IWGrid Help

1.03.0

IWGrid Help

Printed: February 2015 in USA

Publisher

T. L. McCaughn

Technical Editors

T. L. McCaughn

Special thanks to:

All the people who continue to use and enjoy IWBASIC and support Ionic Wind Software.

Table of Contents

Introduction

Part I

1 Introduction

IWGrid is a general purpose custom grid control for use with IWBASIC© / EBASIC© . It allows the User to easily create one or more grids, each with the desired number of cells. The User can then configure each cell individually as to the type of data to be displayed and/or edited, and along with numerous other parameters.

IWGrid consists of a static link library and an include file to be added in the User's source file. Also included with the distribution is this help file and coding examples.

IWGrid General Features:

- Unlimited number of grids per applications.
- Up to 512 columns per grid (including optional row numbers)
- Unlimited number of rows per grid
- Optional Grid Title
- Optional Column Headers
- Optional Row Numbers
- Configure individual cells as one of 15 types
- Optional editable cells
- Optional individual or global cell protection.
- Changeable grid line color.

Grid Title

- Adjustable font and text size
- Optional text shadows
- User definable number of text rows with optional automatic height adjustment
- User definable, background, foreground and shadow colors.

Grid Column Headers

- Adjustable font and text size
- Optional text shadows
- Optional automatic column titles or User defined text
- User definable number of text rows with optional automatic height adjustment
- User definable, background, foreground and shadow colors.
- Fixed or User sizeable column widths when viewing

Grid Cells

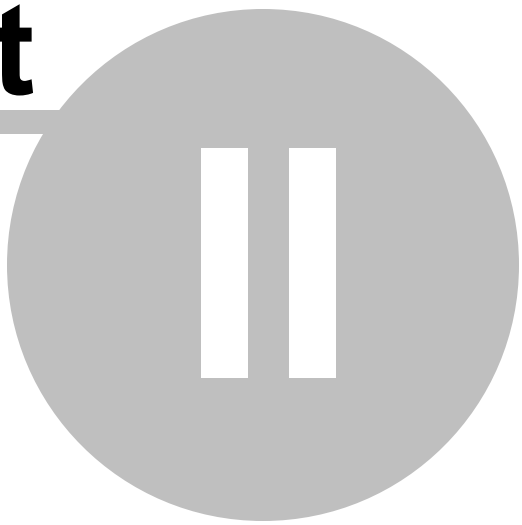
- Global font and text size for all cells
- Optional adjustable or fixed width and height
- Individually configured cells based on cell type
- Background and foreground colors selectable for each cell
- Individual cells are editable or optionally read-only

Minimum Requirements:

- IWBasic© v. 1.737 or above (Copyright 2010-, Ionic Wind Software)
- 128MB of ram.
- 3.3MB Free hard drive space.
- Tested with Windows XP, Windows Vista, and Windows 7.
- Requires that windowssdk.inc be installed on the User's computer in order to compile.

OverView

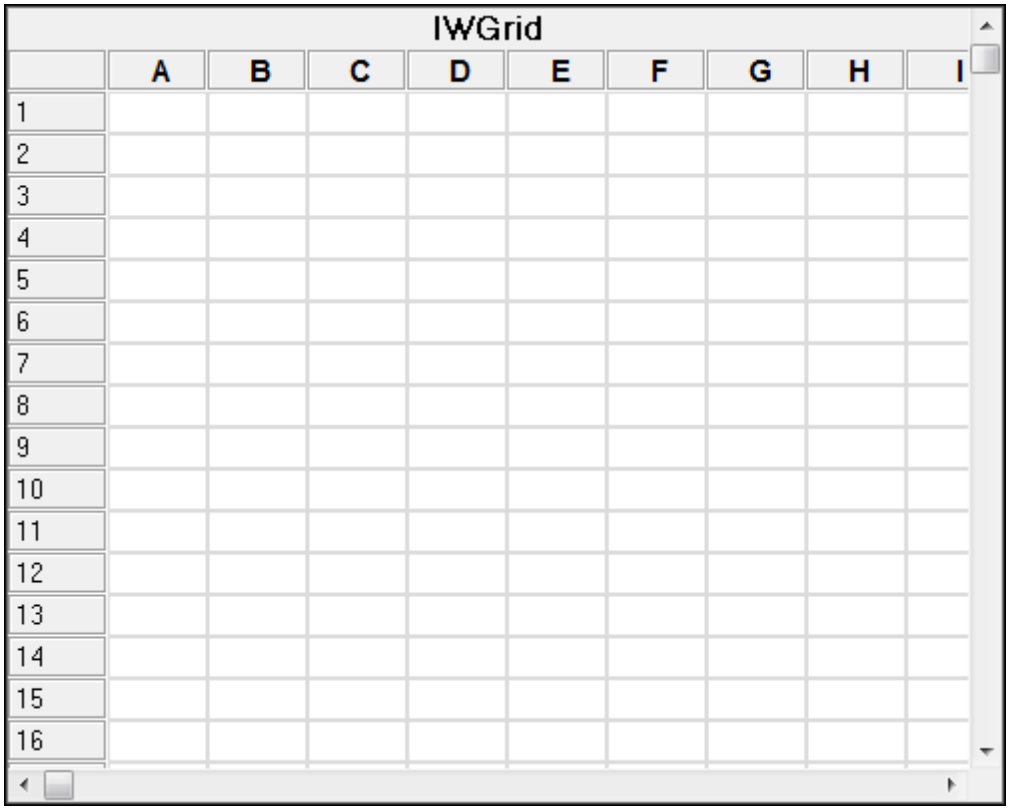
Part



2 OverView

IWGrid is designed to make it easy for the User to display and, optionally edit, data. The following image depicts the basic grid structure.

Basic Grid



With few exceptions the configuration of the grid can be divided into four basic sections:

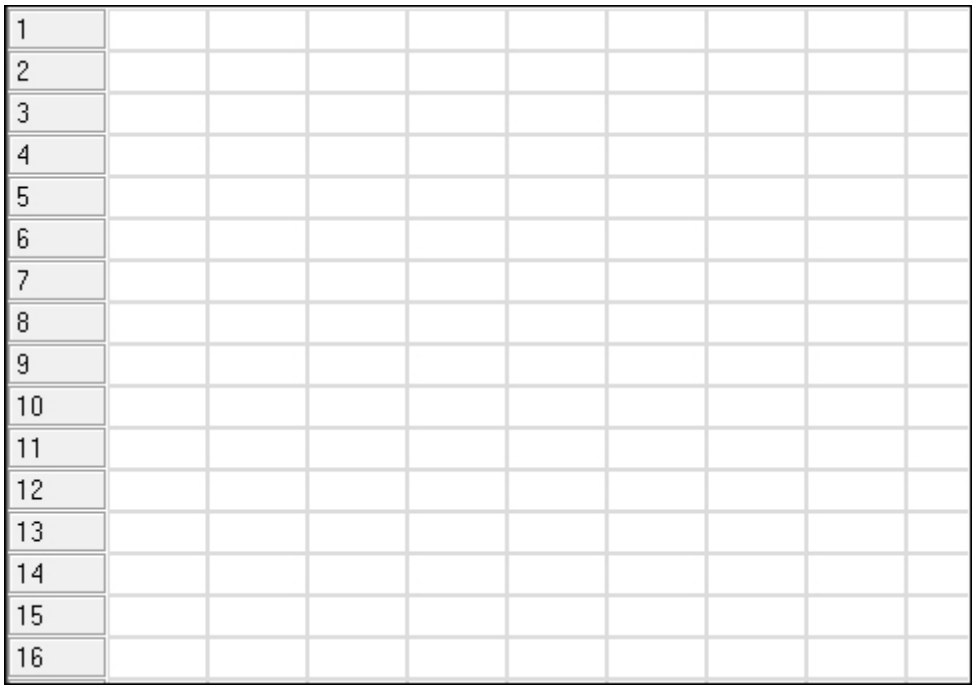
Grid Header

This represents the optional title of the grid. *

Grid Column Header

This represents the optional column labels/titles. *

Grid Body



This represents all the cells contained in the grid *

Grid Cell

Each cell in the Grid Body has certain configurable parameters specific to the cell alone.*

Note: * See the corresponding section in the Getting Started section for additional information.

Getting Started

Part II

3 Getting Started

The process of creating an IWGrid control can be broken down.into five steps Each of these steps are discussed in the following subsections:

- Basic Grid
- Grid Header
- Grid Column Header
- Grid Body
- Grid Cells

Following these five steps, in sequence, will insure that the User's grid associated code remains organized and easy to modify.

3.1 Basic Grid

There are only two aspects to creating an IWGrid.
First, the following line has to be added to any and every source file that will contain code for creating a grid.:

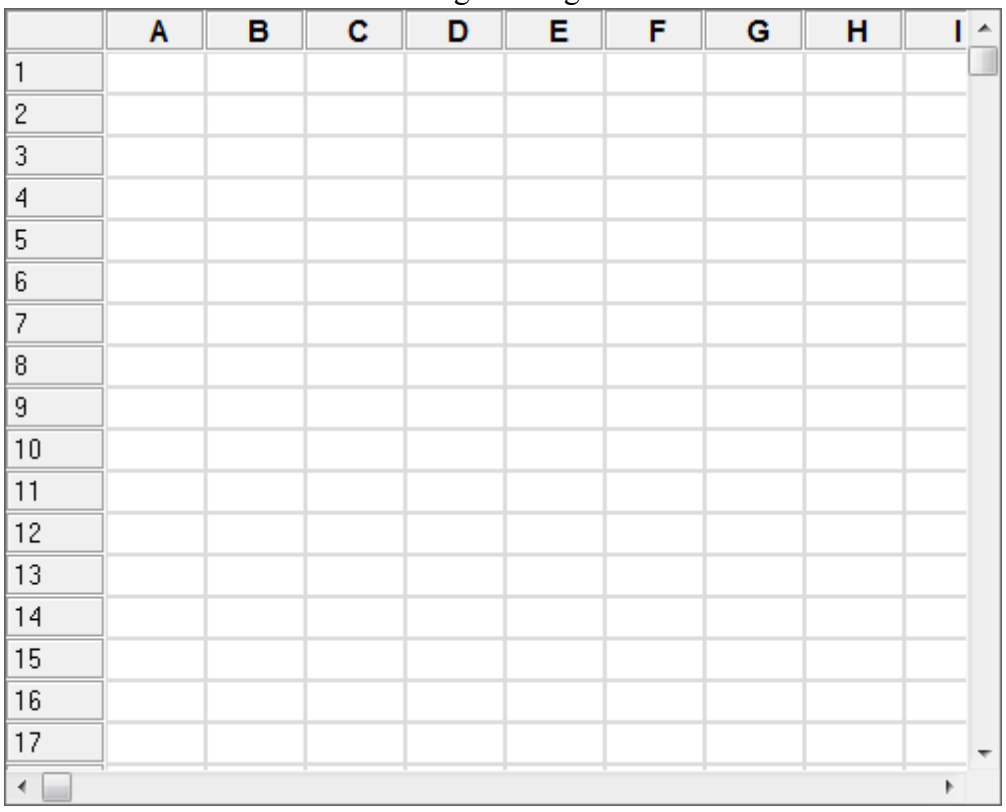
```
$INCLUDE "IWGrid.inc"
```

It is assumed that the User has installed the IWGrid.lib file in the *IWBDev/libs* folder and the IWGrid.inc file has been installed in the *IWBDev/include* folder.

