DEFINE DROPDOWN MENU

Creates a dropdown menu definition

Standard Syntax (xBase Style):

```
DEFINE DROPDOWN MENU BUTTON <ToolBarDropDownButtonName> [ OF <ParentWindowName> ]
   MENUITEM <cItemCaption>
   ACTION <ActionProvedureName> | <bBlock>
   [ NAME <MenuItemName>]
   [ IMAGE <cImageName> ]
   [ CHECKED ]
   [ NOTRANSPARENT ]
   [ TOOLTIP <cToolTipText> ]
   ...
   ...
      [ SEPARATOR ]
    ...
    ...
END MENU
```

Alternate Syntax:

```
DEFINE DROPDOWNMENU OWNERBUTTON <ParentToolBarButtonName> [ PARENT <ParentWindowName>
]
   MENUITEM <cItemCaption>
   ONCLICK <ActionProvedureName>
   [ NAME <MenuItemName>]
   [ IMAGE <cImageName> ]
   [ CHECKED <lValue> ]
   [ TRANSPARENT <lValue> ]
   [ TOOLTIP <cToolTipText> ]
   ...
   ...
   [ SEPARATOR ]
   ...
   ...
END MENU
```

DropDown Menu Properties:

- Parent (R)
- OwnerButton

R: Read-Only

MenuItem Properties:

- Name (R)

- Checked
- Enabled
- Image
- ToolTip

R: Read-Only

MenuItem Events:

- OnClick

You can **DEFINE\RELEASE** Menu at runtime:

```
RELEASE DROPDOWN MENU BUTTON ButtonName OF FormName
RELEASE DROPDOWNMENU OWNERBUTTON ButtonName OF FormName
ReleaseDropDownMenu ( cButtonName, cFormName )
IsDropDownMenuDefined ( cButtonName, cFormName ) --> lBoolean
```

@...EDITBOX \/ DEFINE EDITBOX

Creates an EditBox control

Standard Syntax (xBase Style):

```
@ <nRow> ,<nCol>
EDITBOX<ControlName>
[ OF | PARENT <ParentWindowName> ]
WIDTH <nWidth>
HEIGHT <nHeight>
[ FIELD <FieldName> ]
[ VALUE <cValue> ]
[ READONLY ]
[ FONT <cFontName> SIZE <nFontSize> ]
[ BOLD ] [ ITALIC ] [ UNDERLINE ] [ STRIKEOUT ]
[ TOOLTIP <cToolTipText> ]
[ BACKCOLOR <aColor> ]
[ FONTCOLOR <aColor> ]
[ DISABLEDBACKCOLOR <aColor> ]
[ DISABLEDFONTCOLOR <aColor> ]
[ MAXLENGTH <nInputLength> ]
[ ON GOTFOCUS <OnGotFocusProcedur> | <bBlock> ]
[ ON CHANGE <OnChangeProcedure> | <bBlock> ]
[ ON LOSTFOCUS <OnLostFocusProcedure> | <bBlock> ]
[ HELPID <nHelpId> ]
[ BREAK ]
[ INVISIBLE ]
[ NOTABSTOP ]
[ NOVSCROLL ]
[ NOHSCROLL ]
```

Alternate Syntax:

```
DEFINE EDITBOX <ControlName>
   PARENT <ParentWindowName>
   ROW <nValue>
   COL <nValue>
   WIDTH <nValue>
   HEIGHT <nValue>
   FONTNAME <cValue>
   FONTSIZE <nValue>
   FONTBOLD <lValue>
   FONTITALIC <lValue>
   FONTUNDERLINE <lValue>
   FONTSTRIKEOUT <lValue>
   TOOLTIP <cValue>
   ONGOTFOCUS <ActionProcedure>
   ONLOSTFOCUS <ActionProcedure>
   ONCHANGE <ActionProcedure>
   TABSTOP <lValue>
   HELPID <nValue>
   VISIBLE <lValue>
   FIELD <FieldName>
   VALUE <cValue>
   READONLY <lValue>
   BREAK <lValue>
   VSCROLLBAR <lValue>
   HSCROLLBAR <lValue>
   BACKCOLOR <aColor>
   FONTCOLOR <aColor>
   DISABLEDBACKCOLOR <aColor>
   DISABLEDFONTCOLOR <aColor>
END EDITBOX
```

