
        UNITS thePos);
virtual void HScroll(SCROLL_CONTROL theEventType,
        UNITS thePos);
virtual void MouseDouble(
        CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseDown(
        CPoint theLocation,
        short theButton = 0,
         BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void SetTable(
        CTableAttributes& theAttributes,
        BOOLEAN thePropagate= FALSE);
virtual void AssociateScrollBars(
        NScrollBar* theVScrollBar,
        NScrollBar* theHScrollBar);
virtual void SetCellOrigin(const CCell& theCell);
virtual CCell GetCellOrigin() const;
virtual void SetPixelOrigin(const CPoint& thePoint);
virtual CPoint GetPixelOrigin() const;
virtual void SetCellBounds(const CBounds& theBounds);
virtual CBounds GetCellBounds() const;
virtual void SetCellClipping(BOOLEAN theClip);
virtual BOOLEAN GetCellClipping();
virtual void SetRow(long theRow,
        UNITS theHeight);
```

```
virtual void SetRow(
        long theRow,
        CTableAttributes& theAttributes,
        BOOLEAN thePropagate = FALSE);
virtual void InsertRows(long theRow,
        long theCount = 1);
virtual void DeleteRows(long theRow,
        long theCount = 1);
virtual void SetColumn(long theColumn,
        UNITS theWidth);
virtual void SetColumn(
        long theColumn,
        CTableAttributes& theAttributes,
        BOOLEAN thePropagate = FALSE);
virtual void InsertColumns(long theColumn,
        long theCount = 1);
virtual void DeleteColumns(long theColumn,
        long theCount = 1);
virtual void SetCell(
        const CCell& theCell,
        CTableAttributes& theAttributes);
virtual void SetFocus(const CCell* theCell);
virtual const CCell* GetFocus() const;
virtual void InvalidateRow(long theRow);
virtual void InvalidateColumn(long theColumn);
virtual void InvalidateCell(const CCell& theCell);
virtual void InvalidateBounds(const CBounds& theBounds);
virtual void InvalidateRegion(const CRegion& theRegion);
virtual void InvalidateSelection();
virtual void CellToParameters(
        const CCell& theCell,
        CTableInterpreterParameters* theParameters);
virtual void CellToAttributes(
        const CCell& theCell,
        CTableAttributes* theAttributes);

const CTableAttributeArray& GetAttributes() const;
const CTableCoordinates GetCoordinates() const;
CTableAttributeArray& GetAttributes();
CTableCoordinates& GetCoordinates();
virtual void SetSource(CTableSource* theSource);
virtual CTableSource* GetSource();
virtual const CTableSource* GetSource() const;
virtual const CTdiValue* GetData(const CCell& theCell);
virtual BOOLEAN PutData(
        const CCell& theCell,
        const CTdiValue* theData);
BOOLEAN IsFlushRight() const;
BOOLEAN IsFlushBottom() const;
virtual void ChangeDrawingContext(
        CDrawingContext* theDrawingContext,
        CTableAttributes& theAttributes);
virtual void DrawColumnBackground(
        long theColumn,
        long theFirstRow,
        long theLastRow,
        CTableAttributes* theAttributes);
```

```
virtual void DrawRowBackground(
        long theRow,
        long theFirstColumn,
        long theLastColumn,
        CTableAttributes* theAttributes);
virtual void DrawColumn(
        const CRect& theClip,
        long theColumn,
        long theFirstRow,
        long theLastRow,
        CTableAttributes* theColumnAttributes,
        CTableAttributes* theRowAttributes);
virtual CCell OriginWithinBounds(
        const CCell& aProposedOrigin,
        CPoint* theOffset = NULL);
virtual void AdjustVScrollBar();
virtual void AdjustHScrollBar();
CTableView(const CTableView& theTable) :
        CSubview(theTable),
        itsCoordinates(NULL, STDFont);
CTableView& operator = (const CTableView& theTable) ;
```

Class: CTabStop

Superclass: None

CTabStop(CView* theView);
CTabStop(CNavigator* theSubNavigator);
virtual ~**CTabStop**();
CView* **GetView**() const;
void **SetView**(CView* theView);
CNavigator* **GetSubNavigator**() const;
void **SetSubNavigator**(CNavigator*);
void **AddHotKey**(const CKey&);
BOOLEAN **HasHotKey**(const CKey&) const;
BOOLEAN **GetTitleHotKey**(CKey*) const;
virtual BOOLEAN **IsActive**() const;
virtual BOOLEAN **CanActivate**() const;
virtual void **Activate**();
virtual void **Deactivate**();
virtual void **DoHit**();
virtual void **DoHit**(CONTROL_INFO);

Class: CTaskDoc

Superclass: CDocument

CTaskDoc(CApplication *theApplication,
long theId = PWRIdBase);
virtual void **BuildWindow**(void);
virtual **~CTaskDoc**();

Class: CTaskWin

Superclass: CWindow

virtual int **GetMenuBarId**(void) const;
virtual void **SizeWindow**(int theNewWidth,
int theNewHeight);
virtual BOOLEAN **Close**(void);
virtual void **Enable**(void);
virtual void **Disable**(void);
virtual void **Show**(void);
virtual void **Hide**(void);
virtual void **UpdateUnits**(CUnits* theUnits);
CTaskWin(CDocument *theDocument,
WINDOW theXVTWindow);
virtual **~CTaskWin**();

Class: CTdiBooleanValue

Superclass: CTdiValue

```
CTdiBooleanValue();
CTdiBooleanValue(BOOLEAN theState,
    const CNotifier* theOriginator = NULL,
    long theContext = NULLcmd);
CTdiBooleanValue(const CTdiBooleanValue& theModel);
const CTdiBooleanValue& operator = (const CTdiBooleanValue& theModel);
virtual ~CTdiBooleanValue();
BOOLEAN GetState() const;
void SetState(BOOLEAN theState);
virtual BOOLEAN Copy(const CTdiValue& thePrototype);
virtual BOOLEAN IsConvertible(CTdiValue::Type theType) const;
virtual CTdiValue::Type GetType() const;
virtual CStringRW ToString() const;
virtual int ToInteger() const;
virtual float ToFloat() const;
virtual BOOLEAN ToBoolean() const;
virtual BOOLEAN FromString(const CStringRW& theValue);
virtual BOOLEAN FromInteger(int theValue);
virtual BOOLEAN FromFloat(float theValue);
virtual BOOLEAN FromBoolean(BOOLEAN theValue);
```

Class: CTdiCommandTranslator

Superclass: CNotifier

```
CTdiCommandTranslator(CNotifier* theTarget);
CTdiCommandTranslator(const CTdiCommandTranslator& theTranslator);
const CTdiCommandTranslator& operator = (const CTdiCommandTranslator& theTranslator);
virtual ~CTdiCommandTranslator();
virtual void DoUpdateModel(long theControllerId,
    long theCommand,
    const CModel* theModel);
void AddTranslation(long theFromCommand,
    long theToCommand);
void AddListPopulationTranslations();
void AddListSelectionTranslations();
```

Class: CTdiConnection

Superclass: None

```
CTdiConnection(CNotifier* theProvider,
    CNotifier* theDependent,
    CTdiValue* thePrototype = NULL,
    // consumed by connection
    BOOLEAN autoConnect = TRUE);
virtual ~CTdiConnection();
virtual BOOLEAN Connect();
virtual BOOLEAN Disconnect();
virtual void Request(long theRequestCommand,
    CTdiValue* theData = NULL);
CNotifier* GetProvider() const;
CNotifier* GetDependent() const;
```

CTdiValue* **GetPrototype**() const;
void **SetPrototype**(CTdiValue* thePrototype);
void **SetContext**(long theContext);
long **GetContext**() const;
CTdiConnection(const CTdiConnection& theCopy);
CTdiConnection& **operator** = (const CTdiConnection& theCopy);

Class: CTdiController

Superclass: CController

CTdiController();
virtual **~CTdiController**();
CTdiController(const CTdiController& theController);
CTdiController& **operator** = (const CTdiController& theController);
virtual BOOLEAN **DoChange**(const CNotifier* theProvider,
    long theCommand,
    const CTdiValue* theModel);
virtual BOOLEAN **DoChange**(const CNotifier* theProvider,
    long theCommand,
    const CModel* theModel);
virtual BOOLEAN **AssociateModel**(CNotifier* theDependent,
    CTdiValue* thePrototype);
virtual BOOLEAN **AddDependent**(CNotifier* theDependent,
    CTdiValue* thePrototype);
virtual BOOLEAN **AddDependent**(CNotifier* theDependent);
virtual BOOLEAN **RemoveDependent**(CNotifier* theDependent,
    CTdiValue*& thePrototype);
virtual BOOLEAN **RemoveDependent**(CNotifier* theDependent);
virtual BOOLEAN **AddProvider**(CNotifier* theProvider);
virtual const RWOrdered* **GetDependents**() const;
virtual BOOLEAN **IsDependent**(const CNotifier* theDependent) const;
virtual void **Suspend**();
virtual void **Resume**();
void **SetActiveDependent**(CNotifier* theDependent);
CNotifier* **GetActiveDependent**() const;
BOOLEAN **IsOriginator**(const CTdiValue* theModel,
    const CNotifier* theNotifier);

Class: CTdiDateValue

Superclass: CTdiValue, RWDate

CTdiDateValue();
CTdiDateValue(const RWDate& theDate,
    const CNotifier* theOriginator = NULL,
    long theContext = NULLcmd);
CTdiDateValue(const CTdiDateValue& theModel);
const CTdiDateValue& **operator** = (const CTdiDateValue& theModel);
virtual **~CTdiDateValue**();
virtual BOOLEAN **Copy**(const CTdiValue& thePrototype);
virtual BOOLEAN **IsConvertible**(CTdiValue::Type theType) const;
virtual CTdiValue::Type **GetType**() const;
virtual CStringRW **ToString**() const;
virtual BOOLEAN **FromString**(const CStringRW& theValue);

Class: CTdiFloatValue

Superclass: CTdiValue, CFloat

CTdiFloatValue();
CTdiFloatValue(float theValue,
const CNotifier* theOriginator = NULL,
long theContext = NULLcmd);
CTdiFloatValue(const CTdiFloatValue& theModel);
const CTdiFloatValue& **operator** = (const CTdiFloatValue& theModel);
virtual **~CTdiFloatValue**();
virtual BOOLEAN **Copy**(const CTdiValue& thePrototype);
virtual BOOLEAN **IsConvertible**(CTdiValue::Type theType) const;
virtual CTdiValue::Type **GetType**() const;
virtual CStringRW **ToString**() const;
virtual int **ToInteger**() const;
virtual float **ToFloat**() const;
virtual BOOLEAN **ToBoolean**() const;
virtual BOOLEAN **FromString**(const CStringRW& theValue);
virtual BOOLEAN **FromInteger**(int theValue);
virtual BOOLEAN **FromFloat**(float theValue);
virtual BOOLEAN **FromBoolean**(BOOLEAN theValue);

Class: CTdiIndexConnection

Superclass: CTdiConnection

CTdiIndexConnection(CNotifier* theProvider,
CNotifier* theDependent,
long theContext = NULLcmd,
BOOLEAN autoConnect = TRUE,
CTdiTableFactory* theFactory = NULL);
CTdiIndexConnection(const CTdiIndexConnection& theCopy);
CTdiIndexConnection& **operator** = (const CTdiIndexConnection& theCopy);

Class: CTdiIntegerValue

Superclass: CTdiValue, RWInteger

CTdiIntegerValue();
CTdiIntegerValue(int theValue,
const CNotifier* theOriginator = NULL,
long theContext = NULLcmd);
CTdiIntegerValue(const CTdiIntegerValue& theModel);
const CTdiIntegerValue& **operator** = (const CTdiIntegerValue& theModel);
virtual **~CTdiIntegerValue**();
virtual BOOLEAN **Copy**(const CTdiValue& thePrototype);
virtual BOOLEAN **IsConvertible**(CTdiValue::Type theType) const;
virtual CTdiValue::Type **GetType**() const;
virtual CStringRW **ToString**() const;
virtual int **ToInteger**() const;
virtual float **ToFloat**() const;
virtual BOOLEAN **ToBoolean**() const;
virtual BOOLEAN **FromString**(const CStringRW& theValue);
virtual BOOLEAN **FromInteger**(int theValue);
virtual BOOLEAN **FromFloat**(float theValue);
virtual BOOLEAN **FromBoolean**(BOOLEAN theValue);

Class: CTdiListValue

Superclass: CTdiValue

```
CTdiListValue(const CNotifier* theOriginator,
        long theContext = NULLcmd);
CTdiListValue(const CTdiListValue& theCopy);
const CTdiListValue& operator = (const CTdiListValue& theModel);
virtual ~CTdiListValue();
virtual BOOLEAN Copy(const CTdiValue& thePrototype);
virtual BOOLEAN IsConvertible(CTdiValue::Type theType) const;
virtual CTdiValue::Type GetType() const;
virtual CStringRW ToString() const;
virtual int ToInteger() const;
virtual float ToFloat() const;
virtual BOOLEAN ToBoolean() const;
virtual BOOLEAN FromString(const CStringRW& theValue);
virtual BOOLEAN FromInteger(int theValue);
virtual BOOLEAN FromFloat(float theValue);
virtual BOOLEAN FromBoolean(BOOLEAN theValue);
virtual BOOLEAN Next() const;
virtual BOOLEAN GoTo(size_t theIndex);
virtual size_t Entries() const;
virtual BOOLEAN Clear();
virtual BOOLEAN Append(CTdiValue* theValue);
virtual BOOLEAN Set(size_t theIndex,
        CTdiValue* theValue);
```

Class: CTdiStringValue

Superclass: CTdiValue, CStringRW