The basic grid is created with the IWG_Create command. Typical examples of code are shown below:

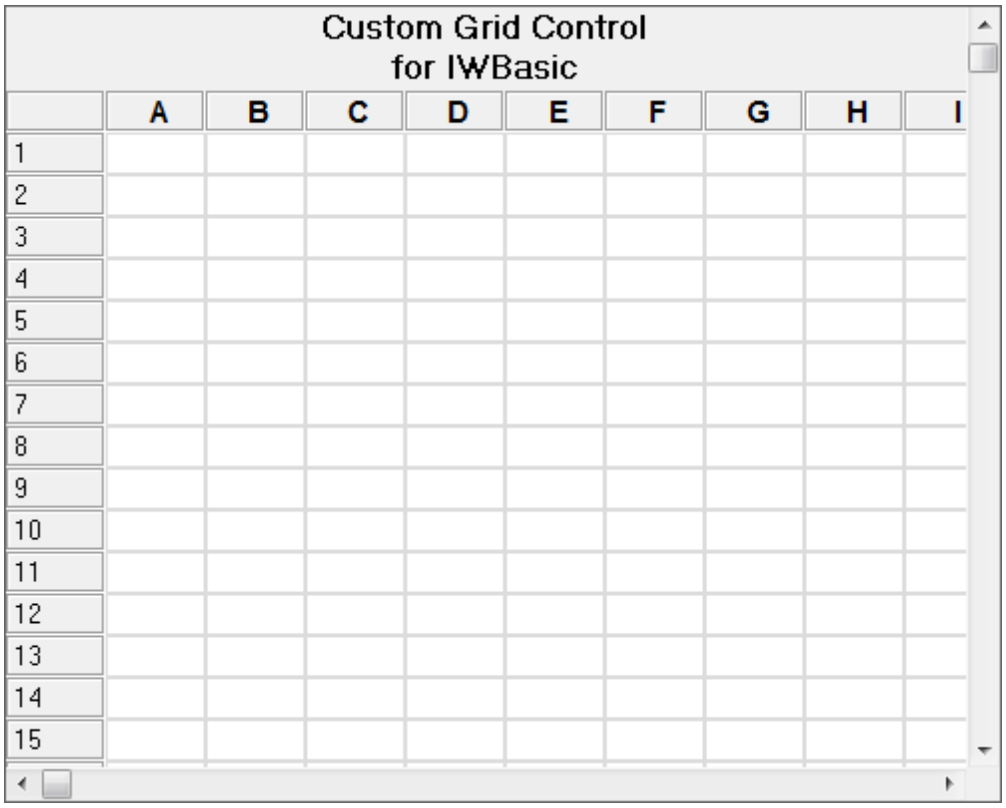
```
IWG_Create(win, "", 20,30,500,400, ID_GRID1)
```

The above line will create the following default grid:




```
IWG_Create(main, "Custom Grid Control\nfor IWBasic", 20,30,500,400, 501)
```

The above line will create the same size grid but with a title, as shown below.



Notice the multi-line title caused by the insertion of *\n* at the desired location. There is no limit to the number of lines the title may contain. If no title is specified when the grid is created it may be added later if desired. See the Getting Started > Grid Header for additional details.

The default grid has 100 rows and 255 columns.

3.2 Grid Title Header

The IWGrid grid title header, or title, is optional. The following sub-sections describe all the grid header options available to the User.

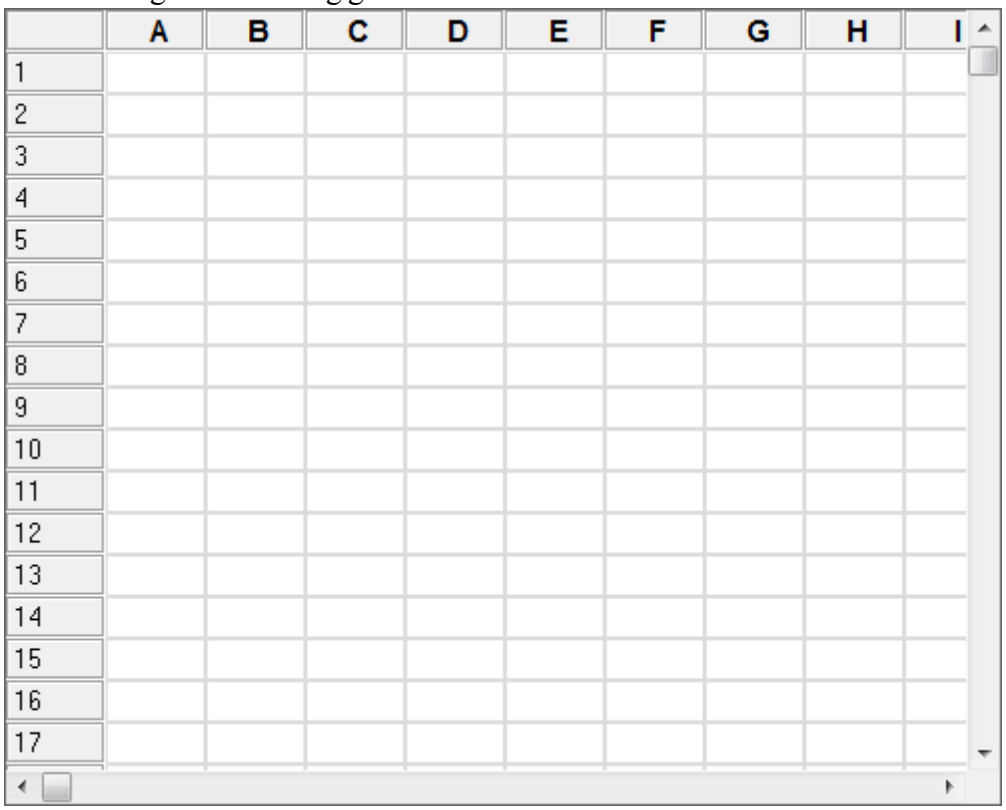
- General
- Color
- Font
- Shadow

3.2.1 General

In its simplest form the grid does not have a title. This occurs when a grid is created (with the IWG_Create command) with a NULL string for a title, as follows:

```
IWG_Create(win, "", 20,30,500,400, ID_GRID1)
```

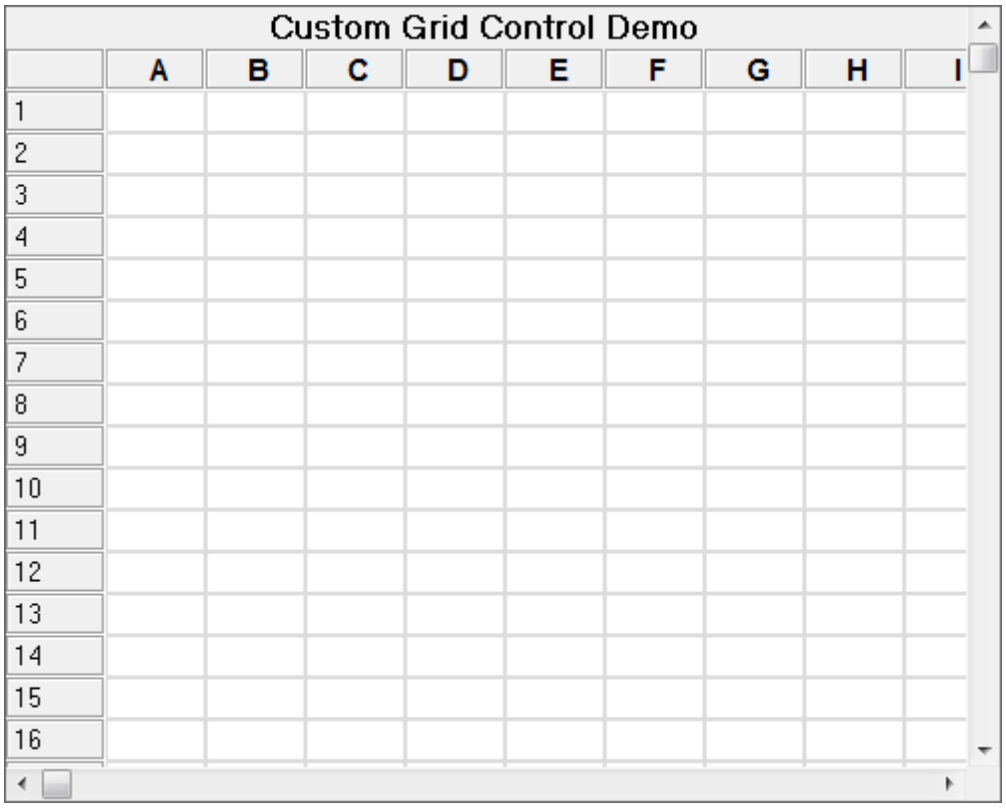
The following is the resulting grid:



By replacing the NULL string with text and using the same command line:

```
IWG_Create(win, "Custom Grid Control Demo", 20,30,500,400, ID_GRID1)
```

the following grid, with a title header, is created.:



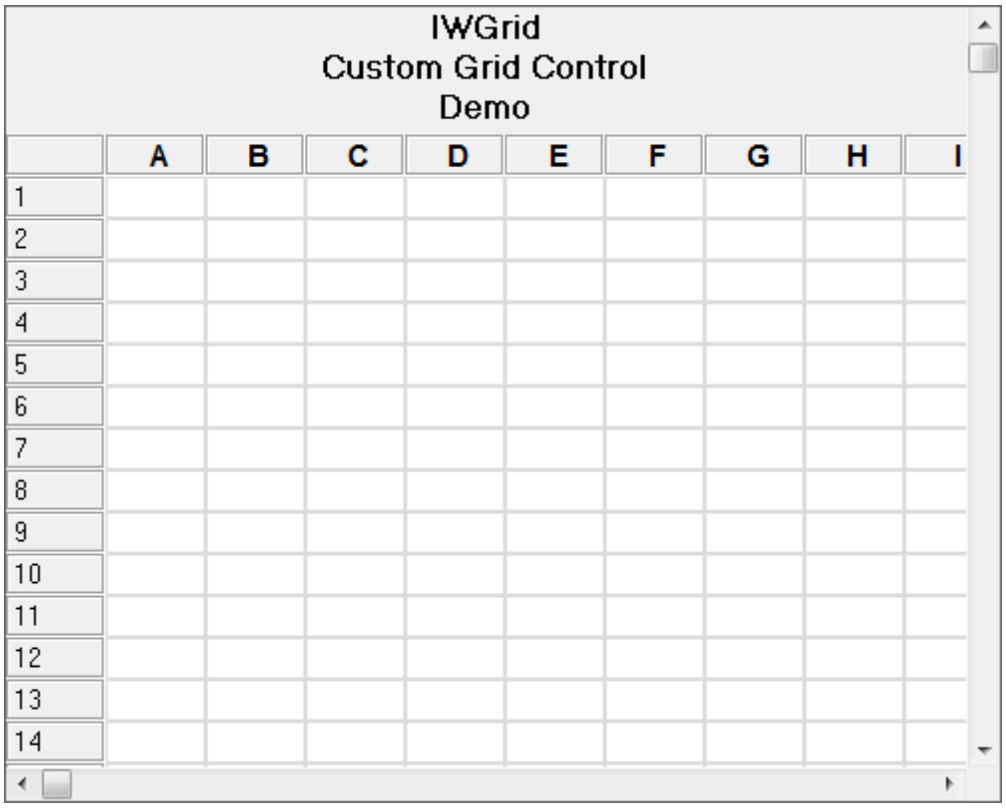
The same exact result will occur if the title is added after the grid is created. This is accomplished using the following two lines of code:

```
IWG_Create(win, "", 20,30,500,400, ID_GRID1)
IWG_SetTitle(win, ID_GRID1, "Custom Grid Control Demo")
```

By adding the newline (\n) character to the title string a title with multiple lines can be created. And it doesn't matter which command is used to accomplish it (IWG_Create or IWG_SetTitle). The following command creates a title with 3 lines:

```
IWG_SetTitle(win, ID_GRID1, "IWGrid\nCustom Grid Control\nDemo")
```

The above results in the following:



There are several things to be aware of concerning the grid header:

- The height of the header is automatically calculated when the title text or font is changed.
- The size of the header subtracts from the space available for displaying grid rows.
- There is no preset limit to the number of lines the title may contain.
- The lines in the title will be centered horizontally within the header space.