Properties:

- Value
- Enabled
- Visible
- Row
- Col
- Width
- Height
- FontName
- FontSize
- FontBold
- FontItalic
- FontUnderline
- FontStrikeout
- ToolTip

- BackColor
- FontColor
- DisabledBackColor
- DisabledFontColor
- CaretPos
- Name (R)
- Field (D)
- Parent (D)
- ReadOnly
- MaxLength (D)
- HelpId (D)
- Break (D)
- TabStop (D)
- VScrollBar (D)
- HScrollBar (D)
- GetTextLength

D: Available at control definition only R: Read-Only

Events:

- OnGotFocus
- OnChange
- OnLostFocus

Methods:

- Show
- Hide
- SetFocus
- Release
- Refresh
- Save

@...FRAME \/ DEFINE FRAME

Creates a frame control

Standard Syntax (xBase Style):

```
@ <nRow> ,<nCol>
FRAME<ControlName>
[ OF | PARENT <ParentWindowName> ]
[ CAPTION <cCaption> ]
WIDTH <nWidth>
HEIGHT <nHeight>
[ FONT <cFontName> ]
[ SIZE <nFontSize> ]
[ BOLD ]
[ ITALIC ]
[ UNDERLINE ]
[ STRIKEOUT ]
[ BACKCOLOR <aBackColor> ]
[ FONTCOLOR <aFontColor> ]
[ TRANSPARENT ]
```

Alternate Syntax:

```
DEFINE FRAME <ControlName>
   PARENT <ParentWindowName>
   ROW <nValue>
   COL <nValue>
   WIDTH <nValue>
   HEIGHT <nValue>
   FONTNAME <cValue>
   FONTSIZE <nValue>
   FONTBOLD <lValue>
   FONTITALIC <lValue>
   FONTUNDERLINE <lValue>
   FONTSTRIKEOUT <lValue>
   CAPTION <cCaption>
   BACKCOLOR <aBackColor>
   FONTCOLOR <aFontColor>
   TRANSPARENT <lValue>
END FRAME
```

Properties:

- Enabled
- Visible
- Row
- Col
- Width

- Height
- Caption
- FontName
- FontSize
- FontBold
- FontItalic
- FontUnderline
- FontStrikeout
- BackColor
- FontColor
- Name (R)
- Parent (D)
- Transparent (D)

D: Available at control definition only R: Read-Only

Methods:

- Show
- Hide
- Release

Note: FontColor property works only for 'Windows Clssic' \/ 'Windows Standard' themes.