```
CTdiStringValue();
CTdiStringValue(const CStringRW& theString,
        const CNotifier* theOriginator = NULL,
        long theContext = NULLcmd);
CTdiStringValue(const CTdiStringValue& theModel);
const CTdiStringValue& operator = (const CTdiStringValue& theModel);
virtual ~CTdiStringValue();
virtual BOOLEAN Copy(const CTdiValue& thePrototype);
virtual BOOLEAN IsConvertible(CTdiValue::Type theType) const;
virtual CTdiValue::Type GetType() const;
virtual CStringRW ToString() const;
virtual int ToInteger() const;
virtual float ToFloat() const;
virtual BOOLEAN ToBoolean() const;
virtual BOOLEAN FromString(const CStringRW& theValue);
virtual BOOLEAN FromInteger(int theValue);
virtual BOOLEAN FromFloat(float theValue);
virtual BOOLEAN FromBoolean(BOOLEAN theValue);
```

Class: CTdiTableConnection

Superclass: CTdiConnection

CTdiTableConnection(CNotifier* theProvider,
CNotifier* theDependent,
int theRow = CURRENT_INDEX,
int theColumn = CURRENT_INDEX,
long theContext = NULLcmd,
BOOLEAN autoConnect = TRUE,
CTdiTableFactory* theFactory = NULL);
CTdiTableConnection(CNotifier* theProvider,
CNotifier* theDependent,
int theRow = CURRENT_INDEX,
CStringRW theColumn = NULLString,
long theContext = NULLcmd,
BOOLEAN autoConnect = TRUE,
CTdiTableFactory* theFactory = NULL);
virtual **~CTdiTableConnection**();
int **GetRowIndex**() const;
int **GetColumnIndex**() const;
CStringRW **GetColumnName**() const;
void **SetRowIndex**(int theIndex);
void **SetColumnIndex**(int theIndex);
void **SetColumnName**(CStringRW theName);
void **UpdatePrototype**();
CTdiTableConnection(const CTdiTableConnection& theCopy);
CTdiTableConnection& **operator** = (const CTdiTableConnection& theCopy);

Class: CTdiTableFactory

Superclass: None

virtual CTdiValue* **NewRowPrototype**(int theRow,
long theContext) = 0;
virtual CTdiValue* **NewColumnPrototype**(int theColumn,
long theContext) = 0;
virtual CTdiValue* **NewColumnPrototype**(CStringRW theColumn,
long theContext) = 0;
virtual CTdiValue* **NewFieldPrototype**(int theRow,
int theColumn,
long theContext) = 0;
virtual CTdiValue* **NewFieldPrototype**(int theRow,
CStringRW theColumnName,
long theContext) = 0;
virtual CTdiValue* **NewIndexPrototype**(long theContext) = 0;

Class: CTdiTimeValue

Superclass: CTdiValue, RWTime

CTdiTimeValue();
CTdiTimeValue(const RWTime& theValue,
const CNotifier* theOriginator = NULL,
long theContext = NULLcmd);
CTdiTimeValue(const CTdiTimeValue& theModel);
const CTdiTimeValue& **operator** = (const CTdiTimeValue& theModel);
virtual **~CTdiTimeValue**();
virtual BOOLEAN **Copy**(const CTdiValue& thePrototype);
virtual BOOLEAN **IsConvertible**(CTdiValue::Type theType) const;
virtual CTdiValue::Type **GetType**() const;
virtual CStringRW **ToString**() const;
virtual BOOLEAN **FromString**(const CStringRW& theValue);

Class: CTdiValue

Superclass: CModel

CTdiValue(const CNotifier* theOriginator = NULL,
long theContext = NULLcmd);
CTdiValue(const CTdiValue& copy);
const CTdiValue& **operator** = (const CTdiValue& copy);
virtual **~CTdiValue**();
long **GetContext**(void) const;
void **SetContext**(long theContext);
const CNotifier* **GetOriginator**() const;
virtual BOOLEAN **Copy**(const CTdiValue& prototype) = 0;
virtual BOOLEAN **IsConvertible**(CTdiValue::Type theType) const;
virtual CTdiValue::Type **GetType**() const;
virtual CStringRW **ToString**() const;
virtual int **ToInteger**() const;
virtual float **ToFloat**() const;
virtual BOOLEAN **ToBoolean**() const;
virtual BOOLEAN **FromString**(const CStringRW& theValue);
virtual BOOLEAN **FromInteger**(int theValue);
virtual BOOLEAN **FromFloat**(float theValue);
virtual BOOLEAN **FromBoolean**(BOOLEAN theValue);
virtual void **Reset**() const;
virtual BOOLEAN **Next**() const;
virtual BOOLEAN **GoTo**(size_t theIndex);
virtual size_t **Entries**() const;
virtual BOOLEAN **Clear**();
virtual BOOLEAN **Append**(CTdiValue* theValue);
virtual BOOLEAN **Set**(size_t theIndex,
CTdiValue* theValue);
virtual BOOLEAN **Change**(long theCommand,
const CModel* theModel);

Class: CText

Superclass: CView

```
CText(CSubview* theEnclosure,
        const CPoint& theTopLeft,
        const CStringRW& theText = NULLString);
CText(CSubview* theEnclosure,
        const CRect& theRegion,
        const CStringRW& theText = NULLString);
CText(const CText& theText);
CText& operator = (const CText& theText);
virtual ~CText(void);
BOOLEAN IText(const CStringRW& theText,
        BOOLEAN isOpaque = FALSE,
        Boolean isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void SetText(const CStringRW& theText);
virtual void SetTitle(const CStringRW& theText);
virtual const CStringRW GetText(void) const;
virtual BOOLEAN IsOpaque(void) const;
virtual void SetOpaque(BOOLEAN isOpaque);
virtual void Select(void);
virtual void Deselect(void);
virtual BOOLEAN IsSelected(void) const;
virtual UNITS GetHeight(void) const;
virtual UNITS GetWidth(void) const;
virtual void Draw(const CRect& theClippingRegion);
virtual void DoPrintDraw(const CRect& theClippingRegion);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void SetSizing(BOOLEAN isSizable);
virtual void Size(const CRect& theRect);
void SetPlacement(Placement thePlacement);
Placement GetPlacement(void) const;
void RecalculateSize(void);
```

Class: CToolBar

Superclass: CSubview

```
enum AutosizeLocation { AL_NONE = 0,
        AL_TOP,
        AL_BOTTOM,
        AL_LEFT,
        AL_RIGHT };
CToolBar(CSubview *theEnclosure,
        AutosizeLocation theLocation = AL_TOP);
CToolBar(CSubview *theEnclosure,
        const CRect &theFrame);
CToolBar(const CToolBar &theToolBar);
CToolBar& operator = (const CToolBar &theToolBar);
void IToolBar(int theBorderWidth = TB_DefaultBorderWidth,
        int theSeparatorWidth = TB_DefaultSeparatorWidth,
        COLOR theFlatColor = TB_DefaultFlatColor,
        COLOR theLightShadowColor = TB_DefaultLightShadowColor,
        COLOR theDarkShadowColor = TB_DefaultDarkShadowColor);
virtual ~CToolBar();
```

virtual void **InsertSeparator**(size_t theLocation);
void **AppendSeparator**(void);
virtual void **RemoveSeparator**(size_t theLocation);
virtual void **RemoveAllSeparators**(void);
virtual const RWBitVec& **GetSeparators**(void) const;
virtual void **ArrangeSubviews**(void);
void **GetSpacingValues**(int *theBorderWidth,
int *theSeparatorWidth) const;
BOOLEAN **IsAutoSized**(void) const;
void **SetAutoSized**(BOOLEAN isAutoSized,
AutosizeLocation theLocation = AL_TOP);
AutosizeLocation **GetAutoSizeLocation**(void) const;
void **UpdateMenuButtons**(void);
void **UpdateMenuButton**(MENU_TAG theMenuTag);
virtual void **AddSubview**(const CView* theSubview);
virtual void **RemoveSubview**(const CView* theSubview);
virtual void **DoSize**(const CRect& theNewSize);
virtual void **DoDraw**(const CRect& theClippingRegion = MAXRect);
virtual void **Draw**(const CRect& theClippingRegion);
virtual void **DoSetFont**(const CFont &theNewFont,
BOOLEAN isUpdate = FALSE);
virtual void **DoSetEnvironment**(const CEnvironment& theNewEnvironment,
BOOLEAN isUpdate = FALSE);
virtual void **SetSizing**(BOOLEAN isSizable):
virtual void **SetDragging**(BOOLEAN isDraggable);
void **Copy**(const CToolBar &theToolBar);
void **AutoSize**(void);
virtual void **InternalArrangeSubviews**(UNITS &theWidth,
UNITS &theHeight,
BOOLEAN isSizingSubviews,
UNITS &aMaxLeft);
UNITS **HLogicalBorderSpace**(void);
UNITS **VLogicalBorderSpace**(void);
void **InsertMenuButton**(MENU_TAG theMenuTag,
CMenuButton *theButton);
void **RemoveMenuButton**(const CMenuButton *theButton);
void **MoveView**(size_t theFromLocation,
size_t theToLocation);

Class: CToolBarAttachment

Superclass: CAttachment

CToolBarAttachment(CToolBar *theToolBar);
CToolBarAttachment(CToolBar *theToolBar,
const CRect &theBestSize);
CToolBarAttachment(const CToolBarAttachment &theAttachment);
CToolBarAttachment& **operator** = (const CToolBarAttachment &theAttachment);
virtual **~CToolBarAttachment**(void);
virtual void **InvalidateSizes**(void);
CToolBar* **GetToolBar**(void) const;
virtual const RWSortedVector* **GetAllFitSizes**(void);
virtual CRect **GetBestSize**(void);
virtual CRect **GetSize**(void);
virtual void **Detach**(BOOLEAN hasTitleBar = TRUE);

virtual void **Popup**(CWindow *theWindow,
BOOLEAN hasTitleBar = FALSE);
virtual void **RegisterSink**(CDragSink *theDragSink,
BOOLEAN isInFront = TRUE);
virtual void **UnregisterSink**(CDragSink *theDragSink);
virtual void **CalculateFitSizes**(void);
virtual void **CalculateBestSize**(void);
virtual void **InternalAttach**(CAttachmentFrame *theFrame,
CAttachmentFrame::AttachmentPoint theAttachmentPoint);
CRect **GetDesiredSize**(void);

Class: CToolPalette

Superclass: None

CToolPalette(const CStringRW& theTitle = NULLString);
CToolPalette(const CToolPalette &theToolPalette);
CToolPalette & **operator =** (const CToolPalette &theToolPalette);
~CToolPalette();
void **AppendTool**(CImage* theImage,
long theId,
BOOLEAN isDraggable = FALSE,
BOOLEAN isEnabled = TRUE);
void **AppendTool**(CImage* theImage,
long theId,
CToolPalette* theSubPalette,
BOOLEAN isEnabled = TRUE);
void **InsertTool**(size_t theLocation,
CImage* theImage,
long theId,
BOOLEAN isDraggable = FALSE,
BOOLEAN isEnabled = TRUE);
void **InsertTool**(size_t theLocation,
CImage* theImage,
long theId,
CToolPalette* theSubPalette,
BOOLEAN isEnabled = TRUE);
CToolPalette* **RemoveTool**(long theId);
CToolPalette* **GetParent**(void);
void **AppendSeparator**(void);
void **InsertSeparatorAfter**(long theId);
void **RemoveSeparatorAfter**(long theId);
void **RemoveAllSeparators**(BOOLEAN itIsRecursive = FALSE);
void **EnableTool**(long theId,
BOOLEAN isEnabled);
BOOLEAN **IsToolEnabled**(long theId) const;
void **SelectTool**(long theId);
long **GetSelectedTool**(void) const;
void **DeselectTools**(void);
void **DeselectAllTools**(void);
void **DeselectLocalTools**(void);
const CImage* **GetToolImage**(long theId);
void **CloseSubPalettes**(CToolPalette* anException = (CToolPalette *) NULL);
void **ClosePalettes**(void);
void **CloseAllPalettes**(void);

CToolPaletteAttachment* **Popup**(CWindow* theWindow,
CPoint theTopLeft,
long theSelectCommand = NULLcmd,
BOOLEAN hasTitleBar = TRUE,
CAttachmentFrame* theFrame = NULL);
CToolPaletteAttachment* **PopupDetached**(CWindow* theWindow,
CPoint theTopLeft,
long theSelectCommand = NULLcmd,
BOOLEAN hasTitleBar = TRUE,
CAttachmentFrame* theFrame = NULL);
void **SetDragSource**(CDragSource *theDragSource);
CDragSource* **GetDragSource**(void);
void **SetTitle**(const CStringRW& theNewTitle);
const CStringRW& **GetTitle**(void) const;
void **SetCommands**(long theEnterCommand,
long theLeaveCommand);
long **GetEnterCommand**(void);
long **GetLeaveCommand**(void);
CTool* **FindTool**(long theId,
CToolPalette** aFoundPalette = (CToolPalette**)NULL,
int* aFoundIndex = (int*) NULL);
CTool* **FindSubTool**(RWGSlist(CTool)* theSearchList,
long theId,
int* aFoundIndex = (int *)NULL);
virtual CToolPaletteAttachment* **ConstructAttachment**(CWindow* theWindow,
long theCommand);

Class: CToolPaletteAttachment

Superclass: CToolBarAttachment

CToolPaletteAttachment(CSubview* theEnclosure,
CToolPalette* theToolPalette,
long theCommand);
CToolPaletteAttachment(CSubview* theEnclosure,
CToolPalette* theToolPalette,
const CRect& theBestSize,
long theCommand);
CToolPaletteAttachment(const CToolPaletteAttachment &theAttachment);
CToolPaletteAttachment& **operator** = (const CToolPaletteAttachment &theAttachment);
~CToolPaletteAttachment(void);
virtual void **Detach**(BOOLEAN hasTitleBar = TRUE);
virtual void **Popup**(CWindow *theWindow,
BOOLEAN hasTitleBar = FALSE);
virtual BOOLEAN **DoUp**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);

Class: CTreeEvent

Superclass: None

CTreeEvent(EventType theType,
CTreeView* theTree,
CTreeItem* theTreeItem);

CTreeEvent(EventType theType,
CTreeView* theTree,
const CPoint& thePoint,
short theButton,
BOOLEAN theIsShift,
BOOLEAN theIsControl);
EventType **GetType**() const;
const CTreeView* **GetTree**() const;
CTreeView* **GetTree**();
const CTreeItem* **GetTreeItem**();
CTreeItem* **GetTreeItem**();
CPoint **GetPoint**() const;
short **GetButton**() const;
BOOLEAN **GetIsShift**() const;
BOOLEAN **GetIsControl**() const;
size_t **GetTabHit**() const;

Class: CTreeItem

Superclass: CNotifier

CTreeItem(CTreeNodeItem* theParent,
const CTreeAttributes* theAttributes);
virtual ~**CTreeItem**();
virtual const CFont& **SetFont**(const CFont& theFont);
virtual COLOR **SetColor**(COLOR theColor);
virtual UNITS **SetHeight**(UNITS theHeight);
virtual UNITS **SetIndent**(UNITS theIndent);
virtual CStringRW **SetString**(const CStringRW& theString);
virtual const CImage* **SetImage**(const CImage* theImage);
virtual const CImage* **SetCollapsedImage**(const CImage* theImage);
virtual const CImage* **SetExpandedImage**(const CImage* theImage);
virtual const CTabSet* **SetTabSet**(const CTabSet* theTabSet);
virtual const CTreeSource* **SetSource**(const CTreeSource* theSource);
const CFont& **GetFont**() const;
COLOR **GetColor**() const;
UNITS **GetHeight**() const;
UNITS **GetIndent**() const;
CStringRW **GetString**() const;
const CImage* **GetImage**() const;
virtual const CImage* **GetCollapsedImage**() const;
virtual const CImage* **GetExpandedImage**() const;
const CTabSet* **GetTabSet**() const;
CTabSet* **GetTabSet**();
const CTreeSource* **GetSource**() const;
const CTreeNodeItem* **GetParent**() const;
CTreeNodeItem* **GetParent**();
const RWCollectable* **GetUserData**() const;
RWCollectable* **GetUserData**();
void **SetUserData**(RWCollectable* theData);
virtual unsigned long **GetNChildren**() const;
virtual unsigned long **GetNVisibleChildren**() const;
virtual const CTreeItem* **GetChild**(unsigned long i) const;
virtual CTreeItem* **GetChild**(unsigned long i);
virtual UNITS **GetCompositeHeight**() const;