The header can be removed at any time with the following command:

```
IWG_SetTitle(win, ID_GRID1, "")
```

3.2.2 Font

When a title is added to a grid with either the IWG_Create or IWG_SetTitle command the current "title" font is used. The default title font is Tahoma with a height of 12 and a weight of 400. The following shows a gride header with a typical title using the default font:

The default font can be changed using the IWG_SetFont command with the @IWGTITLEFONT

flag. The IWG_SetFont command uses the same basic parameters used by IWBasic's SetFont command (typeface, height, weight, and flags). A typical command would be:

```
IWG_SetFont(win, ID_GRID1, @IWGTITLEFONT, "Courier", 20, 800, @SFITALIC)
```

which results in the following:

Notice the height of the header itself (compared to the previous example of the default font). Any time a command is executed that changes the text or font of the title the required height of the header is recalculated.

3.2.3 Shadow

The grid title may have an optional shadow applied to the title text. As an example, shown below is a title with the default font and with no shadow(default):

Using the IWG_ShowTitleShadow command a shadow can be applied to the text:

```
IWG_ShowTitleShadow(win, ID_GRID1,1)
```

which results in the following very subtle change:

The shadow appears to be of little or no value if used with the default colors. The next section will show that the shadow can, indeed, have a dramatic effect.

The IWG_SetShadowOffset command can be used to used to move the shadow down and to the right with positive values and up and to the left with negative values. The default value is 1 pixel.

3.2.4 Color

The following is a typical grid title header using the default colors (and the title shadow turned off - default):

The IWG_SetGridColor command with the @IWGTITLECOLOR flag can be used to change the text color and background color. The following command:

```
IWG_SetGridColor(win, ID_GRID1, @IWGTITLECOLOR, RGB(255,0,0), RGB(255,255,0))
```

results in the title header shown below.

Turning on the title shadow (with the default shadow color) results using

```
IWG_ShowTitleShadow(win, ID_GRID1,1)
```

results in:

The title shadow color can be changed with the IWG_SetGridColor command with the @IWGTITLESHADOWCLR flag. The following command

```
IWG_SetGridColor(win, ID_GRID1, @IWGTITLESHADOWCLR, RGB(0,255,0))
```

gives the following title header.

Another example would be

```
IWG_ShowTitleShadow(win, ID_GRID1,1)
IWG_SetGridColor(win, ID_GRID1, @IWGTITLECOLOR, RGB(0,255,0), RGB(0,0,255))
IWG_SetGridColor(win, ID_GRID1, @IWGTITLESHADOWCLR, RGB(0,255,255))
```

gives the following:

NOTE: The User can use the *Color Picker* tool from IWBasic's IDE *Tools Menu* to facilitate the choice of colors.

3.3 Grid Column Header

The IWGrid grid column header is optional. The following sub-sections describe all the grid column header options and commmands available to the User.
General
Font
Shadow
Color

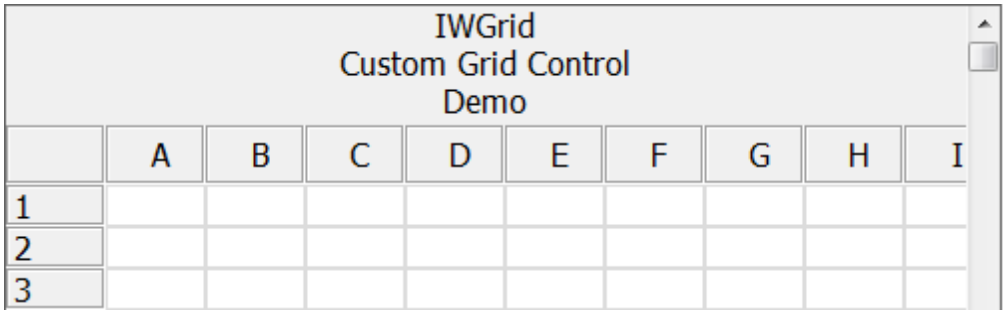
3.3.1 General

Default

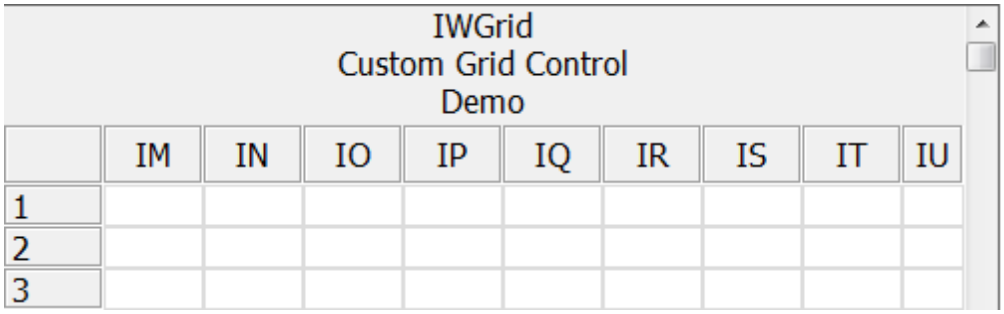
The basic, default grid created with

```
IWG_Create(main, "Custom Grid Control\nfor IWBasic", 20,30,500,400, 501)
```

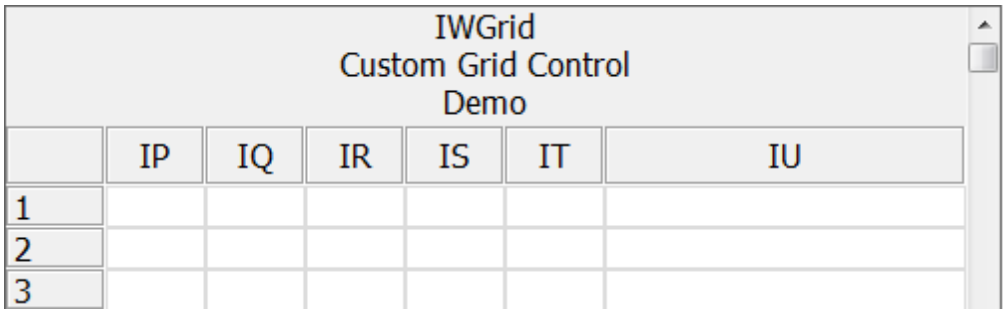
contains a column header row that is auto-numbered, as shown below.



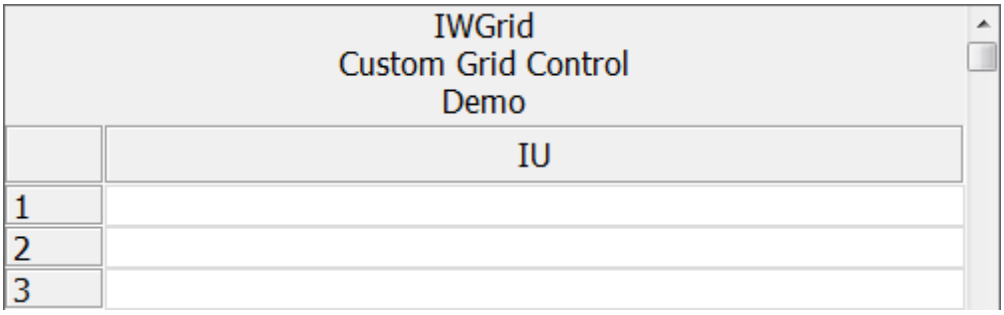
Since the default grid is created with 255 columns, scrolling to the right shows the last column is numbered "IU" (below).



By scrolling a little further the following shows that the last column expands to the right side of the grid



When scrolling is at its maximum extent the following shows that the last column has expanded to completely fill the grid.



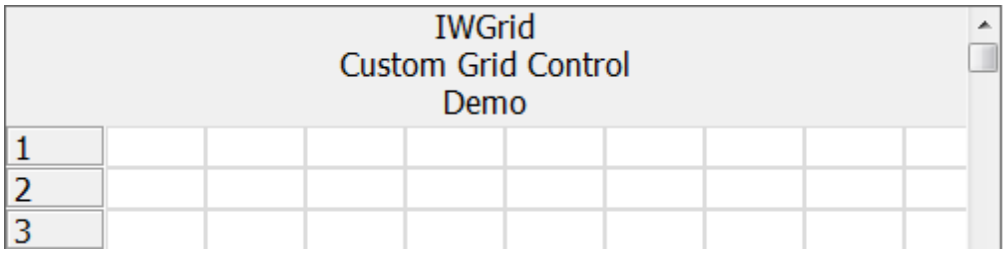
As mentioned previously the default number of columns is 255. The number of columns can be set to a maximum of 512 (includes the row number column) with the IWG_SetDim command. The following command will set the number of rows to 10 and the number of columns to 5.

```
IWG_SetDim(main, 501, 10, 5)
```

If a column header row is not desired it may be removed with the IWG_ShowHeader command.

```
IWG_ShowHeader(main,  501, 0)
```

which results in the following grid:



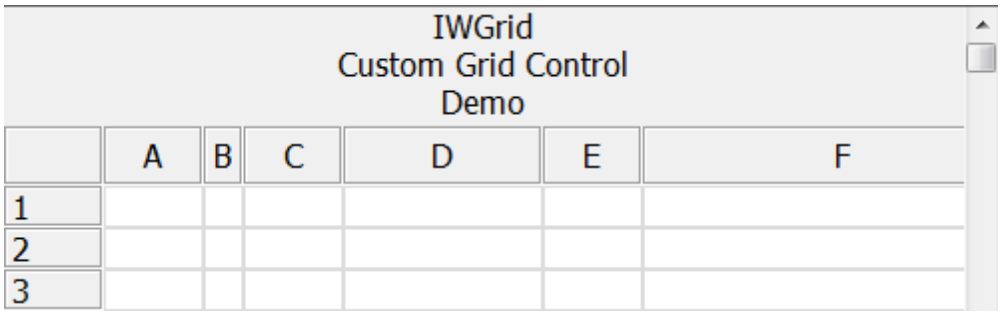
Column Widths

The default width of each column is 50 pixels. A columns width can be changed with the IWG_SetColWidth command

The following commands

```
IWG_SetColWidth(main,501, 2, 20)
IWG_SetColWidth(main,501, 4, 100)
IWG_SetColWidth(main,501, 6, 200)
```

results in the the following;



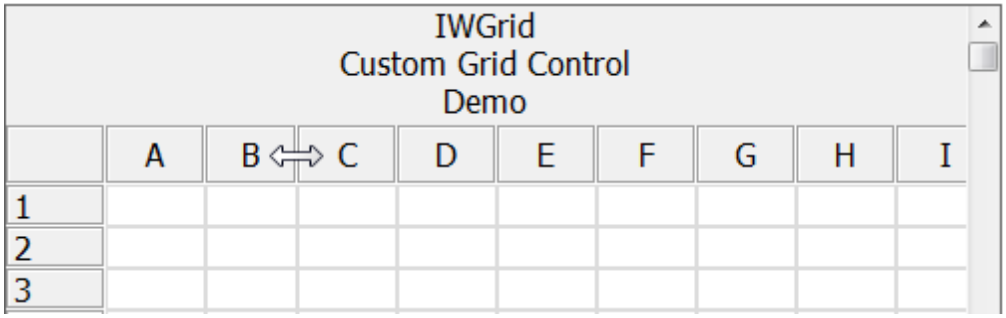
Column Width Resizing

By default, what ever the column widths are set to the End User is not able to alter. However, there are times when it may be advantageous to allow the End User to change the width to temporarily alter the size of the columns.