@...GRID \/ DEFINE GRID

Creates a grid control

Standard Syntax (xBase Style):

```
 @ <nRow> ,<nCol>
 GRID <ControlName>
 [ OF | PARENT <ParentWindowName> ]
 WIDTH <nWidth>
 HEIGHT <nHeight>
 HEADERS <acHeaders>
 WIDTHS <anWidths>
 [ ITEMS <acItems> ]
 [ VALUE <nValue> ]
 [ FONT <cFontname> SIZE <nFontsize> ]
 [ BOLD ] [ ITALIC ] [ UNDERLINE ] [ STRIKEOUT ]
 [ TOOLTIP <cToolTipText> ]
 [ BACKCOLOR <aBackColor> ]
 [ FONTCOLOR <aFontColor> ]
 [ DYNAMICBACKCOLOR <aDynamicBackColor> ]
 [ DYNAMICFORECOLOR <aDynamicBackColor> ]
 [ ON GOTFOCUS <OnGotFocusProcedur> | <bBlock> ]
 [ ON CHANGE <OnChangeProcedure> | <bBlock> ]
 [ ON LOSTFOCUS <OnGotFocusProcedure> | <bBlock> ]
 [ [ ON DBLCLICK <OnDblClickProcedure> | <bBlock> ] |
 [ EDIT | ALLOWEDIT ] ]
 [ ON SAVE <OnSaveProcedure> | <bBlock> ]  [ COLUMNCONTROLS {aControlDef1,aControlDef2
,...aControlDefN}
 [ COLUMNVALID {bValid1,bValid2,...bValidN}
 [ COLUMNWHEN {bWhen1,bWhen2,...bWhenN}
 [ ON HEADCLICK <abBlock> ]
 [ VIRTUAL ]
 [ ITEMCOUNT <nItemCount> ]
 [ ON QUERYDATA <OnQueryDataProcedure> | <bBlock> ]
 [ MULTISELECT ]
 [ NOLINES ]
 [ NOHEADERS ]
 [ IMAGE <acImageNames> ]
 [ JUSTIFY <anJustifyValue> ]
 [ HELPID <nHelpId> ]
 [ BREAK ]
 [ HEADERIMAGES <acHeaderImages> ]
 [ CELLNAVIGATION ]
 [ ROWSOURCE <cRowSource>]
 [ COLUMNFIELDS <acColumnFields> ]
 [ ALLOWAPPEND ]
 [ ALLOWDELETE ]
 [ DYNAMICDISPLAY <abDynamicDisplay> ]
 [ LOCKCOLUMNS <nValue> ]
 [ ON CLICK <OnClickProcedure> ]
 [ ON KEY <OnKeyProcedure> ]
 [ ON CHECKBOXCLICKED <OnCheckBoxClickedProcedure> ]
 [ NOTRANSPARENT ]
 [ NOTRANSPARENTHEADER ]
 [ ON INPLACEEDITEVENT <OnInplaceEditEventProcedure> ]
 [ EDITOPTION <nEditOption> ]
```

Note: nEditOption --> GRID_EDIT_DEFAULT | GRID_EDIT_SELECTALL | GRID_EDIT_INSERTBLANK | GRID_EDIT_INSERTCHAR | GRID_EDIT_REPLACEALL

Alternate Syntax:

```
DEFINE GRID <ControlName>
   PARENT <ParentWindowName>
   ROW <nValue>
   COL <nValue>
   WIDTH <nValue>
   HEIGHT <nValue>
   FONTNAME <cValue>
   FONTSIZE <nValue>
   FONTBOLD <lValue>
   FONTITALIC <lValue>
   FONTUNDERLINE <lValue>
   FONTSTRIKEOUT <lValue>
   TOOLTIP <cValue>
   ONGOTFOCUS <ActionProcedure>
   ONLOSTFOCUS <ActionProcedure>
   ONSAVE <ActionProcedure> ONCHANGE <ActionProcedure>
   TABSTOP <lValue>
   HELPID <nValue>
   VISIBLE <lValue>
   ITEMS <caItems>
   HEADERS <acHeaders>
   WIDTHS <anWidths>
   VALUE <nValue>
   BACKCOLOR <aBackColor>
   FONTCOLOR <aFontColor>
   DYNAMICBACKCOLOR <aDynamicBackColor>
   DYNAMICFORECOLOR <aDynamicBackColor>
   ONDBLCLICK <OnDblClickProcedure>
   ALLOWEDIT <lValue>
   COLUMNCONTROLS {aControlDef1,aControlDef2,...aControlDefN}
   COLUMNVALID {bValid1,bValid2,...bValidN}
   COLUMNWHEN {bWhen1,bWhen2,...bWhenN}
   ONHEADCLICK <abBlock>
   VIRTUAL <lValue>
   ITEMCOUNT <nItemCount>
   ONQUERYDATA <OnQueryDataProcedure>
   MULTISELECT <lValue>
   LINES <lValue>
   SHOWHEADERS <lValue>
   IMAGE <acImageNames>
   JUSTIFY <anJustifyValue>
   HEADERIMAGES <acHeaderImages>
   CELLNAVIGATION <lValue>
   ROWSOURCE <cRowSource>]
   COLUMNFIELDS <acColumnFields>
   ALLOWAPPEND <lValue>
   ALLOWDELETE <lValue>
```

```
    DYNAMICDISPLAY <abDynamicDisplay>
    LOCKCOLUMNS <nValue>
    ONCLICK <OnClickProcedure>
    ONKEY <OnKeyProcedure>
    ONCHECKBOXCLICKED <OnCheckBoxClickedProcedure>
    ONINPLACEEDITEVENT <OnInplaceEditEventProcedure>
    EDITOPTION <nEditOption>
    TRANSPARENT <lValue>
    TRANSPARENTHEADER <lValue>
END GRID
```