```
virtual void SortChildren(const CTreeSorter* theSorter,
        BOOLEAN theRecursive = TRUE);
virtual int compareTo(const RWCollectable*) const;
virtual void Expand(BOOLEAN theRecurse = FALSE,
        BOOLEAN theRefresh = FALSE);
virtual void Collapse(BOOLEAN theRecurse = FALSE);
virtual BOOLEAN IsTerminal() const;
virtual CTreeItemInfoInfo() const;
CTreeItem(const CTreeItem& theCopy) { }
CTreeItem& operator = (const CTreeItem& theCopy);
```

Class: CTreeNodeItem

Superclass: CTreeItem

```
CTreeNodeItem(CTreeNodeItem* itsParent,
        const CTreeAttributes* theAttributes);
virtual ~CTreeNodeItem();
virtual CTreeItem* GetChild(unsigned long i);
virtual const CTreeItem* GetChild(unsigned long i) const;
virtual void AddChild(CTreeItem* theItem);
virtual CTreeItem* AddChild(const CTreeItemInfo& theInfo,
        const CStringRW& theString = NULLString);
virtual CTreeNodeItem* AddNodeChild(const CStringRW& theString = NULLString);
virtual CTreeItem* AddTerminalChild(const CStringRW& theString = NULLString);
virtual void RemoveChild(unsigned long i);
virtual const CImage* GetExpandedImage() const;
virtual const CImage* GetCollapsedImage() const;
virtual UNITS GetCompositeHeight() const;
virtual unsigned long GetNChildren() const;
virtual unsigned long GetNVisibleChildren() const;
virtual void SortChildren(const CTreeSorter* theSorter,
        BOOLEAN theRecursive = TRUE);
virtual void Expand(BOOLEAN theRecurse = FALSE,
        BOOLEAN theRefresh = FALSE);
virtual void Collapse(BOOLEAN theRecurse = FALSE);
virtual BOOLEAN IsTerminal() const;
virtual UNITS SetHeight(UNITS theHeight);
UNITS ChildToOffset(const CTreeItem* theChild) const;
const CTreeItem& OffsetToChild(UNITS theOffset) const;
CTreeItem* OffsetToChild(UNITS theOffset);
const CTreeSorter* GetSorter() const;
BOOLEAN IsCollapsed() const;
CTreeNodeItem(const CTreeNodeItem& theCopy) :
        CTreeItem(theCopy) { }
CTreeNodeItem& operator = (const CTreeNodeItem& theCopy) ;
```

Class: CTreeItemInfo

Superclass: None

```
CTreeItemInfo() { };
CTreeItemInfo(const CTreeItemInfo& theInfo);
CTreeItemInfo& operator = (const CTreeItemInfo& theInfo);
```

Class: CTreeSource

Superclass: None

virtual RWGVector(CTreeItemInfo)* **GetTreeData**(CTreeItem* theParent) const = 0;

Class: CTreeSorter

Superclass: None

virtual Comparison **Compare**(const CTreeItem* theFirstItem,
const CTreeItem* theSecondItem) const = 0;

Class: CTreeStringSorter

Superclass: CTreeSorter

CTreeStringSorter(size_t theSortTab = 0) :
itsSortTab(theSortTab) { }
void **SetSortTab**(size_t theSortTab) {
itsSortTab = theSortTab; }
virtual Comparison **Compare**(const CTreeItem* theFirstItem,
const CTreeItem* theSecondItem) const;

Class: CTreeView

Superclass: CScroller

CTreeView(CSubview *theEnclosure,
const CRect& theRegion);
virtual **~CTreeView**(void);
BOOLEAN **ITreeView**(const CTreeSource* theSource = NULL,
COLOR theColor = COLOR_BLACK,
const CFont& theFont = CFont::GetBestCtlFont(),
UNITS theHeight = CTreeItem::kBestHeight,
UNITS theIndent = CTreeItem::kBestIndent,
const CImage* theImage = NULL,
const CImage* theCollapsedImage = NULL,
const CImage* theExpandedImage = NULL,
const CTabSet* theTabSet = NULL,
BOOLEAN hasHorizontalScrollBar = TRUE,
BOOLEAN hasVerticalScrollBar = TRUE,
BOOLEAN isThumbTracking = FALSE,
UNITS theLineIncrement = 10,
UNITS thePageIncrement = 50,
BOOLEAN isVisible = TRUE,
GLUETYPE theGlue = NULLSTICKY);
virtual void **Key**(const CKey& theKey);
virtual CRect **GetInitialFrame**(CSubview* theEnclosure,
const CRect& theRegion,
BOOLEAN hasHorizontalScrollBar,
BOOLEAN hasVerticalScrollBar) const;
virtual CRect **GetInnerFrame**() const;
virtual CRect **GetClippedFrame**() const;
void **SetDeleteUserData**(BOOLEAN theDelete);
BOOLEAN **GetDeleteUserData**() const;
virtual void **Draw**(const CRect& theClippingRegion);
virtual void **VScroll**(SCROLL_CONTROL theEventType,
UNITS thePos);

```
virtual void HScroll(SCROLL_CONTROL theEventType,
        UNITS thePos);
virtual void ScrollViews(const CPoint& theNewOrigin);
virtual void MakeItemVisible(const CTreeItem* theItem);
CTreeNodeItem* GetRoot();
const CTreeNodeItem* GetRoot() const;
virtual void SetTitleString(const CStringRW& theTitle,
        COLOR theColor = COLOR_BLACK,
        const CFont& theFont = STDFont);
virtual CStringRW GetTitleString() const;
virtual const CFont& GetTitleFont() const;
virtual CRect GetTitleRect() const;
virtual CRect ItemToRect(const CTreeItem* theItem,
        BOOLEAN theRecurse = FALSE) const;
virtual CTreeItem* PointToItem(const CPoint& thePoint);
virtual const CTreeItem* PointToItem(const CPoint& thePoint) const;
virtual void InvalidateItem(const CTreeItem* theItem,
        BOOLEAN theRecurse = FALSE);
virtual const CTreeItem* TopItem() const;
virtual CTreeItem* TopItem();
virtual const RWOrdered& GetSelection() const;
virtual void SetSelection(const RWOrdered& theSelection,
        BOOLEAN theUpdate = FALSE);
virtual void AddToSelection(CTreeItem* theItem,
        BOOLEAN theUpdate = FALSE);
virtual void RemoveFromSelection(CTreeItem* theItem,
         BOOLEAN theUpdate = FALSE);
virtual void ClearSelection(BOOLEAN theUpdate = FALSE);
virtual BOOLEAN SelectionContains(const CTreeItem* theItem);
virtual void InvalidateSelection();
virtual void SetFocusItem(CTreeItem* theItem,
        BOOLEAN theUpdate = FALSE);
virtual const CTreeItem* GetFocusItem() const;
virtual CTreeItem* GetFocusItem();
virtual void SetSelectionPolicy(SelectionPolicy thePolicy);
virtual SelectionPolicy GetSelectionPolicy() const;
virtual void SetExpansionPolicy(ExpansionPolicy thePolicy);
virtual ExpansionPolicy GetExpansionPolicy() const;
virtual void SetTreeStyle(TreeStyle theStyle);
virtual TreeStyle GetTreeStyle() const;
virtual void SetDrawRoot(BOOLEAN theDrawRoot);
virtual BOOLEAN GetDrawRoot() const;
virtual const CTreeAttributes* GetAttributes(COLOR theColor,
        const CFont& theFont,
        UNITS theHeight,
        UNITS theIndent,
        const CImage* theImage,
        const CImage* theCollapsedImage,
        const CImage* theExpandedImage,
        const CTabSet* theTabSet,
        const CTreeSource* theSource);
virtual CTreeItem* CreateTerminal(CTreeNodeItem* theParent,
        const CTreeAttributes* theAttributes);
virtual CTreeNodeItem* CreateNode(CTreeNodeItem* theParent,
        const CTreeAttributes* theAttributes);
virtual CRect DrawTitle(const CRect& theClippingRegion);
```

```
virtual void DrawItem(const CTreeItem* theItem,
        CDrawingContext& theContext,
        const CRect& theClippingRegion);
virtual void DrawSingleItem(const CTreeItem* theItem,
        CDrawingContext& theContext,
        const CRect& theClippingRegion);
virtual CPoint DrawImagePart(const CPoint& theStart,
        const CTreeItem* theItem,
        CDrawingContext& theContext,
                const CRect& theClippingRegion);
virtual void DrawSelectionPart(const CPoint& theStart,
        UNITS theTextLength,
        const CTreeItem* theItem,
        const CRect& theClippingRegion);
virtual void DrawTreePart(const CTreeItem* theItem,
        const CRect& theClippingRegion);
virtual CTreeMouseHandler* CreateMouseHandler();
CTreeView(const CTreeView& theTree) : CScroller(theTree) { };
CTreeView& operator = (const CTreeView& theTree) ;
```

Class: CTypeInfo and CTypeBaseIterator

Superclass: None

```
CTypeInfo(int theTypeId,
        const char* theClassName,
        const CTypeInfo* theBases[]);
virtual ~CTypeInfo();
const char* GetName(void) const;
int GetId(void) const;
int IsSame(const CTypeInfo* theInfo) const;
int HasBase(const CTypeInfo* theBaseInfo) const;
int CanCast(const CTypeInfo* theInfo) const;
CTypeBaseIterator(const CTypeInfo* theInfo,
        int deep = 1);
~CTypeBaseIterator(void);
const CTypeInfo* Next(void);
void Reset(void);
CTypeBaseIterator() {};
void BuildDeepTypeList(const CTypeInfo* theInfo);
```

Class: CUnits, CCentimeterUnits, CCharacterUnits, and CInchUnits

Superclass: CNotifier

CUnits(float theHScreenMapping = 1.0,
float theVScreenMapping = 1.0,
float theHPrinterMapping = 1.0,
float theVPrinterMapping = 1.0,
OutputDevice theDevice = SCREEN);
virtual **~CUnits**(void);
void **SetOwner**(CBoss* theOwner);
CBoss* **GetOwner**(void) const;
void **SetOutputDevice**(OutputDevice theDevice);
OutputDevice **GetOutputDevice**(void) const;
void **SetDynamicMapping**(BOOLEAN isMapping);
BOOLEAN **GetDynamicMapping**(void) const;
void **SetDevelopmentMetrics**(int theDeviceWidth,
int theDeviceHeight,
int theHResolution,
int theVResolution);
void **SetScreenMapping**(float theHScreenMapping,
float theVScreenMapping);
void **SetPrinterMapping**(float theHPrinterMapping,
float theVPrinterMapping);
float **GetHScreenMapping**(void) const;
float **GetVScreenMapping**(void) const;
float **GetHPrinterMapping**(void) const;
float **GetVPrinterMapping**(void) const;
int **HLogicalToPhysical**(UNITS theLogicalUnit) const;
int **VLogicalToPhysical**(UNITS theLogicalUnit) const;
RCT **LogicalToPhysical**(UNITS theLeft,
UNITS theTop,
UNITS theRight,
UNITS theBottom) const;
PNT **LogicalToPhysical**(UNITS theHPos,
UNITS theVPos) const;
UNITS **HPhysicalToLogical**(int thePhysicalUnit) const;
UNITS **VPhysicalToLogical**(int thePhysicalUnit) const;
CPoint **PhysicalToLogical**(const PNT thePnt) const;
CRect **PhysicalToLogical**(const RCT& theRct) const;
static void **SetGlobalUnits**(const CUnits* theUnits);
static const CUnits* **GetGlobalUnits**(void) const;
void **UpdateOwner**(void);
CCentimeterUnits(void);
CCharacterUnits(const CFont& theBaseFont = STDFont);
void **SetBaseFont**(const CFont& theBaseFont = STDFont);
const CFont & **GetBaseFont**(void) const;
static void **NeedsPrinterMapping**(BOOLEAN needsMapping);
void **GetCharSize**(WINDOW theXVTWindow,
int* theWidth,
int* theHeight);
CInchUnits(void);

Class: CValidator

Superclass: None