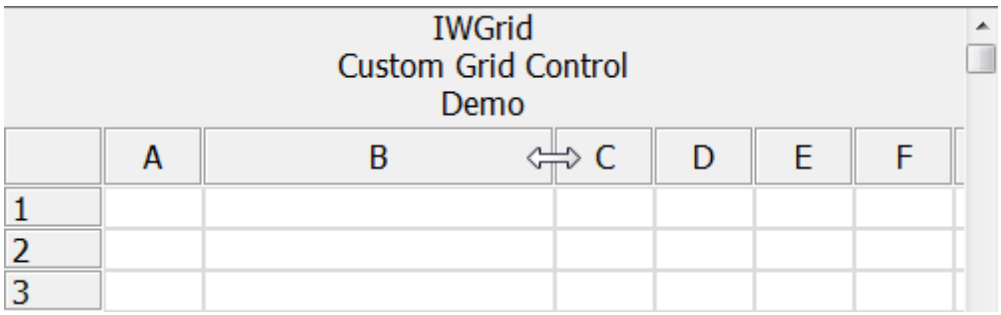
Column resizing can be enabled with the IWG_ColResizing command, as follows:

```
IWG_ColResizing(win, ID_GRID1, 1)
```

When column resizing is enabled, moving the mouse pointer over a column separator in the column header will result in the mouse pointer changing to a resizing cursor, as shown below.



By holding the right mouse button down while the resizing cursor is showing the End User may drag the column divider to increase/decrease the size of the column to the left of the resizing cursor (shown below)



Once the column is the desired width the End User releases the mouse button and the column will remain that size until the End User drags it again or the program is restarted. The application can also be programmed so as to reset the column widths under the User's defined circumstances with the IWG_SetColWidth command.

Column Header Height

The default column header height is 21 pixels. This can be changed with the IWG_SetHeaderHeight command. An example is shown below.

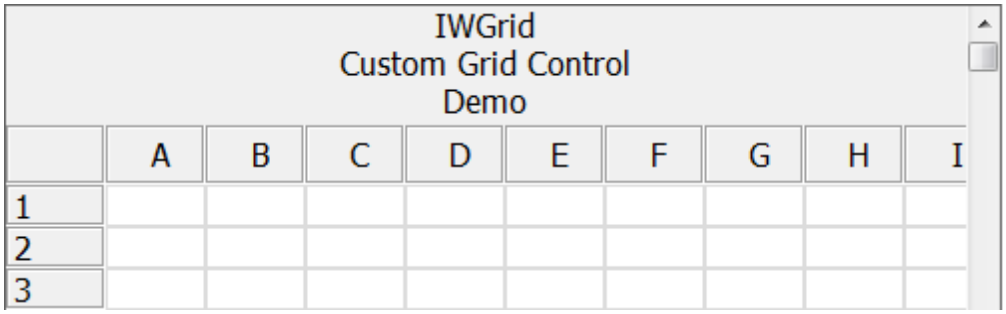
```
IWG_SetHeaderHeight(main, 501,50)
```

Also the User has the option to allow the height of the column header row to be calculated automatically, based upon the contents of the column titles, provided the IWG_AutoNumberHeader command has been previously used to turn off the header title auto-number feature. This means that this feature can be enabled with the IWG_SetHeaderAutoHeight command, as shown below, only when User defined column header titles are being used.

```
IWG_SetHeaderAutoHeight(main, 501,1)
```

Column Titles

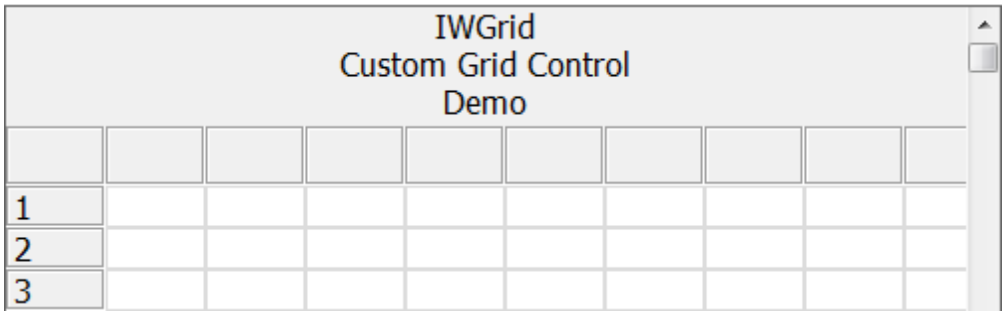
All the previous examples in this section have shown the default column header titles, like below.



If the User desires to maintain the column header row but remove the auto-generated titles, it can be accomplished with the IWG_AutoNumberHeader command shown below.

```
IWG_AutoNumberHeader(win, ID_GRID1, 0)
```

which results in

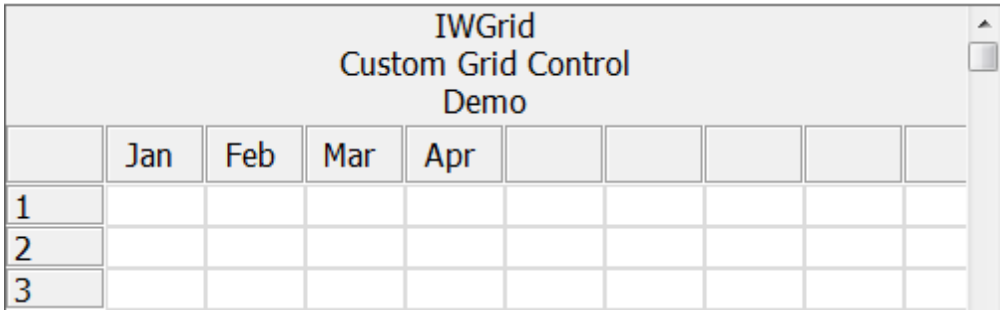


A more common situation is where the column titles are unique and specific to the application. This can be accomplished with a combination of the IWG_AutoNumberHeader and IWG_InitCellDat commands.

The following lines of code

```
IWG_AutoNumberHeader(win, ID_GRID1, 0)
IWG_InitCellDat(win, ID_GRID1, 0, 1, "Jan",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 2, "Feb",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 3, "Mar",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 4, "Apr",@IWGLABEL,@IWG_CENTERALIGN)
```

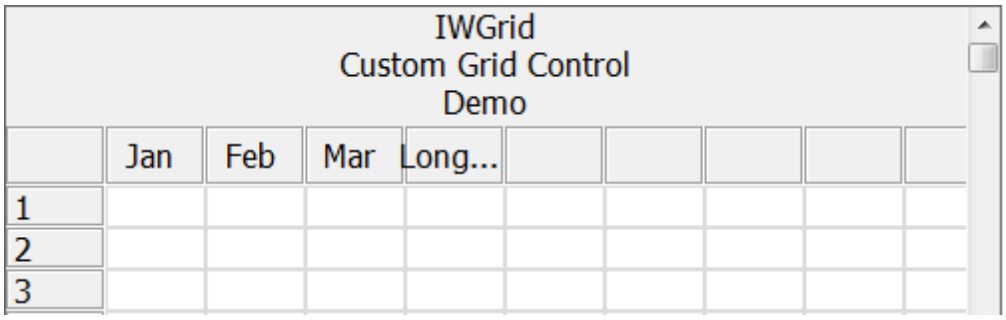
give the following grid



Once the user define column is initialized with the IWG_InitCellDat command, the column title can be changed with the IWG_SetCellDat command.. Using the previous example the title in the fourth column can be changed using the the following command line

```
IWG_SetCellDat(win, ID_GRID1, 0, 4, "Long Text")
```

which results in



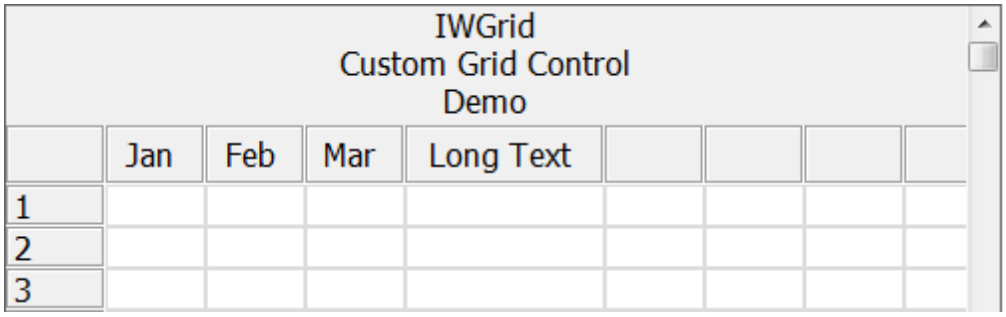
Notice that the full title is not displayed. The title has been truncated and the ellipses indicate that there is more text that is not visible.

With the default grid settings the End User will never know what the rest of the title is. Using the previously described IWG_ColResizing command (above) the User can give the End User the ability to drag the column divider to increase the size of the column until the full title can be seen.. Alternately, the User can widen the column with the previously discussed IWG_SetColWidth command (above).

Using this command line

```
IWG_SetColWidth(win, ID_GRID1, 4, 100)
```

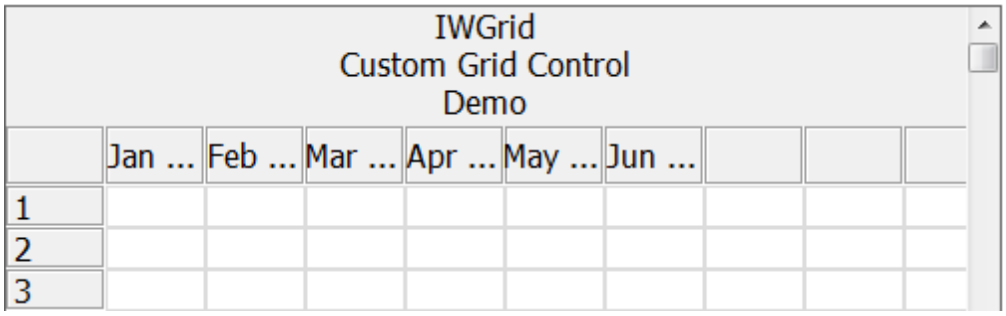
the grid now appears as



There are times when the above would be satisfactory. However, assume grid needs to show 6 columns which will contain data from the first six months of 2013. Starting with the basic default grid and the following commands

```
IWG_InitCellDat(win, ID_GRID1, 0, 1, "Jan 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 2, "Feb 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 3, "Mar 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 4, "Apr 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 5, "May 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 6, "Jun 2013",@IWGLABEL,@IWG_CENTERALIGN)
```

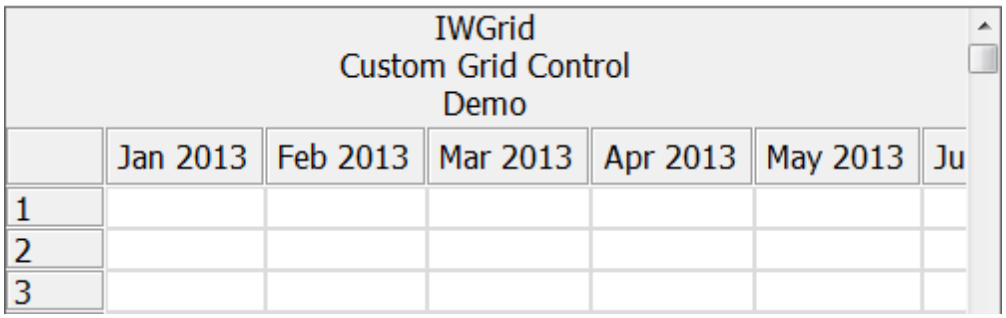
which results in



By using the IWG_SetColumnAutoWidth command before any IWG_InitCellDat commands are executed, the following code:

```
IWG_SetColumnAutoWidth(win, ID_GRID1, 1)
IWG_InitCellDat(win, ID_GRID1, 0, 1, "Jan 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 2, "Feb 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 3, "Mar 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 4, "Apr 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 5, "May 2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 6, "Jun 2013",@IWGLABEL,@IWG_CENTERALIGN)
```

