@...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 ]
            [ EDITOPTION <nEditOption> ]

              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>
        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 (D)
- MultiSelect (D)
- NoLines (D)
- Image (D)
- Justify (D)
- HelpId (D)
- Break (D)

- AllowEdit (D)
- ColumnControls (D)
- ColumnValid (D)
- Headers (D)
- CellNavigation (D)
- Items (D)
- Image (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') }

Events:

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

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.

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)

      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:**

- <ParentWindowName>.<GridControlName>.**PaintDoubleBuffer** [:= | -->] lBoolean
// Paints via double-buffering, which reduces flicker

- <ParentWindowName>.<GridControlName>.**GroupEnabled** [:= | -->] lBoolean
  Note: **Grid Group** is not available when application is running on Windows versions of 32-bits
  You can check if your application is running on Win32 with the function **HMG_IsRunAppInWin32()**
- <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**

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

- <ParentWindowName>.<GridControlName>.**HeaderDYNAMICFONT** (nCol) := {|| {cFontName, nFontSize, [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, nFontSize, [lBold, lItalic, lUnderline, lStrikeOut]} }
- <ParentWindowName>.<GridControlName>.**HeaderImageIndex** (nCol) [:= | -->] nIndex
- <ParentWindowName>.<GridControlName>.**ChangeFontSize** := nSize | NIL // Useful for use Dynamic 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, lBold, lItalic, lUnderline, lStrikeout }

- **New 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 )  --> bColumnDynamicBackColor
            <ParentWindowName>.<GridControlName>.ColumnDYNAMICFORECOLOR (
nColIndex )  --> bColumnDynamicForeColor
            <ParentWindowName>.<GridControlName>.ColumnVALID            (
nColIndex )  --> bColumnValid
            <ParentWindowName>.<GridControlName>.ColumnWHEN             (
nColIndex )  --> bColumnWhen
            <ParentWindowName>.<GridControlName>.ColumnONHEADCLICK      (
nColIndex )  --> bColumnOnHeadClick
            <ParentWindowName>.<GridControlName>.ColumnDISPLAYPOSITION  (
nColIndex )  --> nColumnDisplayPosition
            <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_SELECTALL |

GRID_EDIT_INSERTBLANK | GRID_EDIT_INSERTCHAR |

GRID_EDIT_REPLACEALL
```

- **New 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 )  := bColumnDynamicBackColor
            <ParentWindowName>.<GridControlName>.ColumnDYNAMICFORECOLOR (
nColIndex )  := bColumnDynamicForeColor
            <ParentWindowName>.<GridControlName>.ColumnVALID            (
nColIndex )  := bColumnValid
```

<ParentWindowName>.<GridControlName>.**ColumnWHEN (nColIndex)** := bColumnWhen
<ParentWindowName>.<GridControlName>.**ColumnONHEADCLICK (nColIndex)** := bColumnOnHeadClick
<ParentWindowName>.<GridControlName>.**ColumnDISPLAYPOSITION (nColIndex)** := nColumnDisplayPosition
<ParentWindowName>.<GridControlName>.**CellEx (nRowIndex, nColIndex)** := xValue (very more 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_SELECTALL |**

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 )
```