```
CValidator();
CValidator(CValidatorImplementation* theImplementation);
CValidator(const CValidator& theValidator);
CValidator& operator = (const CValidator& theValidator);
virtual ~CValidator();
virtual BOOLEAN IsValid() const;
virtual BOOLEAN TestMatch(const CStringRW& theText);
virtual CStringRW FormatString(const CStringRW& theText,
        int *theSelectionStart,
        int *theSelectionEnd);
const CValidatorImplementation* Implementation() const;
CValidatorImplementation* Implementation();
void Implementation(CValidatorImplementation* theImplementation);
```

Class: CValidatorFactory

Superclass: None

```
virtual CValidator ConstructValidator(const CStringRW& aPattern,
        BOOLEAN AutoComplete = TRUE);
virtual ~CValidatorFactory();
```

Class: CValidatorFactoryDefault

Superclass: CValidatorFactory

```
CValidatorFactoryDefault();
virtual ~CValidatorFactoryDefault();
virtual CValidator ConstructValidator(const CStringRW& aPattern,
        BOOLEAN AutoComplete = TRUE);
CValidatorFactoryDefault(const CValidatorFactoryDefault&) {};
CValidatorFactoryDefault& operator = (const CValidatorFactoryDefault&);
virtual CValidatorImplementation* ConstructImplementation (const CStringRW& aPattern,
        BOOLEAN AutoComplete);
virtual
        CValidatorImplementation* ConstructPasswordImplementation (const CStringRW& theSubstitute,
        int theMaxLength);
virtual CValidatorConstructProxy(CValidatorImplementation* theImplementation);
virtual void RemoveReference(const CValidatorImplementation& anImplementation);
```

Class: CValidatorInterface

Superclass: None

```
virtual void SetValidator(CValidator theValidator) {
        itsValidator = theValidator;}
virtual CValidator GetValidator() const {
        return itsValidator;}
```

Class: CValidatorMask

Superclass: None

```
CValidatorMask();
virtual ~CValidatorMask();
```

```
virtual BOOLEAN IValidatorMask( void );
void SetControlClass( const void * theControlClass );
const void * GetControlClass( void ) const;
virtual BOOLEAN Key( const CKey& theKey );
```

Class: CVariableGrid

Superclass: CGrid

```
CVariableGrid(CSubview* theEnclosure,
        const CRect& theRegion,
        int theNumberOfRows,
        int theNumberOfColumns);
CVariableGrid(CSubview* theEnclosure,
        CPoint& theTopLeftCorner,
        UNITS theDefaultRowHeight,
        UNITS theDefaultColumnWidth,
        int theNumberOfRows,
        int theNumberOfColumns);
CVariableGrid(const CVariableGrid& theGrid);
CVariableGrid& operator = (const CVariableGrid& theGrid);
virtual ~CVariableGrid(void);
BOOLEAN IVariableGrid(BOOLEAN isClipping = TRUE,
        PLACEMENT thePlacement = TOPLEFT,
        BOOLEAN isGridVisible = FALSE,
        POLICY theSizingPolicy = ADJUSTCellSize,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual CRect GetCellSize(int theRow,
        int theColumn) const;
virtual int GetRow(UNITS theVerticalLocation) const = 0;
virtual int GetCol(UNITS theHorizontalLocation) const = 0;
virtual UNITS GetWidth(int theColumn) const;
virtual UNITS GetHeight(int theRow) const;
virtual void Size(const CRect& theNewSize);
virtual void DoSize(const CRect& theNewSize);
virtual void AdjustCells(ADJUST theAdjustment);
virtual void SizeCol(int theColumn,
        UNITS theNewWidth);
virtual void SizeRow(int theRow,
        UNITS theNewHeight);
virtual void SetDefaultWidth(UNITS theNewDefaultWidth);
virtual void SetDefaultHeight(UNITS theNewDefaultHeight);
virtual UNITS GetDefaultWidth() const;
virtual UNITS GetDefaultHeight() const;
virtual void SizeCell(UNITS newDefaultWidth,
        UNITS newDefaultHeight);
virtual void SetNumCells(int theNumberOfColumns,
        int theNumberOfRows,
        BOOLEAN theGridWillResize = TRUE)= 0;
virtual void AdjustRow(int theRow,
        ADJUST theAdjustment);
virtual void AdjustCol(int theCol,
        ADJUST theAdjustment);
virtual void ShiftColumns(int theColumn,
        int theShift);
void ResetWidth(void);
```

```
void ResetHeight(void);
void ResetFrame(void);
virtual void ResetColumns(void);
virtual void ResetRows(void);
virtual void SetCellSizingPolicy(CELL_SIZING_POLICY theCellSizingPolicy);
virtual int GetCellSizingPolicy(void);
virtual void SetMinimumDefaultCellSize(UNITS theMinWidth,
        UNITS theMinHeight);
virtual void GetMinimumDefaultCellSize(UNITS *theMinWidth,
        UNITS *theMinHeight);
int GetRowNumber(UNITS theNewHeight);
int GetColNumber(UNITS theNewWidth);
void CopyContents(const CVariableGrid& theGrid);
```

Class: CVerticalWireFrame

Superclass: CWireFrame

```
CVerticalWireFrame(CView *theOwner,
        CSubview* theGroupEnclosure = NULL);
virtual void MouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
```

Class: CView

Superclass: CBoss

```
CView(CSubview* theEnclosure,
        const CRect& theRegion,
        const CStringRW& theTitle = NULLString);
CView(const CView& theView);
CView& operator = (const CView& theView);
virtual ~CView(void);
BOOLEAN IView(BOOLEAN isVisible = TRUE,
         GLUETYPE theGlue = NULLSTICKY);
virtual BOOLEAN IsVisible(void) const;
virtual void Show(void);
virtual void Hide(void);
virtual void DoShow(void);
virtual void DoHide(void);
virtual BOOLEAN IsActive(void) const;
virtual void Activate(void);
virtual void Deactivate(void);
virtual void DoActivate(void);
virtual void DoDeactivate(void);
virtual BOOLEAN IsEnabled(void) const;
virtual void Disable(void);
virtual void Enable(void);
virtual void DoEnable(void);
virtual void DoDisable(void);
virtual void MouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
```

```
virtual void MouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void MouseDouble(CPoint theLocation,
        short theButton= 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseDown(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseMove(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseUp(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual void DoMouseDouble(CPoint theLocation,
        short theButton = 0,
        BOOLEAN isShiftKey = FALSE,
        BOOLEAN isControlKey = FALSE);
virtual CView* FindHitView(const CPoint& theLocation) const;
virtual void Invalidate(const CRect& theFrame = MAXRect);
virtual void Draw(const CRect& theClippingRegion);
virtual void DoDraw(const CRect& theClippingRegion = MAXRect);
virtual void Key(int theKey,
        BOOLEAN isShiftKey,
        BOOLEAN isControlKey);
virtual void DoKey(const CKey&);
virtual void Size(const CRect& theNewSize);
virtual void DoSize(const CRect& theNewSize);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual BOOLEAN ClassCanGetKeyFocus() const;
virtual BOOLEAN CanGetKeyFocus() const;
virtual void VScroll(SCROLL_CONTROL theType,
        UNITS thePos);
virtual void HScroll(SCROLL_CONTROL theType,
        UNITS thePos);
virtual CRect GetFrame(void) const;
virtual CRect GetVisibleFrame(void) const;
virtual CRect GetGlobalFrame(void) const;
virtual CRect GetClippedFrame(void) const;
virtual CRect GetLocalFrame(void) const;
virtual CPoint GetOrigin(void) const;
virtual CPoint GetGlobalOrigin(void) const;
virtual void SetOrigin(const CPoint& theDeltaPoint);
virtual void DoSetOrigin(const CPoint& theDeltaPoint);
virtual const CEnvironment* GetEnvironment(void) const;
```

```
virtual void SetEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual void DoSetEnvironment(const CEnvironment& theNewEnvironment,
        BOOLEAN isUpdate = FALSE);
virtual BOOLEAN IsEnvironmentShared(void);
virtual void SetGlueObject(CGlue* theGlue,
        BOOLEAN isDeleteOld = TRUE);
virtual CGlue* GetGlueObject(void);
virtual GLUETYPE GetGlue(void) const;
virtual void SetGlue(GLUETYPE theGlue);
virtual void DoSetGlue(GLUETYPE theGlue);
virtual void Glue();
virtual BOOLEAN IsDraggable(void) const;
virtual void SetDragging(BOOLEAN isDraggable);
virtual void DoSetDragging(BOOLEAN isDraggable);
virtual BOOLEAN IsSizable(void) const;
virtual void SetSizing(BOOLEAN isSizable);
virtual void DoSetSizing(BOOLEAN isSizable);
virtual void SetWireFrame(CWireFrame* theNewWireFrame);
virtual CWireFrame* GetWireFrame(void) const;
virtual CWindow* GetCWindow(void) const;
virtual void SetCommand(long theCommand);
virtual long GetCommand(void) const;
virtual void SetDoubleCommand(long theCommand);
virtual long GetDoubleCommand(void) const;
virtual void SetTitle(const CStringRW& theNewTitle);
virtual const CStringRW GetTitle(void) const;
virtual CSubview* GetEnclosure(void) const;
virtual void SetEnclosure(CSubview* theEnclosure);
virtual WINDOW GetXVTWindow(void) const;
WIN_TYPE GetXVTType(void) const;
virtual BOOLEAN DoPrint(const CRect& theRegion) const;
virtual void DoPrintDraw(const CRect& theRegion);
virtual void PrintDraw(const CRect& theRegion);
virtual const RWOrdered* GetSubObjects(void) const;
virtual void DoCommand(long theCommand,
        void* theData = NULL);
virtual CView* FindDeepSubview(const CPoint& theLocation) const;
virtual CView* FindSubview(const CPoint& theLocation) const;
virtual CView* FindSubview(long theViewId) const;
virtual CView* GetSelectedView(void) const;
virtual CView* FindEventTarget(const CPoint& theLocation) const;
virtual const RWOrdered* GetSubviews(void) const;
virtual CView* FindResourceView(long theContainerId,
        long theId) const;
virtual BOOLEAN HitTest(const CPoint &theLoc) const;
virtual void DoTimer(long theTimerId);
virtual void DoUser(long theUserId,
        void* theData);
virtual void DoUpdateModel(long theControllerId,
        long theCommand,
        const CModel* theModel);
virtual BOOLEAN Prepare(const CRect& theClippingRegion);
void BuildView(CSubview* theEnclosure,
        const CRect& theRegion);
void CopyView(const CView& theView);
```

Class: CViewFactory

Superclass: None

```
virtual CText* ConstructCText(CSubview*,
        const CPoint&,
        const CStringRW&);
virtual CPicture* ConstructCPicture(CSubview*g
        const CPoint&,
        const CImage&);
virtual CFixedGrid* ConstructCFixedGrid(CSubview*,
        const CPoint&,
        UNITS,
        UNITS,
        int,
        int);
virtual CGlue* ConstructGlue(CView*);
virtual CWireFrame* ConstructWireFrame(CView*);
virtual ~CViewFactory();
CViewFactory();
```

Class: CViewFactoryDefault

Superclass: CViewFactory

```
CViewFactoryDefault();
virtual CText* ConstructCText(CSubview*,
        const CPoint&,
        const CStringRW&);
virtual CPicture* ConstructCPicture(CSubview*,
        const CPoint&,
        const CImage&);
virtual CFixedGrid* ConstructCFixedGrid(CSubview*,
        const CPoint&,
        UNITS,
        UNITS,
        int,
        int);
virtual CGlue* ConstructGlue(CView*);
virtual CWireFrame* ConstructWireFrame(CView*);
```

Class: CViewSink

Superclass: CViewSink