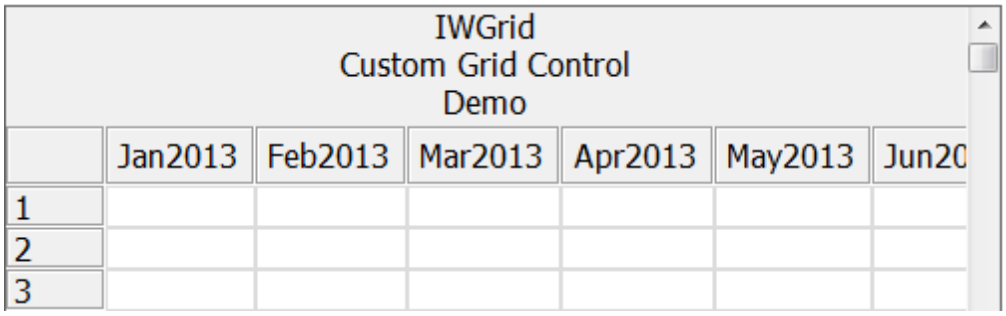
results in the following, which allows the full title to be read.



However, the 6th column is not fully visible which was a requirement. Assume that the overall grid width can not be changed.
That leaves the option to make the column title header two rows high. Just like the grid title, this can be done by inserting a "\n" in the column title string where a linefeed/carriage return is desired. That changes the command lines to

```
IWG_SetColumnAutoWidth(win, ID_GRID1, 1)
IWG_InitCellDat(win, ID_GRID1, 0, 1, "Jan\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 2, "Feb\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 3, "Mar\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 4, "Apr\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 5, "May\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 6, "Jun\n2013",@IWGLABEL,@IWG_CENTERALIGN)
```

which results in

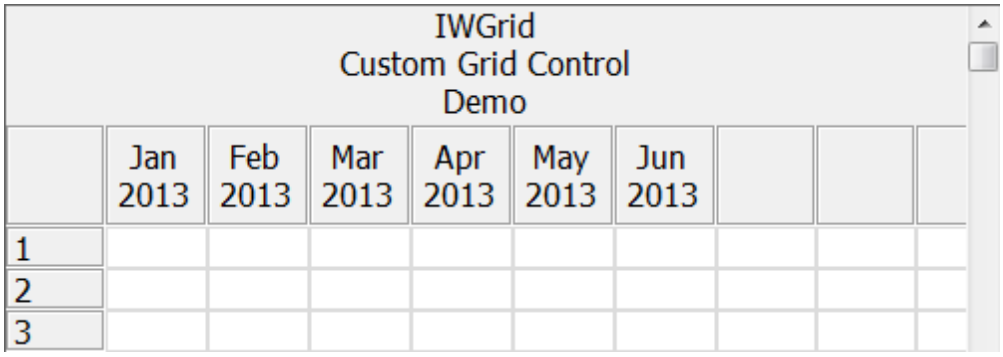


Unlike the the grid title, which is always in auto height adjust mode, the column title header row is by default in fixed height mode. While in this default mode any "\n" characters in the column title string are stripped and all the text shown on one line.

The height mode can be changed with the IWG_SetHeaderAutoHeight command. Like the IWG_SetColumnAutoWidth command this command also has to be executed prior to any IWG_InitCellDat commands being executed. Therefore, the following arrangement of commands:

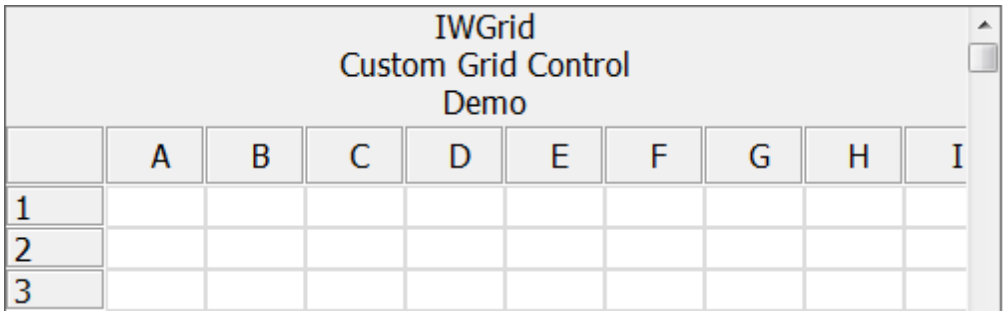
```
IWG_SetHeaderAutoHeight(win, ID_GRID1,1)
IWG_SetColumnAutoWidth(win, ID_GRID1, 1)
IWG_InitCellDat(win, ID_GRID1, 0, 1, "Jan\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 2, "Feb\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 3, "Mar\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 4, "Apr\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 5, "May\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 6, "Jun\n2013",@IWGLABEL,@IWG_CENTERALIGN)
```

results in the desired grid, shown below.



3.3.2 Font

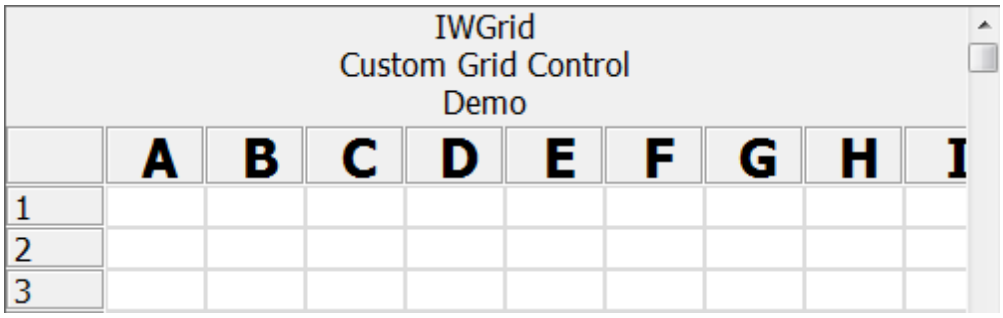
The default column title font is Tahoma with a height of 12 and a weight of 400. The following shows a default grid with the default column titles using the default font:



The default font can be changed using the IWG_SetFont command with the @IWGHEADERFONT flag. The IWG_SetFont command uses the same basic parameters used by IWBasic's SetFont command (typeface, height, weight, and flags). A typical command would be:

```
IWG_SetFont(win, ID_GRID1, @IWGHEADERFONT, "Tahoma", 20, 800, 0)
```

which results in the following:

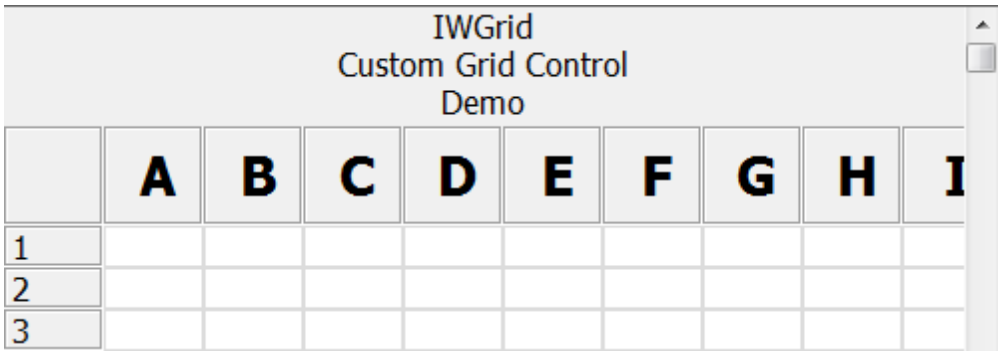


Notice the height of the header itself (compared to the previous example of the default font). Although the height of the font changed the height of the header title row did not change. Unlike the the grid title, which is always in auto height adjust mode, the column title header row is by default in fixed height mode. While in this default mode any change in the height of the txt in the column title string has no affect on the column title header row.

When the columns are auto-numbered, the only way to increase the height of the header title row is with the IWG_SetHeaderHeight. The following command line

```
IWG_SetHeaderHeight(main, 501,50)
```

results in:



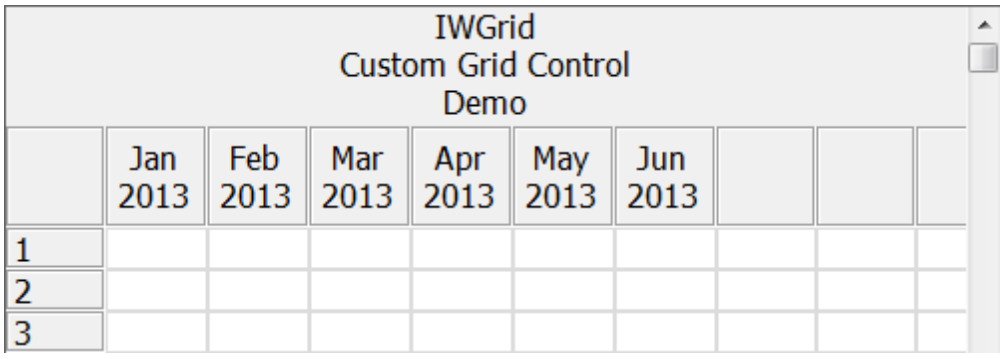
When User define column header titles are enabled (with IWG_AutoNumberHeader), the height mode can be changed with the IWG_SetHeaderAutoHeight command. Like the IWG_SetColumnAutoWidth command this command also has to be executed prior to any IWG_InitCellDat commands being executed. Therefore, the following command:

```
IWG_SetHeaderAutoHeight(win, ID_GRID1,1)
```

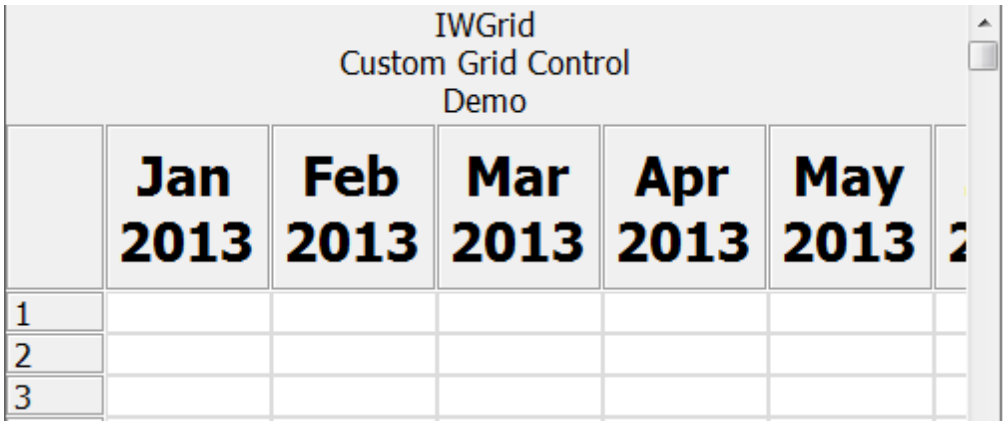
Taking an earlier example, the following code

```
IWG_SetHeaderAutoHeight(win, ID_GRID1,1)
IWG_SetColumnAutoWidth(win, ID_GRID1, 1)
IWG_InitCellDat(win, ID_GRID1, 0, 1, "Jan\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 2, "Feb\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 3, "Mar\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 4, "Apr\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 5, "May\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(win, ID_GRID1, 0, 6, "Jun\n2013",@IWGLABEL,@IWG_CENTERALIGN)
IWG_SetFont(win, ID_GRID1, @IWGHEADERFONT, "Tahoma", 20, 800, 0)
```

this grid

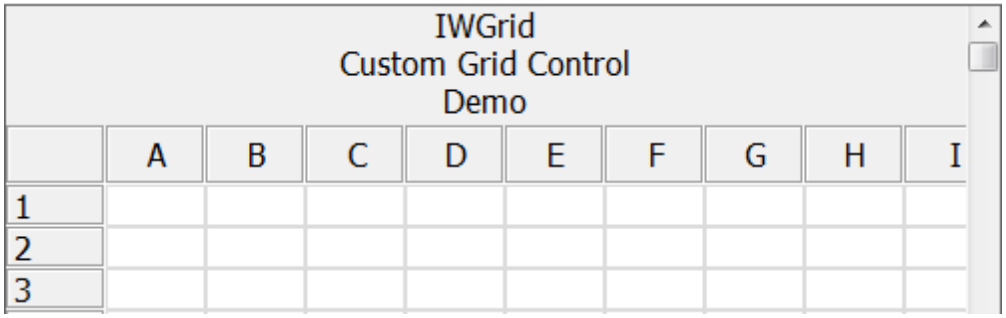


to this grid.