Properties:

- RowSource
- ColumnFields
- AllowAppend
- AllowDelete
- DynamicDisplay
- RecNo
- ColumnWhen
- DynamicBackColor
- DynamicForeColor
- Cell (nRow , nCol)
- Value
- Enabled
- Visible
- Item (nItemIndex)
- ItemCount
- Row
- Col
- Width
- Height
- FontName
- FontSize
- FontBold
- FontItalic
- FontUnderline
- FontStrikeout
- ToolTip
- BackColor
- FontColor
- Header (nColumnNumber)
- HeaderImages (nColumnNumber)

- Name (R)
- Virtual (D)
- Parent (D)
- Widths
- MultiSelect (D)
- NoLines (D)
- Image
- Justify
- HelpId (D)
- Break (D)
- AllowEdit (D)
- ColumnControls
- ColumnValid
- Headers
- CellNavigation (D)
- Items (D)
- LockColumns (D)

D: Available at control definition only R: Read-Only

Properties Available For OnQueryData Procedure:

```
This.QueryData
This.QueryRowIndex
This.QueryColIndex
```

Properties Available For OnDblClick Procedure:

```
This.CellRowIndex
This.CellColIndex
This.CellRow
This.CellCol
This.CellWidth
This.CellHeight
```

- Note: These properties are not available when **OnDblClick** procedure is fired by <Enter> key pressing.

Properties Available For DynamicBackColor \/ DynamicForeColor \/ ColumnValid Processing:

```
This.CellRowIndex
This.CellColIndex
This.CellValue
```

Properties Available For OnSave Procedure:

```
This.AppendBuffer
This.EditBuffer
This.MarkBuffer
```

- This.EditBuffer: Array of one element per edited cell.

The elements has the following structure: { nLogicalRow , nLogicalCol , xValue , nRecNo }

- This.AppendBuffer: Array of one element per appended record. The elements has the following structure: { xFieldValue 1 , ... , xFieldValue n }
- This.MarkBuffer: Array of one element per record marked to be deleted or recalled. The elements has the following structure: { nLogicalRow , nRecNo , cMark ('D' or 'R') }

Properties Available For OnInplaceEditEvent Procedure:

```
This.IsInplaceEditEventInit --> .T. or .F.
This.IsInplaceEditEventRun --> .T. or .F.
This.IsInplaceEditEventFinish --> .T. or .F.
This.InplaceEditGridName --> eg. “Grid_1”
This.InplaceEditParentName --> eg. “Form_1”
This.InplaceEditControlHandle --> Handle of InplaceEdit ColumnControl, eg. Handle of T
EXTBOX, DATEPICKER, TIMEPICKER, COMBOBOX, SPINNER, CHECKBOX, etc.
This.InplaceEditControlIndex --> Return nControlIndex
```

Events***:***

- OnGotFocus
- OnChange
- OnLostFocus
- OnDblClick
- OnHeadClick
- OnQueryData
- OnSave
- OnClick
- OnKey
- OnCheckBoxClicked
- OnInplaceEditEvent

Methods:

- Show
- Append
- Save
- Refresh([lValue])
- Delete
- Recall
- Hide
- AddItem (acItemText)
- ClearBuffer
- DeleteItem (nItemIndex)
- DeleteAllItems
- SetFocus
- DisableUpdate
- EnableUpdate
- Release
- AddColumn ([nColIndex] , [cCaption] , [nWidth] , [nJustify])
- DeleteColumn (nColIndex)