```
CViewSink(CView* theView,
        long theDropCommand,
        long theEnterCommand = NULLcmd,
        long theLeaveCommand = NULLcmd,
        long theDragCommand = NULLcmd);
CViewSink(const CViewSink& theSink);
CViewSink& operator = (const CViewSink &theSink);
virtual ~CViewSink();
void SetSinkCommands(long theDropCommand,
        long theEnterCommand = NULLcmd,
        long theLeaveCommand = NULLcmd,
        long theDragCommand = NULLcmd);
long GetDropCommand() const;
long GetEnterCommand() const;
long GetLeaveCommand() const;
long GetDragCommand() const;
void SetView(CView* theView);
CView* GetView(void) const;
virtual BOOLEAN IsInSink(CPoint theLocation);
virtual void DoDrop(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual void DoEnter(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual void DoLeave(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual void DoDrag(CPoint theLocation,
        short theButton,
        BOOLEAN isShift,
        BOOLEAN isControl,
        long theDragCommand,
        void* theDragData);
virtual CView* GetOwner(void);
```

Class: CViewSource

Superclass: CDragSource

CViewSource(CView* theView,
long theDragCommand,
void* theDragData,
CURSOR theDragCursor = CURSOR_ARROW);
CViewSource(const CViewSource& theCopy);
CViewSource& **operator** = (const CViewSource& theCopy);
virtual **~CViewSource**();
CView* **GetView**() const;
void **SetView**(CView* theView);
long **GetDragCommand**() const;
void* **GetDragData**() const;
void **SetData**(long theDragCommand,
void* theDragData);
CURSOR **GetDragCursor**(void) const;
void **SetDragCursor**(CURSOR theCursor);
void **SetDragSensitivity**(CPoint thePoint);
CPoint **GetDragSensitivity**() const;
virtual BOOLEAN **DoUp**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);
virtual BOOLEAN **DoDown**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);
virtual BOOLEAN **DoMove**(CPoint& theLocation,
short& theButton,
BOOLEAN& isShiftKey,
BOOLEAN& isControlKey);

Class: CVirtualFrame

Superclass: CSubview

BOOLEAN **IVirtualFrame**(BOOLEAN isVisible = TRUE,
GLUETYPE theGlue = NULLSTICKY);
virtual **~CVirtualFrame**(void);
virtual void **SetScrollingOrigin**(const CPoint& theNewOrigin);
virtual CPoint& **GetScrollingOrigin**(void);
virtual void **SetVirtualFrame**(UNITS theNewWidth,
UNITS theNewHeight);
virtual CRect **GetVirtualFrame**(void) const;
virtual void **EnlargeToFit**(const CRect& theRegionToInclude);
virtual void **ShrinkToFit**(void);
virtual void **Draw**(const CRect& theClippingRegion);
virtual void **Size**(const CRect& theNewSize);
virtual CPoint **GetGlobalOrigin**(void) const;
CVirtualFrame(CSubview* theEnclosure,
const CRect& theRegion,
UNITS theVirtualWidth =0,
UNITS theVirtualHeight =0);
CVirtualFrame(const CVirtualFrame& theVirtualFrame);
CVirtualFrame& **operator** = (const CVirtualFrame& theVirtualFrame);
virtual void **ScrollViews**(const CPoint& theNewOrigin);

virtual void **AdjustScrollBars**(void) = 0;
virtual void **AddSubview**(const CView* theSubview);

Class: CWindow

Superclass: CSubview

CWindow(CDocument* theDocument,
const CRect& theRegion,
const CStringRW& theTitle = NULLString,
long theWindowAttributes = WSF_NONE,
WIN_TYPE theWindowType = W_DOC,
int theMenuBarId = MENU_BAR_RID,
WINDOW theEnclosure = TASK_WIN);
CWindow(CDocument* theDocument, WINDOW theXVTWindow);
CWindow(CDocument* theDocument,
long theId,
BOOLEAN isCreatedViews,
WINDOW theParent = TASK_WIN);
CWindow(const CWindow& theWindow);
BOOLEAN **IWindow**(BOOLEAN isBackgroundDrawn = FALSE,
const CStringRW& theTitle = NULLString,
BOOLEAN isVisible = TRUE);
virtual void **DoCommand**(long theCommand,
void* theData = NULL);
virtual void **UpdateMenus**(CMenuBar* theMenuBar);
virtual BOOLEAN **Close**(void);
virtual WINDOW **GetXVTWindow**(void) const;
WIN_TYPE **GetXVTType**(void) const;
virtual long **GetAttributes**(void) const;
virtual int **GetMenuBarId**(void) const;
virtual BOOLEAN **IsClosable**(void) const;
virtual BOOLEAN **IsSizable**(void) const;
virtual CDocument* **GetDocument**(void);
virtual BOOLEAN **IsBackgroundDrawn**(void) const;
virtual void **SetBackgroundDrawing**(BOOLEAN isBackgroundDrawn);
virtual NWinScrollBar* **GetVScrollBar**(void) const;
virtual NWinScrollBar* **GetHScrollBar**(void) const;
virtual void **Key**(int theKey,
BOOLEAN isShiftKey,
BOOLEAN isControlKey);
virtual void **Size**(const CRect& theNewSize);
virtual void **Draw**(const CRect& theClippingRegion);
virtual void **Hide**(void);
virtual void **Show**(void);
virtual void **Enable**(void);
virtual void **Disable**(void);
virtual void **DoDisable**(void);
virtual void **DoEnable**(void);
Cstatic void **SetCreationIcon**(long theId);
virtual void **SetEnclosure**(CSubview* theEnclosure);
virtual void **SetTitle**(const CStringRW& theNewTitle);
virtual void **SetSelectedView**(CView *theSubview);
virtual CRect **GetClippedFrame**(void) const;
virtual const CEnvironment* **GetEnvironment**(void) const;

virtual void **DoMenuCommand**(MENU_TAG theMenuItem,
BOOLEAN isShiftKey,
BOOLEAN isControlKey);
virtual void **ChangeFont**(const CFont &theFont);
virtual void **DoModal**(void);
void **SetMenuBar**(CMenuBar* theMenuBar,
BOOLEAN isDeleteOld = TRUE);
CMenuBar* **GetMenuBar**(void);
virtual void **SizeWindow**(int theWidth,
int theHeight);
virtual void **DoControl**(int theControlID,
CONTROL_INFO theControlInfo);
virtual void **DoActivateWindow**(void);
virtual void **DoDeactivateWindow**(void);
virtual void **DoTimer**(long theTimerId);
virtual void **DoUser**(long theUserId,
void* theData);
TXEDdIT **GetHiddenXVTTextEdit**(void) const;
virtual CMouseManager* **GetMouseManager**(void);
virtual void **SetMouseManager**(CMouseManager* theMouseManager);
virtual CNavigator* **GetNavigator**(void);
virtual void **SetNavigator**(CNavigator* theNavigator);
BOOLEAN **IsQueuedUpdate**(void) const;
void **QueueUpdates**(void);
void **UnionQueuedUpdates**(RCT &theRect);
void **FlushQueuedUpdates**(void);
BOOLEAN **QueueSize**(int theWidth,
int theHeight);
void **SuspendSizing**();
void **ResumeSizing**();
CWindow& **operator** = (const CWindow& theWindow);
virtual **~CWindow**(void);
void **DoHScroll**(SCROLL_CONTROL theEvent,
short thePos);
void **DoVScroll**(SCROLL_CONTROL theEvent,
short thePos);
virtual void **UpdateUnits**(CUnits* theUnits);
virtual void **CreateWindow**(WINDOW the Enclosure);
virtual void **InitWindow**(void);
virtual CMouseManager* **ConstructMouseManager**();
virtual CNavigator* **ConstructNavigator**

Class: CWindowFactory

Superclass: None

virtual CMouseManager* **ConstructMouseManager**(CWindow*,
CMouseManager* theParent);
virtual CNavigator* **ConstructNavigator**(CWindow*);
virtual NWinScrollBar* **ConstructHorizScrollBar**(CWindow*);
virtual NWinScrollBar* **ConstructVertScrollBar**(CWindow*);
virtual ~**CWindowFactory**();
CWindowFactory();

Class: CWindowFactoryDefault

Superclass: CWindowFactory

CWindowFactoryDefault();
virtual CMouseManager* **ConstructMouseManager**(CWindow*,
CMouseManager* theParent);
virtual CNavigator* **ConstructNavigator**(CWindow*);
virtual NWinScrollBar* **ConstructHorizScrollBar**(CWindow*);
virtual NWinScrollBar* **ConstructVertScrollBar**(CWindow*);

Class: CWindowNavigator

Superclass: CNavigator

CWindowNavigator(CWindow* theWindow);
~**CWindowNavigator**();

Class: CWireFrame

Superclass: CView

CWireFrame(CView *theOwner,
CSubview* theGroupEnclosure = NULL);
CWireFrame(const CWireFrame& theFrame);
CWireFrame& **operator** = (const CWireFrame& theFrame);
virtual **~CWireFrame**(void);
virtual BOOLEAN **IWireFrame**(BOOLEAN isSizable = FALSE,
BOOLEAN isDraggable = FALSE,
UNITS theMinimalWidth = 0,
UNITS theMinimalHeight = 0);
virtual void **SetDragging**(BOOLEAN isDraggable);
virtual void **SetSizing**(BOOLEAN isSizable);
virtual BOOLEAN **IsSizable**(void) const;
virtual BOOLEAN **IsDraggable**(void) const;
virtual BOOLEAN **IsSelected**(void) const;
virtual void **Select**(void);
virtual void **DoDeselect**(CView* theView);
virtual void **Deselect**(CView* theView);
virtual CView* **GetOwner**(void) const;
virtual CSubview* **GetGroupEnclosure**(void) const;
virtual void **SetGroupEnclosure**(CSubview* theGroupEnclosure) const;
virtual void **MouseDouble**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
virtual void **MouseDown**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
virtual void **MouseMove**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);
virtual void **MouseUp**(CPoint theLocation,
short theButton = 0,
BOOLEAN isShiftKey = FALSE,
BOOLEAN isControlKey = FALSE);

```
virtual void Draw(const CRect& theClippingRegion);
virtual void Size(const CRect& theNewSize);
virtual void SetOrigin(const CPoint& theDeltaPoint);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
BOOLEAN SizeOwner(void);
virtual void DrawWireFrame(BOOLEAN isDrawing);
virtual void DrawFrameGrabbers(BOOLEAN isDrawing);
virtual void Drag(CPoint theNewLocation);
virtual void ReSize(CPoint theNewLocation);
void SetGroupRegion(void);
void ClearGroupRegion(void);
virtual BOOLEAN DoDraggingCheck(const CPoint& theLocation);
virtual BOOLEAN DoBorderCheck(const CPoint& theLocation);
virtual void DoAutoScroll(UNITS theHorizontalIncrement,
        UNITS theVerticalIncrement);
```

Class: Mem

Superclass: None

```
void * operator new(size_t sz,
        char* file,
        int line);
void operator delete(void *p);
void DumpMemory(void);
void CleanMemory(void);
```

Class: NButton

Superclass: CNativeView

```
NButton(CSubview* theEnclosure,
        const CRect& theRegion,
        const CStringRW theTitle = NULLString,
        long theAttributes = 0L);
NButton(CSubview* theEnclosure,
        long theContainerId,
        long theId);
NButton(const NButton& theButton);
NButton& operator = (const NButton& theButton);
virtual ~NButton();
BOOLEAN IButton(const CStringRW theTitle = NULLString,
        BOOLEAN isEnabled = TRUE,
        long theCommand = NULLcmd,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void DoHit(CONTROL_INFO theControlInfo);
virtual void Key(int theKey,
        BOOLEAN isShiftKey = TRUE,
        BOOLEAN isControlKey = TRUE);
virtual BOOLEAN ClassCanGetKeyFocus(void) const;
```

Class: NCheckBox

Superclass: CNativeView

NCheckBox(CSubview* theEnclosure,
const CRect& theRegion,
const CStringRW theTitle = NULLString,
long theAttributes = 0L);
NCheckBox(CSubview* theEnclosure,
long theContainerId,
long theId);
NCheckBox(const NCheckBox& theCheckBox);
NCheckBox& **operator** = (const NCheckBox& theCheckBox);
virtual **~NCheckBox**();
BOOLEAN **ICheckBox**(const CStringRW& theTitle = NULLString,
BOOLEAN isEnabled = TRUE,
long theDeselectCommand = NULLcmd,
long theSelectCommand = NULLcmd,
BOOLEAN isVisible = TRUE,
long theGlue = NULLSTICKY);
virtual BOOLEAN **IsSelected**(void) const;
virtual void **Select**(void);
virtual void **Deselect**(void);
virtual long **GetSelectCommand**(void) const;
virtual void **SetSelectCommand**(long theNewSelectCommand);
virtual long **GetDeselectCommand**(void) const;
virtual void **SetDeselectCommand**(long theNewSelectCommand);
virtual void **DoHit**(CONTROL_INFO theControlInfo);
virtual BOOLEAN **ClassCanGetKeyFocus**(void) const;
virtual void **DoUpdateModel**(long theControllerId,
long theCommand,
const CModel* theModel);
virtual void **DoCommand**(long theCommand,
void* theData = NULL);

Class: NEditControl

Superclass: CNativeView

NEditControl(CSubview* theEnclosure,
const CRect& theRegion,
const CStringRW& theTitle = NULLString,
long theAttributes = 0L);
NEditControl(CSubview* theEnclosure,
long theContainerId,
long theId);
NEditControl(const NEditControl& theEditControl);
NEditControl& **operator** = (const NEditControl& theEditControl);
virtual **~NEditControl**();
virtual void **Deactivate**(void);
virtual void **Activate**(void);
void **DoHit**(CONTROL_INFO theInfo);
virtual void **SetFocusCommands**(long theKeyFocusCommand,
long theKeyFocusLostCommand);
virtual void **SetTextCommand**(long theTextCommand);
long **GetKeyFocusCommand**(void) const;
long **GetKeyFocusLostCommand**(void) const;
long **GetTextCommand**(void) const;

virtual void **SetTitle**(const CStringRW& theNewTitle);
void **GetTextSelection**(int *theFirstIndex,
int *theLastIndex) const;
void **SetTextSelection**(int theFirstIndex,
int theLastIndex);
virtual BOOLEAN **ClassCanGetKeyFocus**(void) const;
virtual void **SetValidator**(CValidator theValidator ;

Class: NGroupBox

Superclass: CNativeView

NGroupBox(CSubview* theEnclosure,
const CRect& theRegion,
const CStringRW& theTitle = NULLString,
long theAttributes = 0L);
NGroupBox(CSubview* theEnclosure,
long theContainerId,
long theId);
NGroupBox(const NGroupBox& theGroupBox);
NGroupBox& **operator** = (const NGroupBox& theGroupBox);
virtual **~NGroupBox**();
void **DoHit**(CONTROL_INFO theInfo);
virtual void **Activate**(void);

Class: NIcon

Superclass: CNativeView

NIcon(CSubview* theEnclosure,
const CRect& theRegion,
long theIconId,
long theAttributes = 0L);
NIcon(CSubview* theEnclosure,
long theContainerId,
long theId);
NIcon(const NIcon& theIcon);
NIcon& **operator** = (const NIcon& theIcon);
virtual **~NIcon**();
virtual void **DoHit**(CONTROL_INFO theControlInfo);
virtual void **Activate**(void);

Class: NLineText

Superclass: CNativeTextEdit

NLineText(CSubview* theEnclosure,
const CPoint& theTopLeftPoint,
UNITS theLength,
unsigned theAttributes = TX_BORDER,
UNITS theRightMargin = 1000,
int theCharacterLimit = 1000);
NLineText(const NLineText& theLineText);
NLineText& **operator** = (const NLineText& theLineText);
virtual **~NLineText**();