3.3.3 Shadow

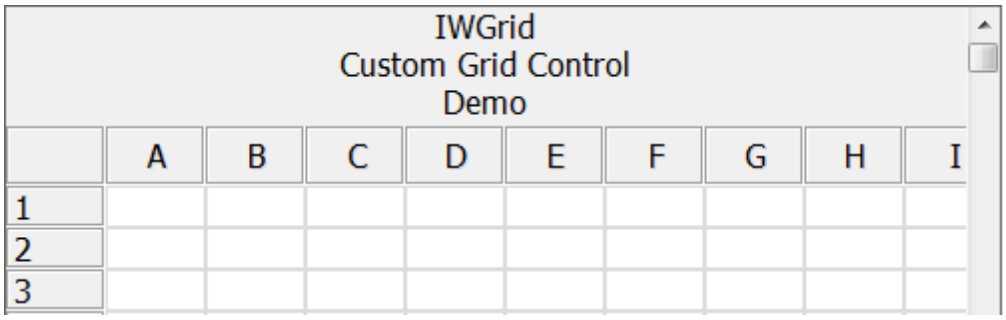
The column header titles may have an optional shadow applied to the titles' text. As an example, shown below is a column header row with default titles, default font and with no shadow(default):



Using the IWG_ShowHeaderShadow command a shadow can be applied to all the titles' text:

```
IWG_ShowHeaderShadow(main, 501, 1)
```

which results in the following very subtle change:



The shadows appear to be of little or no value if used with the default colors. The next section will show that the shadows can, indeed, have a dramatic effect.

The IWG_SetShadowOffset command can be used to used to move the shadow down and to the right with positive values and up and to the left with negative values. The default value is 1 pixel.

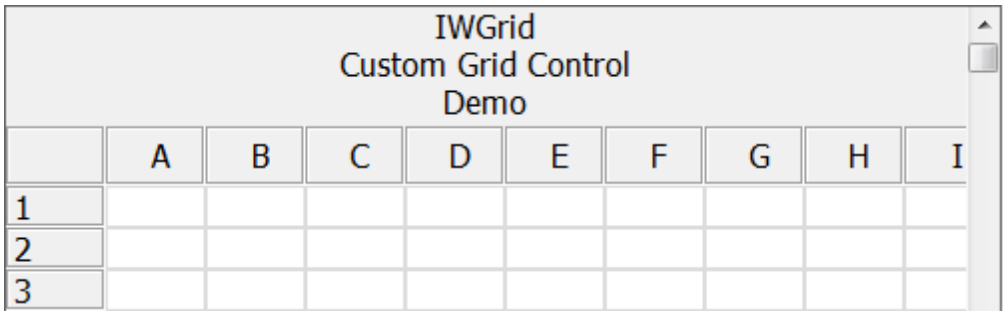
3.3.4 Color

The following commands affect Grid Column Header coloring

IWG_SetGridColor (when used with the following type flags)

- @IWGHEADERCOLOR - The header row fg and bg colors
- @IWGHEADERSHADOWCLR - The color of the shadow under the header's text. [default =RGB(220,220,220)]

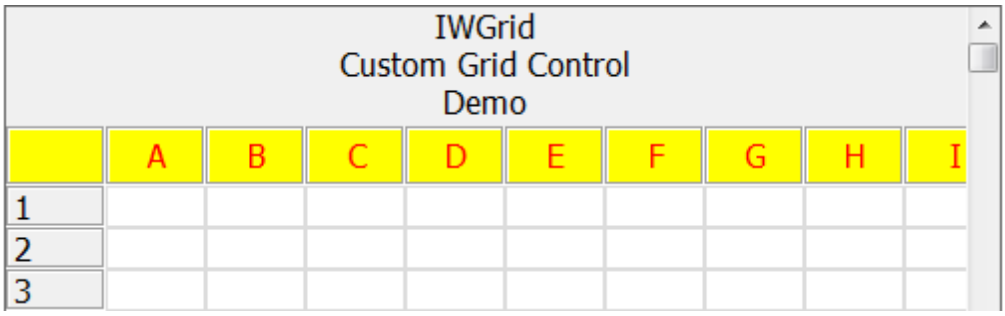
The following is the default column header row using the default colors (and the column header row title shadow turned off - default):



The IWG_SetGridColor command with the @IWGHEADERCOLOR flag can be used to change the text color and background color. The following command:

```
IWG_SetGridColor(win, ID_GRID1, @IWGHEADERCOLOR, RGB(255,0,0), RGB(255,255,0))
```

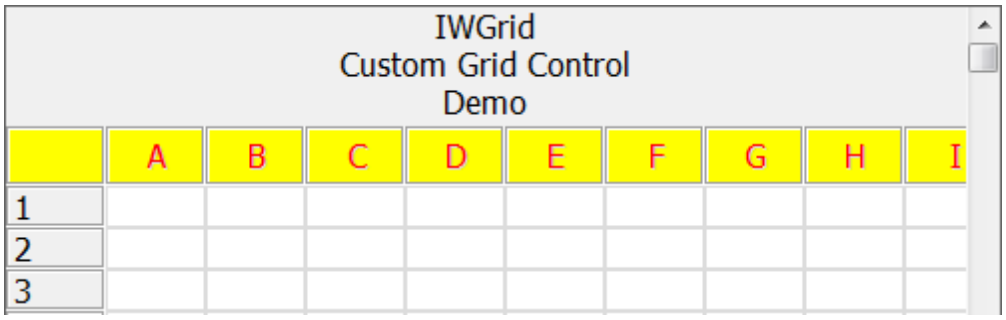
results in the column header row titles shown below.



Turning on the column header row title shadow (with the default shadow color) using

```
IWG_ShowHeaderShadow(win, ID_GRID1,1)
```

results in:



The column header row titles' shadow color can be changed with the IWG_SetGridColor command with the @IWGHEADERSHADOWCLR flag. The following command

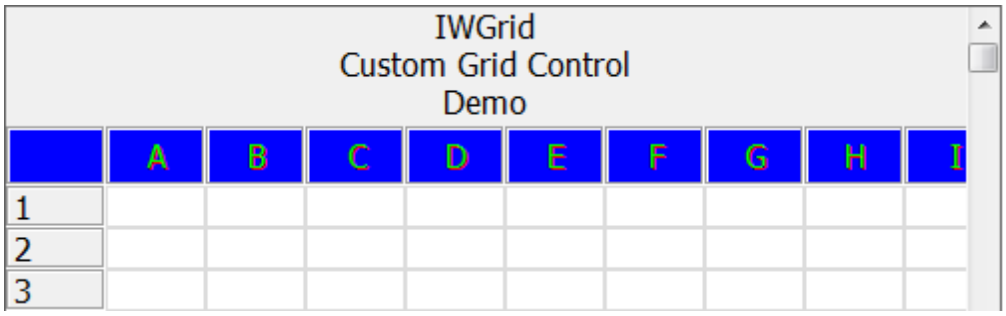
```
IWG_SetGridColor(win, ID_GRID1, @IWGHEADERSHADOWCLR, RGB(0,255,0))
```

gives the following column header row titles.

Another example would be

```
IWG_ShowHeaderShadow(win, ID_GRID1,1)
IWG_SetGridColor(win, ID_GRID1, @IWGHEADERCOLOR, RGB(0,255,0), RGB(0,0,255))
IWG_SetGridColor(win, ID_GRID1, @IWGHEADERSHADOWCLR, RGB(255,0,0))
```

gives the following:



NOTE: The User can use the *Color Picker* tool from IWBasic's IDE *Tools Menu* to facilitate the choice of colors.

3.4 Grid Body

The IWGrid grid body is the heart of the grid. The following sub-sections describe all the grid body options and commands available to the User. Excludes cell specific options.
General
Color
Font

3.4.1 General

The general commands deal with the grid body as a whole as opposed to individual cells. Not included in the list are the commands involving color and fonts. They are covered in following sub-sections.

The general commands can be divided into two groups; those that impact the grid and those that supply information about the grid.

Configuration Commands

Command	Purpose
IWG_AutoAddRow	Automatically increases the number of rows in the grid when a cell is added that is outside the current bounds. Default = 1
IWG_DeleteAllCells	Clears all cell entries in the grid
IWG_SetColumnAutoWidth	Automatically sets the column width to accommodate the longest text in the column. Default = 0
IWG_SetColWidth	Sets the default width of an individual column. Default = 50
IWG_SetCursorPosition	Sets the grids current cell position and draws the cursor box. Default r = 1,c = 1
IWG_SetDim	Sets the desired number of rows and columns in the grid. Default rows = 100; columns = 255
IWG_SetHiLightRow	Used to control the highlighting of the entire row containing the current cell. Default = 0 And example is shown in the Color section.
IWG_SetRowAutoHeight	Used to allow cell contents to control overall row height. Default = 0
IWG_SetRowHeight	Sets the height of all rows in the grid. Default = 21 pixels
IWG_ShowRowNumbers	Used to show/hide row numbers. Default = 1

Informational Commands

Command	Purpose
IWG_CellLocate	Utility to convert the contents of @LPARAM to its row and column components (when the grid receives a notification message from a cell).
IWG_GetColumns	Used to get the number of columns in the grid.
IWG_GetColumnWidth	Used to get the width of a specified column..
IWG_GetCurrentCol	Used to get the column that currently has the focus.
IWG_GetCurrentRow	Used to get the row that currently has the focus.

IWG_GetDim	Used to get the number of rows and columns currently in the grid.
IWG_GetRowHeight	Used to get the height of the rows in the grid.
IWG_GetRows	Used to get the number of rows in the grid.

3.4.2 Color

The following commands affect Grid Body coloring on a global basis for all cells.

IWG_SetGridColor (when used with the following type flags)

- @IWGPROTECTCOLOR - The background color of protected cells. [default = RGB (255,255,255)]
- @IWGUNPROTECTCOLOR - The background color of un-protected cells. [default =RGB (255,255,255)]
- @IWGHILIGHTCOLOR - The background color of highlighted cells. [default = RGB(0,0,128)]
- @IWGHILIGHTTEXTCOLOR - The text color of highlighted cells [default =RGB (255,255,255)]
- @IWGCURSORCOLOR - The cell cursor box color. [default = RGB(255,255,255)]
- @IWGGRIDLINECOLOR - The color of the grid's gridlines. [default = RGB(220,220,220)]

IWG_SetHiLightRow
Used to set an internal flag that causes the @IWGHILIGHTCOLOR and @IWGHILIGHTTEXTCOLOR to be applied to all cells in a row when any cell in the row has been selected as the current cell.