- Hints:

- The leftmost column in a grid control must be left aligned.
- When used in control definition, Header property must be loaded with a character array containing as elements as control columns.
- When AddColumn \/ DeleteColumn methods are used, all items in grid (if any) will be lost.
- If MULTISELECT clause is used VALUE must be a numeric array, containing the index position of selected items.
- If EDIT clause is used, by doubleclicking an item, will open an editing window allowing to change the item content.
- EDIT and MULTISELECT clauses can't be used simultaneously.
- When RowSource is specified, the workarea specified, must be open at control definition.
- Hotkey when RowSource is specified:

  ALT + A --> Append

  ALT + D --> Delete

ALT + R --> ReCall

ALT + S --> Save (On Save)

ALT + U --> ClearBuffer

Control Definition Array:

TEXTBOX

```
{ cControlType , cDataType , cInputMask , cFormat }
cControlType = 'TEXTBOX' (Required)
cDataType = 'CHARACTER' , 'NUMERIC' , 'DATE' (Required)
cInputMask = cInputMask (Optional)
cFormat = cFormat (Optional)
```

DATEPICKER

```
{ cControlType , cControlStyle }
cControlType = 'DATEPICKER' (Required)
cControlStyle = 'DROPDOWN' , 'UPDOWN' (Required)
```

COMBOBOX

```
{ cControlType , acItems , axReturnValues }
cControlType 'COMBOBOX' (Required)
acItems (Required)
axReturnValues (Optional)
```

SPINNER

```
{ cControlType , nRangeMin , nRangeMax }
cControlType 'SPINNER' (Required)
nRangeMin (Required)
nRangeMax (Required)
```

CHECKBOX

```
{ cControlType , cCheckedLabel , cUnCheckedLabel }
cControlType 'CHECKBOX' (Required)
cCheckedLabel (Required)
cUnCheckedLabel (Required)
```

Note: Data type for each column will depend control specified.

```
NUMERIC TEXTBOX : NUMERIC
DATE TEXTBOX : DATE
CHARACTER TEXTBOX : CHARACTER
SPINNER : NUMERIC
COMBOBOX : NUMERIC (Any type if axReturnValues is specified)
CHECKBOX : LOGICAL
```

Sample:

```
@ 10,10 GRID Grid_1 ;
WIDTH 620 ;
HEIGHT 330 ;
HEADERS {'Column 1','Column 2','Column 3','Column 4',;
'Column 5'} ;
WIDTHS {140,140,140,140,140} ;
ITEMS aRows ;
EDIT ;
COLUMNCONTROLS { {'TEXTBOX','NUMERIC','$ 999,999.99'},;
{'DATEPICKER','DROPDOWN'} ,;
{'COMBOBOX',{'One','Two','Three'}} , ;
{ 'SPINNER' , 1 , 20 } , ;
{ 'CHECKBOX' , 'Yes' , 'No' } }
```

Justify constants:

```
GRID_JTFY_LEFT
GRID_JTFY_RIGHT
GRID_JTFY_CENTER
```

GRID Control improvement

New Features:

Grid Grouping:

```
<ParentWindowName>.<GridControlName>.GroupEnabled [ := | -->] lBoolean
Note: Grid Group is not available when application is running on Windows versions of 3
2-bits
You can check if your application is running on Win32 with the function HMG_IsRunAppIn
Win32()
<ParentWindowName>.<GridControlName>.GroupDeleteAll
<ParentWindowName>.<GridControlName>.GroupDelete ( nGroupID )
<ParentWindowName>.<GridControlName>.GroupDeleteAllItems ( nGroupID )
<ParentWindowName>.<GridControlName>.GroupExist ( nGroupID ) --> lBoolean
<ParentWindowName>.<GridControlName>.GroupCheckBoxAllItems ( nGroupID ) := lBoolean
<ParentWindowName>.<GridControlName>.GroupGetAllItemIndex ( nGroupID ) --> anItemIndex
<ParentWindowName>.<GridControlName>.GroupExpand ( nGroupID )
<ParentWindowName>.<GridControlName>.GroupCollapsed ( nGroupID )
<ParentWindowName>.<GridControlName>.GroupAdd ( nGroupID [, nPosition ] )
<ParentWindowName>.<GridControlName>.GroupInfo ( nGroupID ) [ := | -->] { [ cHeader ]
, [ nAlignHeader ] , [ cFooter ] , [ nAlingFooter ] , [ nState ] }
<ParentWindowName>.<GridControlName>.GroupItemID ( nItem ) [ := | -->] nGroupID

Note: nAlignHeader [ := | -->] GRID_GROUP_LEFT | GRID_GROUP_CENTER | GRID_GROUP_RIGHT
nAlingFooter [ := | -->] GRID_GROUP_LEFT | GRID_GROUP_CENTER | GRID_GROUP_RIGHT
nState [ := | -->] GRID_GROUP_NORMAL | GRID_GROUP_COLLAPSED
```

Gird CheckBox:

```
<ParentWindowName>.<GridControlName>.CheckBoxEnabled [ := | -->] lBoolean
<ParentWindowName>.<GridControlName>.CheckBoxItem ( nRow ) [ := | -->] lBoolean
<ParentWindowName>.<GridControlName>.CheckBoxAllItems := lBoolean
```

Grid Header Fonts, Images etc.,

```
<ParentWindowName>.<GridControlName>.PaintDoubleBuffer [ := | -->] lBoolean // Paints
via double-buffering, which reduces flicker
<ParentWindowName>.<GridControlName>.HeaderDYNAMICFONT ( nCol ) := {|| {cFontName, nFo
ntSize, [ lBold, lItalic, lUnderline, lStrikeOut ]} }
<ParentWindowName>.<GridControlName>.HeaderDYNAMICFORECOLOR ( nCol ) := {|| aColor }
<ParentWindowName>.<GridControlName>.HeaderDYNAMICBACKCOLOR ( nCol ) := {|| aColor }
<ParentWindowName>.<GridControlName>.Image ( lTransparent ) := { "image1.png", "image2
.bmp", ... }
<ParentWindowName>.<GridControlName>.ImageIndex ( nRow , nCol ) [ := | -->] nIndex
<ParentWindowName>.<GridControlName>.ImageList [ := | -->] hImageList
<ParentWindowName>.<GridControlName>.ColumnDYNAMICFONT ( nCol ) := {|| {cFontName, nFo
ntSize, [ lBold, lItalic, lUnderline, lStrikeOut ]} }
<ParentWindowName>.<GridControlName>.HeaderImageIndex ( nCol ) [ := | -->] nIndex
<ParentWindowName>.<GridControlName>.ChangeFontSize := nSize | NIL // Useful for use D
ynamic Font with more (less) Height than the size of font the Grid control
```

Dynamic Font

```
ARRAY FONT <cFontName> SIZE <nFontSize> [ BOLD ] [ ITALIC ] [ UNDERLINE ] [ STRIKEOUT
] --> { cFontName, nFontSize, lBold, lItalic, lUnderline, lStrikeout }
CREATE ARRAY FONT <cFontName> SIZE <nFontSize> [ BOLD <lBold> ] [ ITALIC <lIitalic> ]
[ UNDERLINE <lUnderline> ] [ STRIKEOUT <lStrikeout> ] --> { cFontName, nFontSize, lBol
d, lItalic, lUnderline, lStrikeout }
```