```
BOOLEAN ILineText(unsigned theAttributes = TX_BORDER,
        UNITS theRightMargin = 1000,
        int theCharactedLimit = 1000,
        const CStringRW& theInitialText = NULLString,
        BOOLEAN isAutoSelected = FALSE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = SAMESTICKY);
T_CNUM GetInsertPosition(void) const;
void SetInsertPosition(T_CNUM thePos);
virtual void SetAttribute(unsigned theAttribute,
        BOOLEAN isSet = TRUE);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void Size(const CRect& theNewSize);
virtual void Key(int theKey,
        BOOLEAN isShift,
        BOOLEAN isControl);
void SetValidation(RWCRegexp theExpression);
RWCRegexp GetValidation(void) const;
void KeyValidateRegX(const CKey&);
int ResetHeight(const CFont& theFont,
        const CWindow& theWindow) const;
CRect GetInitialFrame(CSubview* theEnclosure,
        const CPoint& theTopLeft,
        UNITS theLength);
```

Class: NListBox

Superclass: CNativeSelectList

```
NListBox(CSubview* theEnclosure,
        const CRect& theRegion,
        const CStringCollection& theItems,
        long theControlAttributes = CTL_FLAG_MULTIPLE);
NListBox(CSubview* theEnclosure,
        long theContainerId,
        long theId);
BOOLEAN IListBox(const CStringCollection& theItems);
NListBox(const NListBox& theNativeList);
NListBox& operator = (const NListBox& theNativeList);
virtual ~NListBox();
virtual void DoHit(CONTROL_INFO theControlInfo);
virtual void Size(const CRect& theRect);
```

Class: NListButton

Superclass: CNativeSelectList

```
NListButton(CSubview* theEnclosure,
        const CRect& theRegion,
        const CStringCollection& theItems,
        int thePosition = 0,
        long theControlAttributes = CTL_FLAG_NATIVE_JUST);
NListButton(CSubview* theEnclosure,
        long theContainerId,
        long theId);
BOOLEAN IListButton(const CStringCollection& theItems,
        int thePosition = 0);
NListButton(const NListButton& theNativeList);
```

NListButton& **operator** = (const NListButton& theNativeList);
virtual **~NListButton**();
virtual CRect **GetVisibleFrame**(void) const;
virtual void **DoHit**(CONTROL_INFO theControlInfo);

Class: NListEdit

Superclass: CNativeList

NListEdit(CSubview* theEnclosure,
const CRect& theRegion,
const CStringCollection& theItems,
const CStringRW& theTitle = NULLString);
NListEdit(CSubview* theEnclosure,
long theContainerId,
long theId);
NListEdit(const NListEdit& theNativeList);
BOOLEAN **IListEdit**(const CStringCollection& theItems,
const CStringRW& theTitle = NULLString);
NListEdit& **operator** = (const NListEdit& theNativeList);
virtual **~NListEdit**();
virtual void **SetFocusCommands**(long theKeyFocusCommand,
long theKeyLostFocusCommand);
virtual void **SetTextCommand**(long theTextCommand);
long **GetKeyFocusCommand**(void) const;
long **GetKeyFocusLostCommand**(void) const;
long **GetTextCommand**(void) const;
virtual void **DoHit**(CONTROL_INFO theControlInfo);
virtual CRect **GetVisibleFrame**(void) const;
virtual void **Activate**(void);
virtual void **Deactivate**(void);
void **GetTextSelection**(int *theFirstIndex,
int *theLastIndex) const;
void **SetTextSelection**(int theFirstIndex,
int theLastIndex);

Class: NNotebook

Superclass: CNativeView

NNotebook (CSubview *theEnclosure,
const CRect& theRegion);
NNotebook (const NNotebook& theNotebook);
~NNotebook ();
BOOLEAN **INotebook** ();
virtual void **DoHit** (CONTROL_INFO theCtlInfo);
virtual **AddPage** (short theTab,
short thePage,
const CStringRW& theTitle);
virtual **AddTab** (short theTab,
const CStringRW& theTitle,
CImage theTabImage);
virtual const **CStringRW GetTitle** (void) const;
virtual **CFaceWindow* GetFace** (short theTab,
short thePage);
virtual **CFaceWindow* GetFacePage** (CFaceWindow* theFace,
short* theTab,
short* thePage);

virtual **CFaceWindow* GetFrontPage** (short* theTab,
short* thePage);
virtual **CStringRW GetPageTitle** (short theTab,
short thePage) const;
virtual **CStringRW GetTabTitle** (short theTab) const;
virtual **CImage GetTabImage** (short theTab);
virtual short **GetNumPages** (short theTab);
virtual short **GetNumTabs** ();
virtual void **RemovePage** (short theTab, short thePage);
virtual void **RemoveTab** (short theTab);
virtual void **SetFace** (CFaceWindow* theFace,
short theTab,
short thePage,
long theId);
virtual void **SetPageTitle**(short theTab,
short thePage,
CStringRW theTitle);
virtual void **SetFrontPage**(short theTab,
short thePage);
virtual void **SetTabTitle**(short theTab,
CStringRW theTitle);
virtual void **SetTabImage**(short theTab,
CImage theImage);
virtual void **SetTabHotKey** (short theTab,
short thePage,
const CKey& theKey);
virtual **CNavigator* GetNavigator**();

Class: NRadioButton

Superclass: CNativeView

NRadioButton(CSubview *theEnclosure,
const CRect& theRegion,
CRadioGroup* theRadioGroup,
long theAttributes = 0L,
const CStringRW& theTitle = NULLString);
NRadioButton(RadioGroup *theEnclosure,
long theContainerId,
long theId);
NRadioButton(const NRadioButton& theButton);
NRadioButton& **operator** = (const NRadioButton& theButton);
virtual **~NRadioButton**(void);
BOOLEAN **IRadioButton**(const CStringRW& theTitle = NULLString,
BOOLEAN isEnabled = TRUE,
long theCommand = NULLcmd,
BOOLEAN isVisible = TRUE,
long theGlue = NULLSTICKY);
virtual void **DoHit**(CONTROL_INFO controlInfo);
virtual BOOLEAN **ClassCanGetKeyFocus**(void) const;

Class: NRadioNavigator

Superclass: CRadioNavigator

NRadioNavigator(CRadioGroup* theRadioGroup,
CSubview* theEnclosure = NULL);
virtual void **ActivateTabStop**(CTabStop*);

```
static RWBoolean TabStopTestViewIdIs(const CTabStop* theTabStop,
        long theId);
virtual CTabStop* PickTabStop(RWtestGeneric
        theTabStopTest = 0,
        void* = NULL,
        RWtestGeneric theNavTest = 0,
        void* = NULL);
```

Class: NScrollBar

Superclass: CNativeView

```
NScrollBar(CSubview* theEnclosure,
        const CRect& theRegion,
        DIRECTION theDirection,
        long theAttributes = 0L);
NScrollBar(CSubview* theEnclosure,
        long theContainerId,
        long theId);
NScrollBar(const NScrollBar& theScrollBar);
NScrollBar& operator = (const NScrollBar& theCSrollBar);
virtual ~NScrollBar();
BOOLEAN IScrollBar(UNITS theMinPosition = 0,
        UNITS theMaxPosition = 0,
        UNITS theCurrentPosition = 0,
        UNITS theLineIncrement = 1,
        UNITS thePageIncrement = 20,
        BOOLEAN isEnabled = TRUE,
        BOOLEAN isVisible = TRUE,
        GLUETYPE theGlue = NULLSTICKY);
virtual void SetRange(UNITS theMinPosition,
        UNITS theMaxPosition,
        UNITS theCurrentPosition = SAME);
virtual UNITS GetMinPosition(void) const;
virtual UNITS GetMaxPosition(void) const;
virtual void SetPosition(UNITS theNewPosition);
virtual UNITS GetPosition(void) const;
virtual void SetIncrements(UNITS theLineIncrements,
        UNITS thePageIncrements);
virtual UNITS GetLineIncrement(void) const;
virtual UNITS GetPageIncrement(void) const;
virtual void SetOwner(CView* theOwner);
virtual CView* GetOwner(void) const;
virtual void SetThumbSize(UNITS theThumbSize);
virtual UNITS GetThumbSize(void) const;
void InsertOwner(CView* theOwner);
void RemoveOwner(CView* theOwner);
virtual const RWOrdered* GetOwners() const;
static int NativeWidth(void);
static int NativeHeight(void);
virtual void DoHit(CONTROL_INFO theControlInfo);
virtual void Size(const CRect& theNewSize);
virtual const CStringRW GetTitle(void) const;
virtual void VScroll(SCROLL_CONTROL theEventType,
        UNITS thePosition);
virtual void HScroll(SCROLL_CONTROL theEventType,
        UNITS thePosition);
virtual void Show(void);
```

virtual void **DoUpdateModel**(long theControllerId,
long theCommand,
const CModel* theModel);
virtual void **DoCommand**(long theCommand,
void* theData = NULL);
NScrollBar(WINDOW theCreatedControl,
CSubview* theEnclosure,
const CRect& theRegion,
DIRECTION theDirection);
virtual SCROLL_TYPE **GetSBType**(void) const;

Class: NScrollText

Superclass: NTextEdit

NScrollText(CSubview* theEnclosure,
const CRect& theRegion,
BOOLEAN hasHorizontalScrollBar = TRUE,
BOOLEAN hasVerticalScrollBar = TRUE,
unsigned theAttributes = TX_BORDER,
UNITS theRightMargin = 1000,
int theCharacterLimit = 1000,
BOOLEAN isVisible = TRUE);
NScrollText(CSubview* theEnclosure,
long theContainerId,
long theId);
NScrollText(CWindow* theScrollableWindow,
unsigned theAttributes = TX_BORDER,
UNITS theRightMargin = 1000,
int theCharacterLimit = 1000,
BOOLEAN isVisible = TRUE);
NScrollText(const NScrollText& theScrollText);
NScrollText& **operator** = (const NScrollText& theScrollText);
virtual **~NScrollText**(void);
BOOLEAN **IScrollText**(BOOLEAN isThumbTracking = TRUE,
unsigned theAttributes = TX_BORDER,
UNITS theRightMargin = 1000,
int theCharacterLimit = 1000,
const CStringRW theInitialText = NULLString,
BOOLEAN isAutoSelected = FALSE,
BOOLEAN isVisible = TRUE,
long theGlue = NULLSTICKY);
virtual void **SetHIncrements**(UNITS theLineIncrement,
UNITS thePageIncrement);
virtual void **SetVIncrements**(UNITS theLineIncrement,
UNITS thePageIncrement);
virtual UNITS **GetHLineIncrement**(void) const;
virtual UNITS **GetVLineIncrement**(void) const;
virtual UNITS **GetHPageIncrement**(void) const;
virtual UNITS **GetVPageIncrement**(void) const;
static void **ScrollTextCallback**(TXEDIT theTextEdit,
T_LNUM theOriginLine,
T_LNUM theNumberOfLines,
T_CNUM theCharacterOffset);
void **GetScrollPosition**(T_PNUM* theParagraph,
T_LNUM* theParagraphLine,
T_LNUM* theOriginLine,
T_CNUM* theCharacterOffset);

```
void SetScrollPosition(T_LNUM theOriginLine,
        T_CNUM theCharacterOffset);
void SetScrollPosition(T_PNUM theParagraph,
        T_LNUM theParagraphLine,
        T_CNUM theCharacterOffset);
virtual void VScroll(SCROLL_CONTROL theEventType,
        UNITS thePosition);
virtual void HScroll(SCROLL_CONTROL theEventType,
        UNITS thePosition);
virtual void SetMargin(UNITS theRightMargin);
virtual void SetFont(const CFont &theNewFont,
        BOOLEAN isUpdate = FALSE);
virtual void Size(const CRect& theNewSize);
virtual void Hide(void);
virtual void Show(void);
virtual void SetEnclosure(CSubview* theEnclosure);
void CreateScrollBars(BOOLEAN hasVScrollBar,
        BOOLEAN hasHScrollBar,
        UNITS theLineIncrement,
        UNITS thePageIncrement);
CRect GetInitialFrame(CSubview* theEnclosure,
        const CRect& theRegion,
        BOOLEAN hasHorizontalScrollBar,
        BOOLEAN hasVerticalScrollBar);
CRect GetInnerFrame(void);
virtual void PositionTextEdit(void);
void RemoveScrollBar(const NScrollBar* theScrollBar);
```

Class: NText

Superclass: CNativeView

```
NText(CSubview* theEnclosure,
        const CRect& theRegion,
        const CStringRW& theTitle = NULLString,
        long theAttributes = 0L;
NText(CSubview* theEnclosure,
        long theContainerId,
        long theId);
NText(const NText& theText);
NText& operator = (const NText& theText);
virtual ~NText();
void DoHit(CONTROL_INFO theInfo);
```

Class: NTextEdit

Superclass: CNativeTextEdit

```
NTextEdit(CSubview* theEnclosure,
        const CRect& theRegion,
        unsigned theAttributes = TX_BORDER,
        int theRightMargin = 1000,
        int theCharacterLimit = 1000);
NTextEdit(const NTextEdit& theTextEdit);
NTextEdit& operator = (const NTextEdit& theTextEdit);
virtual ~NTextEdit();
```