To demonstrate the affect of the various colors the following grid will be used, starting with all default colors.

	A	B	C	D
1				
2	123456			
3		123456		
4			123456	
5				123456
6				

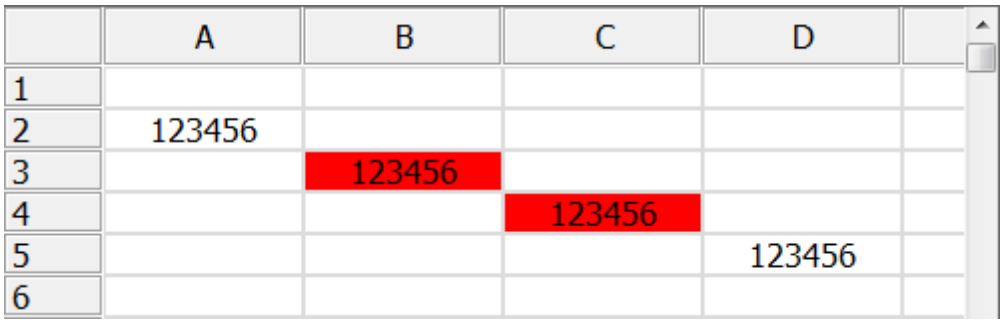
NOTE: For the purpose of the following examples the type of the cells that contain data is of no importance. However, it is important to note that the B3 and C4 cells have been "protected" from editing via the IWG_SetCellProt command.

@IWGPROTECTCOLOR

The following command changes the background color of the protected cells to red:

```
IWG_SetGridColor(main, 501, @IWGPROTECTCOLOR, RGB(255,0,0))
```

which gives the following grid:

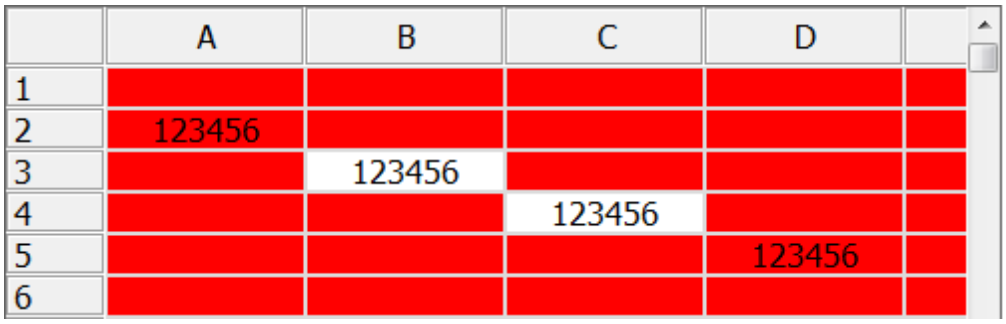


@IWGUNPROTECTCOLOR

The following command changes the background color of the un-protected cells to red:

```
IWG_SetGridColor(main, 501, @IWGUNPROTECTCOLOR, RGB(255,0,0))
```

which gives the following grid:

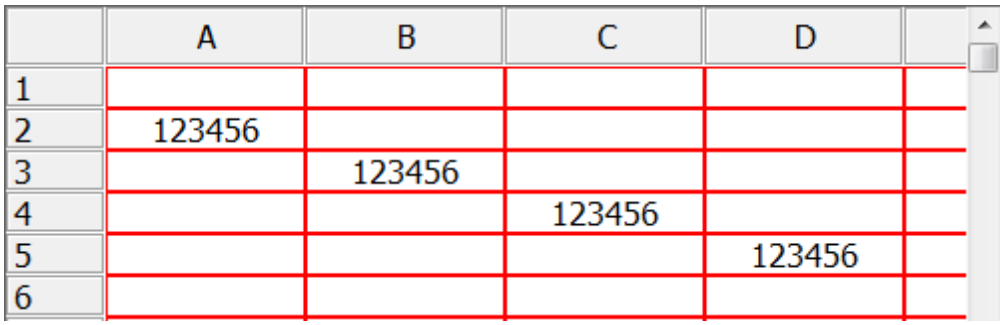


@IWGGRIDLINECOLOR

The following command changes the background color of the grid lines to red:

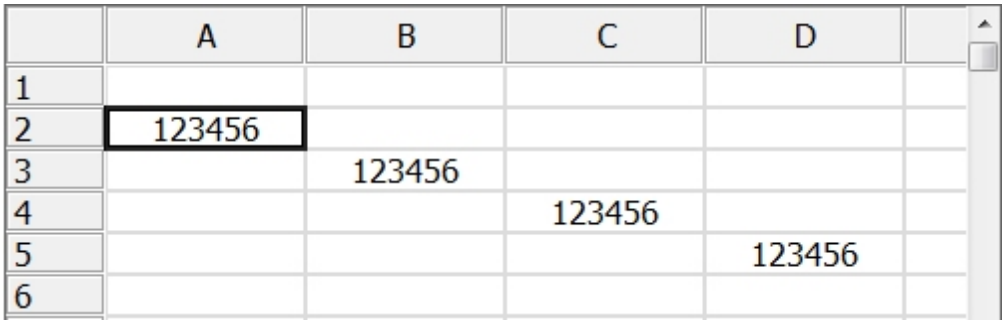
```
IWG_SetGridColor(main, 501, @IWGGRIDLINECOLOR, RGB(255,0,0))
```

which gives the following grid:



@IWGCURSORCOLOR

When a cell is currently selected, but not being edited, it has a Cursor "box" drawn around the cell as shown below:

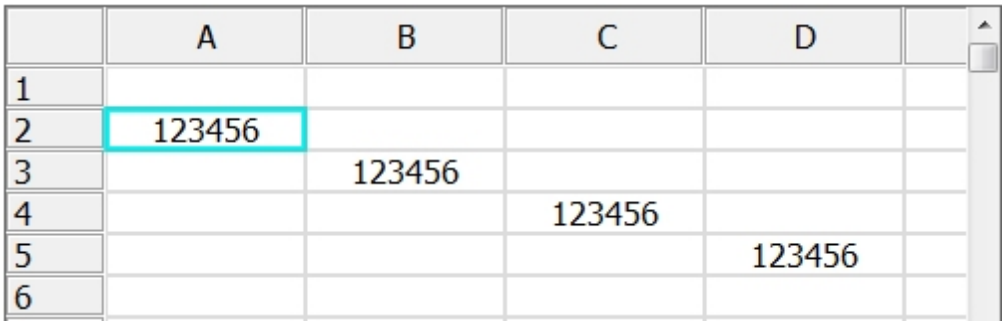


The current cursor box color is exclusive or'ed with the current color of the background color under the cursor.

The following command changes the cursor box color to red:

```
IWG_SetGridColor(main, 501, @IWGCURSORCOLOR, RGB(255,0,0))
```

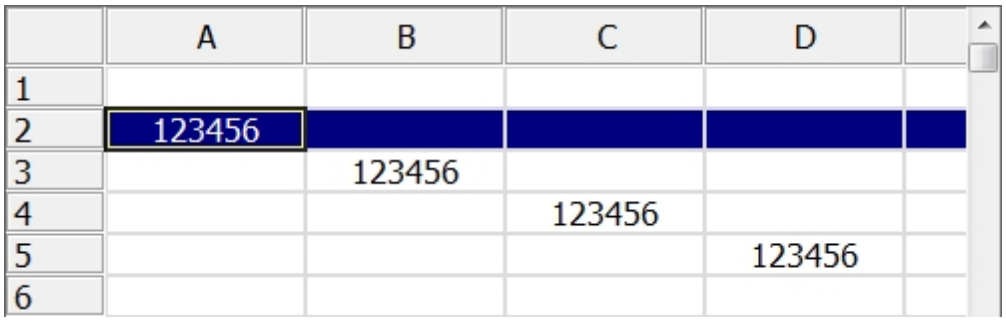
which gives the following grid:



The resulting color is not red because of the XORing of the two colors.

When the IWG_SetHiLightRow command is used to turn on row highlighting two other colors come into play. These two colors will apply to the text and background in the same row as the currently selected cell.

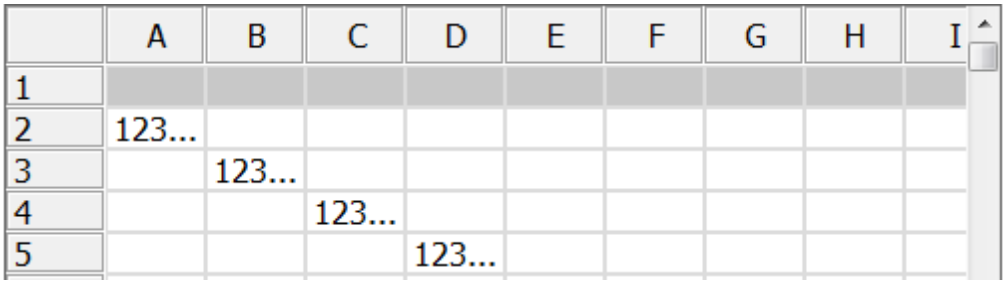
The following shows the default appearance when the highlighting is turned on.



See the Grid Cell > Color section for a further discussion of cell and grid body coloring, and the order of precedence of applying colors.

3.4.3 Font

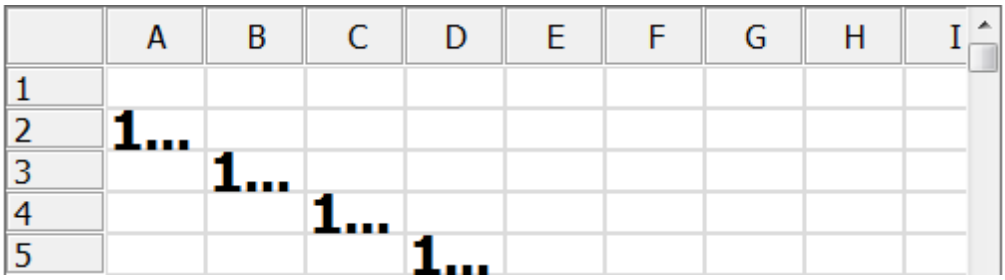
The default cell font for the row number header column and for the main grid body is Tahoma with a height of 12 and a weight of 400. The following shows a default grid with configured cells using the default fonts:



The default fonts can be changed using the IWG_SetFont command with the @IWGCELLFONT or @IWGROWNUMFONT flag. The IWG_SetFont command uses the same basic parameters used by IWBasic's SetFont command (typeface, height, weight, and flags). A typical command would be:

```
IWG_SetFont(win, ID_GRID1, @IWGCELLFONT, "Tahoma", 20, 800, 0)
```

which results in the following:

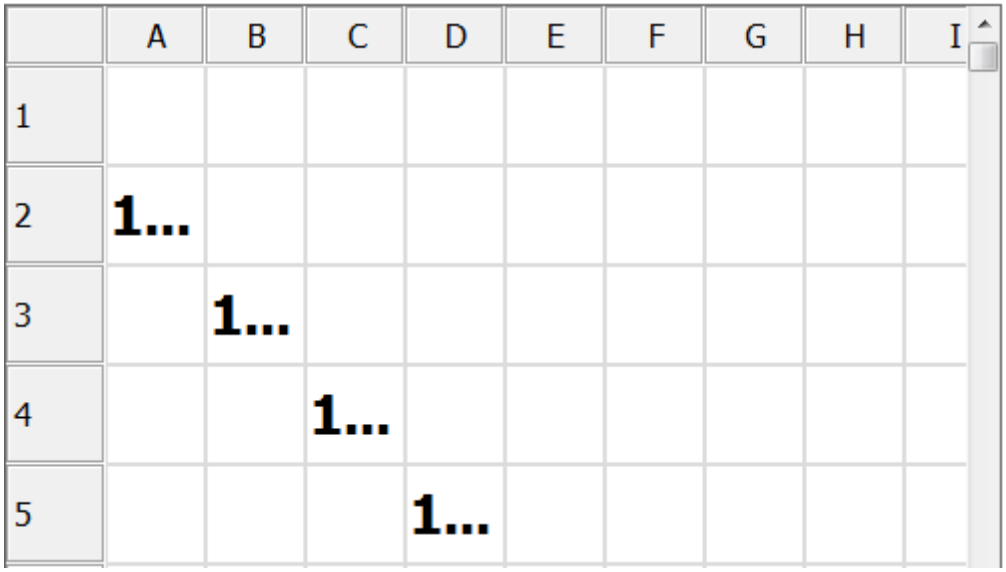


Notice the height of the cell rows (compared to the previous example of the default cell font). Although the height of the font changed the height of the rows did not change. Unlike the the grid title, which is always in auto height adjust mode, the column title header row is by default in fixed height mode. While in this default mode any change in the height of the txt in the column title string has no affect on the column title header row. Also notice that the row number column font has not changed.