Additional Get Properties:

```
<ParentWindowName>.<GridControlName>.ColumnCOUNT --> nColumnCount
<ParentWindowName>.<GridControlName>.ColumnHEADER ( nColIndex ) --> cColumnHeader
<ParentWindowName>.<GridControlName>.ColumnWIDTH ( nColIndex ) --> nColumnWidth
<ParentWindowName>.<GridControlName>.ColumnJUSTIFY ( nColIndex ) --> nColumnJustify
<ParentWindowName>.<GridControlName>.ColumnCONTROL ( nColIndex ) --> aColumnControl
<ParentWindowName>.<GridControlName>.ColumnDYNAMICBACKCOLOR ( nColIndex ) --> bColumnD
ynamicBackColor
<ParentWindowName>.<GridControlName>.ColumnDYNAMICFORECOLOR ( nColIndex ) --> bColumnD
ynamicForeColor
<ParentWindowName>.<GridControlName>.ColumnVALID ( nColIndex ) --> bColumnValid
<ParentWindowName>.<GridControlName>.ColumnWHEN ( nColIndex ) --> bColumnWhen
<ParentWindowName>.<GridControlName>.ColumnONHEADCLICK ( nColIndex ) --> bColumnOnHead
Click
<ParentWindowName>.<GridControlName>.ColumnDISPLAYPOSITION ( nColIndex ) --> nColumnDi
splayPosition
<ParentWindowName>.<GridControlName>.CellEx ( nRowIndex, nColIndex ) --> xValue (very
more fast that .Cell)
<ParentWindowName>.<GridControlName>.CellRowFocused --> nCellRowIndex
<ParentWindowName>.<GridControlName>.CellColFocused --> nCellColIndex
<ParentWindowName>.<GridControlName>.CellRowClicked --> nCellRowIndex
<ParentWindowName>.<GridControlName>.CellColClicked --> nCellColIndex
<ParentWindowName>.<GridControlName>.CellNavigation --> lBoolean
<ParentWindowName>.<GridControlName>.EditOption --> GRID_EDIT_DEFAULT | GRID_EDIT_SELE
CTALL |
GRID_EDIT_INSERTBLANK | GRID_EDIT_INSERTCHAR |
GRID_EDIT_REPLACEALL
```

Additional Set Properties:

```
<ParentWindowName>.<GridControlName>.ColumnHEADER ( nColIndex ) := cColumnHeader
<ParentWindowName>.<GridControlName>.ColumnWIDTH ( nColIndex ) := [ nColumnWidth ] |
[ GRID_WIDTH_AUTOSIZE ] |
[ GRID_WIDTH_AUTOSIZEHEADER ]
<ParentWindowName>.<GridControlName>.ColumnJUSTIFY ( nColIndex ) := nColumnJustify
<ParentWindowName>.<GridControlName>.ColumnCONTROL ( nColIndex ) := aColumnControl
<ParentWindowName>.<GridControlName>.ColumnDYNAMICBACKCOLOR ( nColIndex ) := bColumnDy
namicBackColor
<ParentWindowName>.<GridControlName>.ColumnDYNAMICFORECOLOR ( nColIndex ) := bColumnDy
namicForeColor
<ParentWindowName>.<GridControlName>.ColumnVALID ( nColIndex ) := bColumnValid
<ParentWindowName>.<GridControlName>.ColumnWHEN ( nColIndex ) := bColumnWhen
<ParentWindowName>.<GridControlName>.ColumnONHEADCLICK ( nColIndex ) := bColumnOnHeadC
lick
<ParentWindowName>.<GridControlName>.ColumnDISPLAYPOSITION ( nColIndex ) := nColumnDis
playPosition
<ParentWindowName>.<GridControlName>.CellEx ( nRowIndex, nColIndex ) := xValue (very m
ore fast that .Cell)
<ParentWindowName>.<GridControlName>.BackGroundImage ( **nAction, cPicture, nRow, nCol
 )**
nAction = GRID_SETBKIMAGE_NONE | GRID_SETBKIMAGE_NORMAL | GRID_SETBKIMAGE_TILE | GRID_
SETBKIMAGE_WATERMARK
<ParentWindowName>.<GridControlName>.CellNavigation := lBoolean
<ParentWindowName>.<GridControlName>.EditOption := GRID_EDIT_DEFAULT | GRID_EDIT_SELEC
TALL | GRID_EDIT_INSERTBLANK | GRID_EDIT_INSERTCHAR | GRID_EDIT_REPLACEALL
```