```
BOOLEAN ITextEdit(unsigned theAttributes = TX_BORDER,
        int theRightMargin = 1000,
        int theCharacterLimit = 1000,
        const CStringRW theInitialText = NULLString,
        BOOLEAN isAutoSelected = FALSE,
        BOOLEAN isVisible = TRUE,
        long theGlue = NULLSTICKY);
void GetInsertPosition(T_PNUM* thePar,
        T_LNUM* theLine,
        T_CNUM* theChar) const;
void SetInsertPosition(T_PNUM thePar,
        T_LNUM theLine,
        T_CNUM theChar);
virtual BOOLEAN SetParagraph(const CStringRW& theText,
        T_PNUM theParagraph);
virtual BOOLEAN AddParagraph(const CStringRW& theText,
        T_PNUM theParagraph);
virtual BOOLEAN AppendToParagraph(const CStringRW& theText,
        T_PNUM theParagraph);
virtual BOOLEAN DeleteParagraph(T_PNUM theParagraph);
virtual CStringRW GetParagraph(T_PNUM theParagraph) const;
virtual void SelectParagraph(T_PNUM theParagraph,
        T_LNUM theStartLine = 0,
        T_LNUM theEndLine = LAST,
        T_CNUM theStartChar = 0,
        T_CNUM theEndChar = LAST);
virtual T_CNUM GetNCharInPar(T_PNUM theParagraph) const;
virtual T_PNUM GetNParInText(void) const;
virtual T_PNUM GetNParInSelection(void) const;
virtual void SetLine(const CStringRW& theText,
        T_PNUM theParagraph,
        T_LNUM theLine);
virtual CStringRW GetLine(T_PNUM theParagraph,
        T_LNUM theLine) const;
virtual void SelectLine(T_LNUM theLine = 0,
        T_PNUM theParagraph = 0,
        T_CNUM theStartingChar = 0,
        T_CNUM theEndingChar = LAST);
virtual T_CNUM GetNCharInLine(T_PNUM theParagraph = 0,
        T_LNUM theLine = 0) const;
virtual T_LNUM GetNLineInPar(T_PNUM theParagraph) const;
virtual T_LNUM GetNLineInText(void) const;
virtual T_LNUM GetNLineInSelection(void) const;
virtual CStringRW GetSomeText(T_PNUM theStartParagraph,
        T_PNUM theEndParagraph,
        T_LNUM theStartLine,
        T_LNUM theEndLine,
        T_CNUM theStartChar,
        T_CNUM theEndChar);
virtual void SelectSomeText(T_PNUM theStartParagraph,
        T_PNUM theEndParagraph,
        T_LNUM theStartLine,
        T_LNUM theEndLine,
        T_CNUM theStartChar,
        T_CNUM theEndChar);
NTextEdit(CSubview* theEnclosure);
```

Class: NWinScrollBar

Superclass: NScrollBar

NWinScrollBar(CSubview* theEnclosure,
DIRECTION theDirection);
NWinScrollBar(const NWinScrollBar& theScrollBar);
NWinScrollBar& **operator** = (const NWinScrollBar& theScrollBar);
virtual **~NWinScrollBar**();
virtual void **Size**(const CRect& theNewSize);
virtual void **SetTitle**(const CStringRW& theTitle);
virtual void **Hide**(void);
virtual void **Show**(void);
virtual void **SetEnclosure**(CSubview *theEnclosure);
virtual void **Enable**(void);
virtual void **Disable**(void);
virtual SCROLL_TYPE **GetSBType**(void) const;
virtual void **Close**(void);
virtual void **CreateControl**(void);

Predefined Values

Default Menubar IDs

```
#ifndef MENU_BAR_RID
#define MENU_BAR_RID 9001
```

Default Parameters

```
#define LAST 0x0000FFFF          // Last item in a list
#define SAME 0xF0F0F0F0          // Use the previously-defined value
```

Directional Flags

```
#define PWRTOP                   0x0001
#define PWRBOTTOM                0x0002
#define PWRLEFT                  0x0004
#define PWRRIGHT                 0x0008
typedef enum {VERTICAL, HORIZONTAL} DIRECTION;
typedef DIRECTION SBTYPE;
```

Error Handling

```
#include PWR_INCL_PATH( pwr, PwrError.h, pwrerror.h )
```

Globals

```
#define MENUSeparator (CObjectRWC::GetG()
      ->GetMENUSeparator())
#define MAXRect (CObjectRWC::GetG()->GetMAXRect())
#define STDFont (CObjectRWC::GetG()->GetSTDFont())
#define NULLCRegion (CObjectRWC::GetG()
      ->GetNULLCRegion())
#define NULLString (CObjectRWC::GetG()->GetNULLString())
```

Glue Types

```
typedef long GLUETYPE;
#define LEFTSTICKY 0xFF000000L
#define TOPSTICKY 0x00FF0000L
#define RIGHTSTICKY 0x0000FF00L
#define BOTTOMSTICKY 0x000000FFL
#define SAMESTICKY 0xF0F0F0F0L
#define ALLSTICKY 0xFFFFFFFFL
#define TOPLEFTSTICKY 0xFFFF0000L
#define TOPRIGHTSTICKY 0x00FFFF00L
#define BOTTOMLEFTSTICKY 0xFF0000FFL
#define BOTTOMRIGHTSTICKY 0x0000FFFFL
#define NULLSTICKY 0x00000000L
```

Internal XVT-Power++ Commands

```
#define PWRCommandBase 20000
#define NOWINDOWcmd (PWRCommandBase + 1)      // all windows closed
#define NATIVEViewCmd (PWRCommandBase + 2)    // trapped by CScroller
#define NULLcmd (PWRCommandBase + 3)          // default command setting
#define WFSelectCmd (PWRCommandBase + 4)      //wireframe selection
#define WFDeselectCmd (PWRCommandBase + 5)    // wireframe deselection
#define WFSizeCmd (PWRCommandBase + 6)        // wireframe move/size
#define TableEventCmd (PWRCommandBase + 7)    // a CTableEvent command
#define TreeEventCmd (PWRCommandBase + 8)     // a CTableEvent command
#define TDISuspendCmd PWRCommandBase + 9)     // suspend TDI notifications
#define TDIResumeCmd (PWRCommandBase + 10)    // resume TDI notifications
#define TDIFlushCmd (PWRCommandBase + 11)     // flush current state over TDI
```

Internal XVT-Power++ ID Numbers

```
#define PWRIdBase 20000
#define PWREnvBaseId 5000
#define PWRDefTitleSize 100
#define NULLid (PWRIdBase + 1)          //Default ID for objects with no ID
#define TASKDOCid (PWRIdBase + 2)       // ID number of the task document
```

Internal XVT-Power++ Resources

```
#define PWRRidBase 20000
#define NULLwidth 32
#define NULLheight 30
#define NULLicon (PWRRidBase + 1)
```

Min and Max (not defined by some compilers)

```
#ifndef min
#define min(x,y) ((x) < (y) ? (x) : (y))
#endif // min
#ifndef max
define max(x,y) ((x) > (y) ? (x) : (y))
endif // max
```

TDI Macros and Context Commands

```
#define TDI_EVENT( _DoNotify ) \
do { \
        if ( GetPrimaryController() != NULL ) \
            _DoNotify; \
} while( 0 )
#define PWRTdiIDBase 20000
#define PWRTdiIDMax 20999
```

Data change commands

```
#define TDISelectCmd                (PWRTdiIDBase + 1)
#define TDIReplaceCmd               (PWRTdiIDBase + 2)
#define TDIClearCmd                 (PWRTdiIDBase + 3)
#define TDIAppendCmd                (PWRTdiIDBase + 4)
#define TDIRemoveCmd                (PWRTdiIDBase + 5)
#define TDIOptionClearCmd           (PWRTdiIDBase + 6)
#define TDIOptionReplaceCmd         (PWRTdiIDBase + 7)
#define TDIOptionAppendCmd          (PWRTdiIDBase + 8)
#define TDIInsertCmd                (PWRTdiIDBase + 9)
```

Data Request Commands

```
#define TDIIndexCmd                 (PWRTdiIDBase + 100)
#define TDICountCmd                 (PWRTdiIDBase + 101)
#define TDIAllCmd                   (PWRTdiIDBase + 102)
#define TDICurrentCmd               (PWRTdiIDBase + 103)
#define TDINextCmd                  (PWRTdiIDBase + 104)
#define TDIPreviousCmd              (PWRTdiIDBase + 105)
#define TDIFirstCmd                 (PWRTdiIDBase + 106)
#define TDILastCmd                  (PWRTdiIDBase + 107)
#define TDIGotoCmd                  (PWRTdiIDBase + 108)
```

Text Editing

```
#define INSETBORDER 4
#define SZ_MAXTITLE (255*XVT_MAX_MB_SIZE)
```

Units

```
typedef float UNITS;
```

XVT Portability Toolkit Values

This section contains some of the XVT Portability Tookit values, those which XVT-Power++ uses. These selected values are provided here for your convenience.
For a complete listing of the XVT Portability Tookit API, see the *XVT Portability Toolkit Quick Reference Release 5*.

Selected Data Types and Events

ASK_RESPONSE

```
typedef enum e_ask_resp {       /* response from xvt_dm_post_ask fcn */
        RESP_DEFAULT,           /* default button */
        RESP_2,                 /* second button */
        RESP_3                  /* third button */
} ASK_RESPONSE;
```

COLOR

```
typedef unsigned long COLOR;    /* color encapsulation */
```

CURSOR

```
typedef short CURSOR;       /* cursor shape */
```

DIRECTORY

```
typedef struct s_dir {...} DIRECTORY;
```

DRAW_MODE

```
typedef enum e_mode {       /* drawing (transfer) mode */
        M_COPY,             /* patCopy */
        M_OR,               /* patOr */
        M_XOR,              /* patXor */
        M_CLEAR,            /* patBic */
        M_NOT_COPY,         /* notPatCopy */
        M_NOT_OR,           /* notPatOr */
        M_NOT_XOR,          /* notPatXor */
        M_NOT_CLEAR         /* notPatBic */
} DRAW_MODE;
```

EOL_FORMAT

```
typedef enum e_eol {        /* terminator found by xvt_fsys_find_eol function */
        EOL_NORMAL,         /* normal (or first) line terminator */
        EOL_DIFF,           /* terminator different from previous */
        EOL_NONE            /* end of buffer before any terminator */
} EOL_FORMAT;
```

FILE_SPEC

```
typedef struct s_filespec {         /* file specification */
        DIRECTORY dir;              /* directory */
        char type[6];               /* file type or extension */
        char name[SZ_FNAME + 1];    /* file name (may be partial path) */
        char creator[6];            /* file creator */
} FILE_SPEC;
```

FL_STATUS

```
typedef enum e_file {       /* result from file open and save dialogs */
        FL_BAD,             /* error occurred */
        FL_CANCEL,          /* cancel button clicked */
        FL_OK               /* OK button clicked */
} FL_STATUS;
```

PAT_STYLE

```
typedef enum e_pat {
        PAT_NONE,
        PAT_HOLLOW,
        PAT_SOLID,
        PAT_HORZ,
        PAT_VERT,
        PAT_FDIAG,
        PAT_BDIAG,
        PAT_CROSS,
        PAT_DIAGCROSS,
        PAT_RUBBER,
        PAT_SPECIAL
} PAT_STYLE;
```

PEN_STYLE

```
typedef enum e_pen_style {
        P_SOLID,
        P_DOT,
        P_DASH
} PEN_STYLE;
```

WIN_TYPE

```
typedef enum e_win_type {       /* type of window */
        W_NONE,                 /* marker for end of WIN_DEF array */
        W_DOC,                  /* document window */
        W_PLAIN,                /* window with plain border */
        W_DBL,                  /* window with double border */
        W_PRINT,                /* XVT internal use only */
        W_TASK,                 /* task window */
        W_SCREEN,               /* screen window */
        W_NO_BORDER,            /* no border */
        W_PIXMAP,               /* pixmap */
        W_MODAL,                /* modal window */
        WD_MODAL,               /* modal dialog */
        WD_MODELESS,            /* modeless dialog */
        WC_PUSHBUTTON,          /* button control */
        WC_RADIOBUTTON,         /* radio button control */
        WC_CHECKBOX,            /* check box control */
        WC_HSCROLL,             /* horizontal scrollbar control */
        WC_VSCROLL,             /* vertical scrollbar control */
        WC_EDIT,                /* edit control */
        WC_TEXT,                /* static text control */
        WC_LBOX,                /* list box control */
        WC_LISTBUTTON,          /* button with list */
        WC_LISTEDIT,            /* edit field with list */
        WC_GROUPBOX,            /* group box */
        WC_TEXTEDIT,            /* text edit object */
        WC_ICON,                /* icon control */
        WC_NUM_WIN_TYPES        /* number of WIN_TYPEs */
} WIN_TYPE;
```

WINDOW

```
typedef long WINDOW;            /* window descriptor */
```

XVT_CODESET_MAP

```
typedef struct s_codeset_map { ... } *XVT_CODESET_MAP;
```

XVT_DISPLAY_TYPE

```
typedef enum e_display_type {
        XVT_DISPLAY_MONO,           /* monochrome display */
        XVT_DISPLAY_GRAY_16,        /* 16-entry grayscale */
        XVT_DISPLAY_GRAY_256,       /* 256-entry grayscale */
        XVT_DISPLAY_COLOR_16,       /* 16-entry color */
        XVT_DISPLAY_COLOR_256,      /* 256-entry color */
        XVT_DISPLAY_DIRECT_COLOR    /* full color capabilities */
} XVT_DISPLAY_TYPE;
```

***XVT_FNTID**

```
typedef struct {...} * XVT_FNTID;/* font object handle */
```

XVT_FONT_ATTR_MASK

```
typedef unsigned long XVT_FONT_ATTR_MASK;
```

***XVT_FONT_DIALOG**

```
/* Prototype for the application supplied font selection dialog */
typedef BOOLEAN (* XVT_FONT_DIALOG) (
        WINDOW win,                 /* window to send E_FONT event to */
        XVT_FNTID font_id,          /* default font id */
        PRINT_RCD * precp,          /* print record or NULL */
        unsigned long reserved      /* reserved */
 );
```

use prototype:

```
BOOLEAN XVT_CALLCONV1 xvt_font_dialog (WINDOW win, XVT_FNTID font_id,
PRINT_RCD * precp, unsigned long reserved);
```

***XVT_FONT_MAPPER**

```
/* Prototype for the application supplied font mapper */
typedef BOOLEAN (* XVT_FONT_MAPPER) (
XVT_FNTID font_id);
```

use prototype:

```
BOOLEAN XVT_CALLCONV1 font_mapper (XVT_FNTID font_id);
```

XVT_FONT_STYLE_MASK

```
typedef unsigned long XVT_FONT_STYLE_MASK;
```

XVT_FORMAT_HANDLER

```
typedef const char *(*XVT_FORMAT_HANDLER)( WINDOW win,
        char * instr, int * start, int * end, void * data );
```

XVT_HELP_TID