New Methods:

```
<ParentWindowName>.<GridControlName>.AddColumnEx ( [ nColIndex ],[ cCaption ],[ nWidth
 ],[ nJustify ],[aColumnControl] )
<ParentWindowName>.<GridControlName>.AddItemEx ( aItem, nRow )
```

Set\/Get Grid New Properties\/Methods with equivalent syntax:

- **THIS.**XXX --> where XXX is the name of the new Grid Properties\/Methods
- AddColumn, AddColumnEx and DeleteColumn properties now NOT clean the Grid (NOT Delete all items),

  for compatibility with old behavior of ADDCOLUMN and DELETECOLUMN:

- **SET GridDeleteAllItems [TRUE|ON] | [FALSE|OFF]**
- **IsGridDeleteAllItems()** --> Return .T. or .F.
- Set colors and display mode in GRID cell navigation mode:

```
CellNavigationColor ( _SELECTEDCELL_FORECOLOR, aRGBcolor )
CellNavigationColor ( _SELECTEDCELL_BACKCOLOR, aRGBcolor )
CellNavigationColor ( _SELECTEDCELL_DISPLAYCOLOR, lBoolean )
CellNavigationColor ( _SELECTEDROW_FORECOLOR, aRGBcolor )
CellNavigationColor ( _SELECTEDROW_BACKCOLOR, aRGBcolor )
CellNavigationColor ( _SELECTEDROW_DISPLAYCOLOR, lBoolean )
```

- Get colors in GRID cell navigation mode:

```
aRGBcolor := CellNavigationColor ( _SELECTEDCELL_FORECOLOR )
aRGBcolor := CellNavigationColor ( _SELECTEDCELL_BACKCOLOR )
aRGBcolor := CellNavigationColor ( _SELECTEDROW_FORECOLOR )
aRGBcolor := CellNavigationColor ( _SELECTEDROW_BACKCOLOR )
```

@...HYPERLINK \/ DEFINE HYPERLINK

Creates an hyperlink control

Standard Syntax (xBase Style):

```
 @ <nRow>,<nCol> HYPERLINK <ControlName>
 [ OF <ParentWindowName> ]
 [ VALUE <cControlValue> ]
 [ ADDRESS <cLinkAddress>]
 [ WIDTH <nWidth> ]
 [ HEIGHT <nHeight> ]
 [ AUTOSIZE ]
 [ FONT <cFontName> ]
 [ SIZE <nFontSize> ]
 [ BOLD ]
 [ ITALIC ]
 [ TOOLTIP <cToolTipText> ]
 [ BACKCOLOR <anBackColor> ]
 [ FONTCOLOR <anFontColor> ]
 [ RIGHTALIGN ]
 [ CENTERALIGN ]
 [ HELPID <nHelpId> ]
 [ HANDCURSOR ]
 [ INVISIBLE ]
```

Alternate Syntax:

```
DEFINE HYPERLINK <ControlName>
   PARENT <ParentWindowName>
   ROW <nValue>
   COL <nValue>
   WIDTH <nValue>
   HEIGHT <nValue>
   FONTNAME <cValue>
   FONTSIZE <nValue>
   FONTBOLD <lValue>
   FONTITALIC <lValue>
   BACKCOLOR <anBackColor>
   FONTCOLOR <anFontColor>
   HELPID <nValue>
   VISIBLE <lValue>
   VALUE <cControlValue>
   ADDRESS <cLinkAddress>
   HANDCURSOR <lValue>
   AUTOSIZE <lValue>
   ALIGNMENT Left | Right | Center
   TOOLTIP <cToolTipText>
END HYPERLINK
```

Properties:

- Value
- Enabled
- Visible
- Row
- Col
- Width
- Height
- FontName
- FontSize
- FontBold
- FontItalic
- FontUnderline
- FontStrikeout
- AutoSize
- Address
- Name (R)
- Parent (D)
- BackColor
- FontColor
- HelpId (D)
- Alignment (D)
- Handcursor (D)

D: Available at control definition only R: Read-Only

Methods:

- Show
- Hide
- Release