```
typedef long XVT_HELP_TID;
```

XVT_IMAGE_FORMAT

```
typedef enum {
        XVT_IMAGE_NONE,
        XVT_IMAGE_CL8,
        XVT_IMAGE_RGB,
        XVT_IMAGE_MONO
} XVT_IMAGE_FORMAT;
```

***XVT_PALETTE**

```
typedef struct {...} * XVT_PALETTE;
```

***XVT_PALETTE_ATTR**

```
typedef struct {...} * XVT_PALETTE_ATTR;
```

XVT_PALETTE_TYPE

```
typedef enum {
        XVT_PALETTE_NONE,
        XVT_PALETTE_STOCK,
        XVT_PALETTE_CURRENT,
        XVT_PALETTE_CUBE16,
        XVT_PALETTE_CUBE256,
        XVT_PALETTE_USER
} XVT_PALETTE_TYPE;
```

XVT_PATTERN

```
typedef struct s_pat { ... } *XVT_PATTERN;
```

Selected Constants and Portable Attributes

Attribute Definitions for xvt_vobj_get/set_attr

Note that non-portable attributes are defined by the platform header and are documented in the platform-specific books.

Font Attributes

```
#define ATTR_FONT_CACHE_SIZE ...
#define ATTR_FONT_DIALOG ...
#define ATTR_FONT_MAPPER ...
```

Misc Attributes

```
#define ATTR_DEBUG_FILENAME ...
#define ATTR_DEFAULT_PALETTE_TYPE ...
#define ATTR_ERRMSG_FILENAME ...
#define ATTR_ERRMSG_HANDLER ...
#define ATTR_EVENT_HOOK ...
#define ATTR_HELP_CONTEXT ....
#define ATTR_HELP_HOOK ...
#define ATTR_KEY_HOOK ...
#define ATTR_MEMORY_MANAGER ...
#define ATTR_SUPPRESS_UPDATE_CHECK ...
```

Object Size Attributes

```
#define ATTR_CTL_BUTTON_HEIGHT ...
#define ATTR_CTL_CHECKBOX_HEIGHT ...
#define ATTR_CTL_EDIT_TEXT_HEIGHT ...
#define ATTR_CTL_HORZ_SBAR_HEIGHT ...
#define ATTR_CTL_RADIOBUTTON_HEIGHT ...
#define ATTR_CTL_STATIC_TEXT_HEIGHT ...
#define ATTR_CTL_VERT_SBAR_WIDTH ...
```

Predefined Windows

```
#define ATTR_SCREEN_WINDOW ...
#define ATTR_TASK_WINDOW ...
```

System Configuration Attributes

```
#define ATTR_BACK_COLOR ...
#define ATTR_DISPLAY_TYPE ...
#define ATTR_HAVE_COLOR ...
#define ATTR_HAVE_MOUSE ...
#define ATTR_NUM_TIMERS ...
#define ATTR_XVT_CONFIG ...
```

System Metric Attributes

```
#define ATTR_DOC_STAGGER_HORZ ...
#define ATTR_DOC_STAGGER_VERT ...
#define ATTR_PRINTER_HEIGHT ...
#define ATTR_PRINTER_HRES ...
#define ATTR_PRINTER_VRES ...
#define ATTR_PRINTER_WIDTH ...
#define ATTR_SCREEN_HEIGHT ...
#define ATTR_SCREEN_HRES ...
#define ATTR_SCREEN_VRES ...
#define ATTR_SCREEN_WIDTH ...
```

Colors

```
#define COLOR_BLACK ...
#define COLOR_BLUE ...
#define COLOR_CYAN ...
#define COLOR_DKGRAY ...
#define COLOR_GRAY ...
#define COLOR_GREEN ...
#define COLOR_LTGRAY ...
#define COLOR_MAGENTA ...
#define COLOR_RED ...
#define COLOR_WHITE
#define COLOR_YELLOW ...
```

Control, Window, and Dialog Creation Flags

```
#define CTL_FLAG_CENTER_JUST ...      /* centered text */
#define CTL_FLAG_CHECKED ...
#define CTL_FLAG_DEFAULT ...
#define CTL_FLAG_DISABLED ...
#define CTL_FLAG_GROUP ...
#define CTL_FLAG_INVISIBLE ...
#define CTL_FLAG_LEFT_JUST ...        /* left justified text */
#define CTL_FLAG_MAC_GENEVA9 ...      /* opt1 */
#define CTL_FLAG_MAC_MONACO9 ...      /* opt2 */
#define CTL_FLAG_MAC_MULTILINE ...    /* opt3 */
#define CTL_FLAG_MAC_WORDWRAP ... /* opt4 */
#define CTL_FLAG_MULTIPLE ...
#define CTL_FLAG_NATIVE_JUST ....     /* default text justification */
#define CTL_FLAG_PM_SYSICON ...
#define CTL_FLAG_READONLY ...
#define CTL_FLAG_RIGHT_JUST ...       /* right justified text */
#define DLG_FLAG_DISABLED ...
#define DLG_FLAG_INVISIBLE ...
#define WSF_CLOSE ...                 /* is user closeable */
#define WSF_DECORATED ...             /* sizeable, closeable, and has horz. & vert. scrollbar */
#define WSF_DEFER_MODAL ...           /* modal status deferred (not processed by
                                            xvt_win_create*) */
#define WSF_DISABLED ...              /* is initially disabled */
#define WSF_FLOATING ...              /* is floating */
#define WSF_HSCROLL ...               /* has horizontal scrollbar outside client area */
#define WSF_ICONIZABLE ...            /* may iconize window */
#define WSF_ICONIZED ...              /* is initially iconized */
#define WSF_INVISIBLE ...             /* is initially invisible */
#define WSF_MAXIMIZED ...             /* initially maximized */
#define WSF_NO_MENUBAR ...            /* has no menubar of its own */
#define WSF_NONE ...                  /* no flags set */
#define WSF_PLACE_EXACT ...           /* place modal window at specified coordinates */
#define WSF_SIZE ...                  /* is user sizeable */
#define WSF_SIZEONLY ...              /* lacks border rectangles (Mac only) */
#define WSF_VSCROLL ...               /* has vertical scrollbar outside client area */
```

Cursors

```
#define CURSOR_ARROW ...        /* arrow */
#define CURSOR_CROSS ...        /* crosshair */
#define CURSOR_IBEAM ...        /* I-beam */
#define CURSOR_HELP ...         /* help system */
#define CURSOR_PLUS ...         /* plus sign (fatter than crosshair) */
#define CURSOR_USER ...         /* user defined shape (>= 11) */
#define CURSOR_WAIT ...         /* waiting symbol (e.g., hourglass) */
```

Dialog Push Button Control ID Constants

```
#define DLG_CANCEL ...          /* cancel button was clicked */
#define DLG_NO ...              /* other button was clicked */
#define DLG_OK ...              /* default button was clicked */
#define DLG_OUTLINE ...         /* ID of userItem on Mac (internal use) */
#define DLG_YES ...             /* synonym */
```

Event Masks

```
#define EM_ALL ...
#define EM_CHAR ...
#define EM_CLOSE ...
#define EM_COMMAND ...
#define EM_CONTROL ...
#define EM_CREATE ...
#define EM_DESTROY ...
#define EM_FOCUS ...
#define EM_FONT ...
#define EM_HELP ...
#define EM_HSCROLL ...
#define EM_MOUSE_DBL ...
#define EM_MOUSE_DOWN ...
#define EM_MOUSE_MOVE ...
#define EM_MOUSE_UP ...
#define EM_NONE ...
#define EM_QUIT ...
#define EM_SIZE ...
#define EM_TIMER ...
#define EM_UPDATE ...
#define EM_USER ...
#define EM_VSCROLL ...
```

File Attributes

```
#define XVT_FILE_ATTR_ATIME ...
#define XVT_FILE_ATTR_CREATORSTR ...
#define XVT_FILE_ATTR_CTIME ...
#define XVT_FILE_ATTR_DIRECTORY ...
#define XVT_FILE_ATTR_DIRSTR ...
#define XVT_FILE_ATTR_EXECUTE ...
#define XVT_FILE_ATTR_EXIST ...
#define XVT_FILE_ATTR_FILESTR ...
#define XVT_FILE_ATTR_MAXIMUM ...
#define XVT_FILE_ATTR_MINIMUM ...
#define XVT_FILE_ATTR_MTIME ...
#define XVT_FILE_ATTR_NUMLINKS ...
#define XVT_FILE_ATTR_TYPESTR ...
#define XVT_FILE_ATTR_READ ...
#define XVT_FILE_ATTR_SIZE ...
#define XVT_FILE_ATTR_WRITE ...
```

Font Support

(Guaranteed Portable Support)

#define XVT_FFN_COURIER ...
#define XVT_FFN_FIXED ...
#define XVT_FFN_HELVETICA ...
#define XVT_FFN_SYSTEM ...
#define XVT_FFN_TIMES ...

Font Styles

#define XVT_FS_BLINK ...
#define XVT_FS_BOLD ...
#define XVT_FS_INVERSE ...
#define XVT_FS_ITALIC ...
#define XVT_FS_NONE ...
#define XVT_FS_OUTLINE ...
#define XVT_FS_SHADOW ...
#define XVT_FS_STRIKEOUT ...
#define XVT_FS_UNDERLINE ...
#define XVT_FS_USER1 ...
#define XVT_FS_USER2 ...
#define XVT_FS_USER3 ...
#define XVT_FS_USER4 ...
#define XVT_FS_USER5 ...
#define XVT_FS_WILDCARD ...

Key Codes

```
#define K_BTAB ...          /* back tab */
#define K_DEL ...           /* delete (same as ASCII) */
#define K_DOWN ...          /* down arrow */
#define K_CLEAR ...         /* clear */
#define K_COPY ...          /* copy */
#define K_CUT ...           /* cut */
#define K_END ...           /* end */
#define K_F1 ...            /* function key 1 */
#define K_F2 ...
#define K_F3 ...
#define K_F4 ...
#define K_F5 ...
#define K_F6 ...
#define K_F7 ...
#define K_F8 ...
#define K_F9 ...
#define K_F10 ...
#define K_F11 ...
#define K_F12 ...
#define K_F13 ...
#define K_F14 ...
#define K_F15 ...
#define K_F16 ...
#define K_F17 ...
#define K_F18 ...
#define K_F19 ...
#define K_F20 ...
#define K_F21 ...
#define K_F22 ...
#define K_F23 ...
#define K_F24 ...
#define K_HELP ...          /* help */
#define K_HOME ...          /* home */
#define K_INS ...           /* insert */
#define K_KP0 ...           /* keypad '0' */
#define K_KP1 ...
#define K_KP2 ...
#define K_KP3 ...
#define K_KP4 ...
#define K_KP5 ...
#define K_KP6 ...
#define K_KP7 ...
#define K_KP8 ...
#define K_KP9 ...           /* keypad '9' */
#define K_KPADD ...         /* keypad '+' */
#define K_KPDIV ...         /* keypad '/' */
#define K_KPMULT ...        /* keypad '*' */
#define K_KPSUB ...         /* keypad '-' */
#define K_LEFT ...          /* left arrow */
#define K_LEND ...          /* line end */
#define K_LHOME ...         /* line home */
#define K_NEXT ...          /* next screen */
#define K_PASTE ...         /* paste */
#define K_PREV ...          /* previous screen */
#define K_RIGHT ...         /* right arrow */
#define K_WLEFT ...         /* word left */
#define K_WRIGHT ...        /* word right */
#define K_UP ...            /* up arrow */
```

Machine-related Constants

```
#define CHAR_MAX ...
#define INT_MAX ...
#define LONG_MAX ...
#define SHRT_MAX ...
#define UCHAR_MAX ...
#define UINT_MAX ...
#define ULONG_MAX ...
#define USHRT_MAX ...
```

Predefined Windows (value is WINDOW)

```
#define SCREEN_WIN ...
#define TASK_WIN ...
```

Resource ID Constants

```
#define DB_ABORT ...
#define DB_ABOUT ...
#define DB_ASK ...
#define DB_ERROR ...
#define DB_FONTSEL ...
#define DB_HELPTEXT ...
#define DB_HELPTOPICS ...
#define DB_NOTE ...
#define DB_OPEN ...
#define DB_RESPONSE ...
#define DB_SAVE ...
#define DB_WARNING ...
#define ICON_RSRC ...
#define MENU_BAR_RID ...          /* ID for default menubar resource */
#define STR_HELPTYPE ...          /* string resource for help file type */
#define XVT_STRING_RES_BASE ...
```

Text Edit Attributes

```
#define ATTR_R40_TXEDIT_BEHAVIOR ...   /* revert to R4.0 text edit behavior */
#define TX_AUTOHSCROLL ...             /* autoscroll horizontally */
#define TX_AUTOVSCROLL ...             /* autoscroll vertically */
#define TX_BORDER ...                  /* rectangular border */
#define TX_DISABLED ...                /* disabled state */
#define TX_ENABLECLEAR ...             /* leave CLEAR enabled always */
#define TX_INVISIBLE ...               /* visibility state */
#define TX_ONEPAR ...                  /* one paragraph only (no \r) */
#define TX_OVERTYPE ...                /* overtype mode */
#define TX_NOCOPY ...                  /* no copy allowed */
#define TX_NOCUT ...                   /* no cut allowed */
#define TX_NOMENU ...                  /* no edit menu changes */
#define TX_NOPASTE ...                 /* no paste allowed */
#define TX_READONLY ..                 /* text is not editable */
#define TX_WRAP ...                    /* wrap text to margin */
```

Tool Brush Constants (xvt_dwin_set_std_cbrush flags)

```
#define TL_BRUSH_BLACK ...
#define TL_BRUSH_WHITE ...
```

Tool Pen Constants (xvt_dwin_set_std_cpen flags)

```
#define TL_PEN_BLACK ...
#define TL_PEN_HOLLOW ...
#define TL_PEN_RUBBER ...
#define TL_PEN_WHITE ...
```