One way to increase the height of the cell rows is with the IWG_SetRowHeight. The following command line

```
IWG_SetRowHeight(main, 501,50)
```

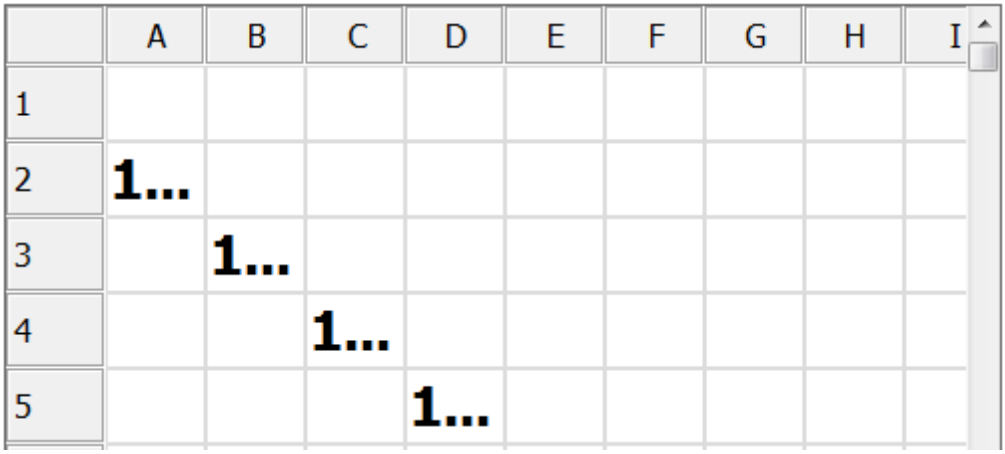
which can be used at any time, results in:



The height can also be changed with the IWG_SetRowAutoHeight command. This command has to be executed prior to any IWG_InitCellDat commands being executed. Therefore, the following command:

```
IWG_SetRowAutoHeight(win, ID_GRID1,1)
```

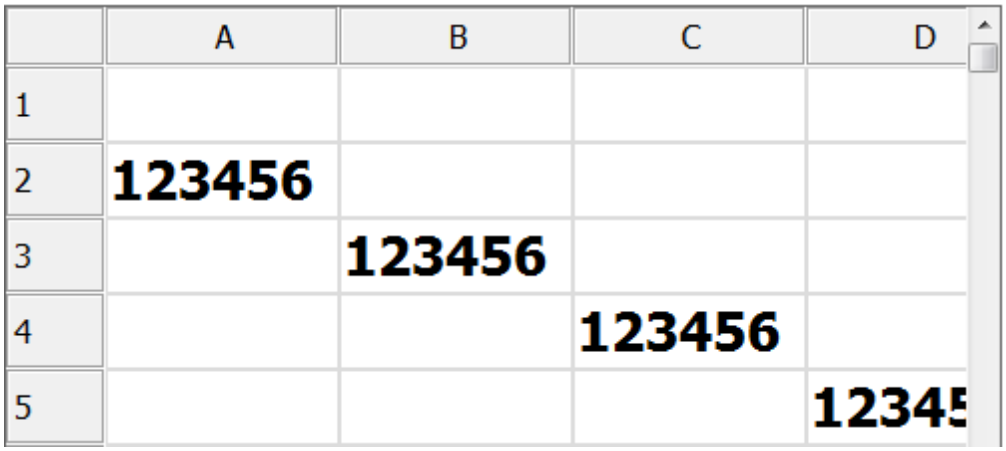
results in this grid



In all the examples so for the cell contents has been visually truncated with ellipses. As discussed in the Getting Started > Grid Column Header the column width can be increased at any time with the IWG_SetColWidth command or, prior to the execution of any IWG_InitCellDat commands, with IWG_SetColumnAutoWidth command.

```
IWG_SetColumnAutoWidth(win, ID_GRID1, 1)
```

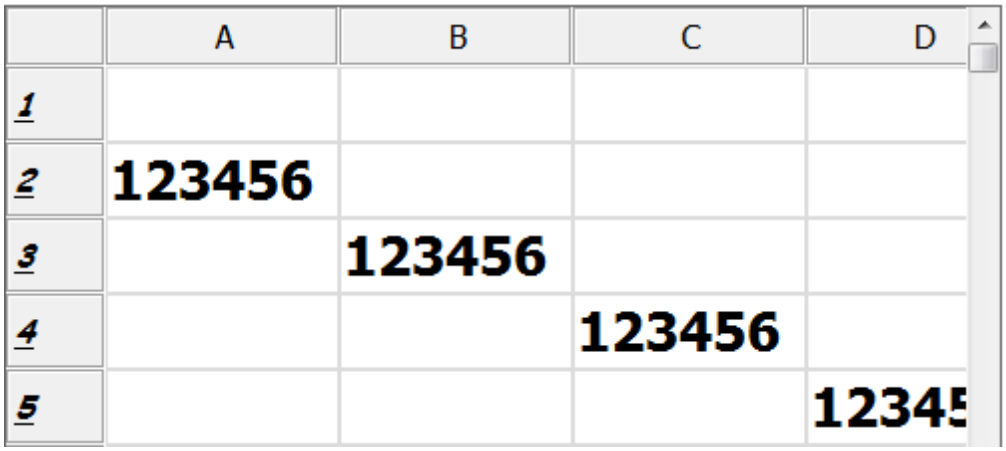
results in



And, to show that the row number column font can indeed be changed. The following command

```
IWG_SetFont(win, ID_GRID1, @IWGROWNUMFONT, "comic sans ms", 12, 800,@SFITALIC|
@SFUNDERLINE)
```

results in



3.5 Grid Cells

The individual cell is the heart of IWGrid. The following sub-sections describe all the grid cell options and commands available to the User.

Initalizing
Updating
Editing
Reading
Color
Font
Cell Types

3.5.1 Initializing

Each cell in a grid can be individually configured or left un-configured. The IWG_InitCellDat command is used to perform the initial configuration of a cell. Exception: If the cell is a @IWGMASK type cell then the IWGM_InitMaskCellDat command is used. In most cases these are the only commands needed for configuration. Exceptions will be discussed in the appropriate sub-section of Getting Started > Grid Cells > Cell Types.

The format of the IWG_InitCellDat command is

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, *celltype*, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.
celltype	-	The constant designation for the type of cell. Use one of the 15 constants in the table below (and described in the Getting Started > Grid Cells > Cell Types section).
flags	-	The optional styles to be applied to the cell's contents. See the table below and the specified cell type's section in Getting Started > Grid Cells > Cell Types .

Valid cell type constants:

@IWGLABEL	-	Static text that can not be selected nor edited.
@IWGANY	-	Any ASCII character between 0x20 and 0xFF is allowed
@IWGTXTNUM	-	A-Z, a-z, 0-9, '-' , and '.'
@IWGTXT	-	A-Z, a-z
@IWGNUMI	-	0-9 and '-'
@IWGNUMF	-	0-9, '-', and '.'
@IWGHEX	-	0-9, A-F, a-f
@IWGCOMBO	-	Any ASCII character between 0x20 and 0xFF
@IWGCHECK	-	A checked or unchecked checkbox
@IWGSUBBUTTON	-	A text field that accepts any ASCII character between 0x20 and 0xFF and a button. Clicking the button passes the edit field to a User defined subroutine. The output of the subroutine returns a string which is placed in the edit field.

@IWGMASK	-	Allows a User defined mask to format the input data. Note: Must be used with the IWGM_InitMaskCellDat command.
@IWGMONEY	-	0-9, '-', and '.'. The input is automatically formatted to optionally include a currency symbol. Also thousand separators(,) are inserted at the proper locations and decimals are rounded/ truncated to the proper number of decimal places.
@IWGIMAGE	-	A scalable .bmp, .jpg, or .gif image
@IWGBUTTON	-	A button that looks and acts much like a normal IWBasic @BUTTON control.
@IWGIPADDRESS	-	An IP address.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default. Does not apply to @IWGCOMBO, @IWGCHECK, and @IWGSUBBUTTON cell types.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right. Does not apply to @IWGCOMBO, @IWGCHECK, and @IWGSUBBUTTON cell types.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center. Does not apply to @IWGCOMBO, @IWGCHECK, and @IWGSUBBUTTON cell types.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.
@IWG_NOSCALE	-	Used only with @IWGIMAGE type cells
@IWG_RATIO	-	Used only with @IWGIMAGE type cells

3.5.2 Updating

Once a cell has been initialized it may later be update under program control via the IWG_SetCellDat. See exceptions below.

The format of the IWG_SetCellDat command is

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Exceptions:

1) Due to the fact that the initialization of an @IWGIMAGE type cell creates a handle to a bitmap the IWG_SetCellDat command can not be used with that type of cell.
2) For different reasons the same is true for the @IWGCHECK type cell.
3) For @IWGMASK type cells the IWGM_SetCellFormatDat or IWGM_SetCellRawDat command must be used.

See the details in their respective sub-sections in the Getting Started > Grid Cells > Cell section.

3.5.3 Editing

At runtime, the End User can modify most configured cells in some form or fashion. This is true only when the entire grid is not protected and the selected cell is not individually protected. The modification process is initiated by clicking the desired cell.

The specifics of how cells may be modified are covered in the sub-sections of the Getting Started > Grid Cells > Cell Types sub-sections.

3.5.4 Reading

Once a cell has been initialized it may later have its current contents read under program control via the IWG_GetCellDat. See exceptions below.

The format of the IWG_GetCellDat command is

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell.

Exceptions:

1) The @IWGCHECK type cell uses the IWG_GetCheckState command.
3) For @IWGMASK type cells the IWGM_GetCellFormatDat or IWGM_GetCellRawDat command must be used.

See the details in their respective sub-sections in the Getting Started > Grid Cells > Cell Types sub-sections.

3.5.5 Color

To obtain the maximum benefit from individual cell coloring, the User must have a good understanding of:

- the various commands used to define colors that may affect a cell's color;
- the conditions under which various colors are applied to a cell;
- the commands that control grid/cell options that can change those conditions;
- and the order of precedence that colors are applied to a cell.

Color commands that may affect a cell's color

IWG_SetGridColor (when used with the following type flags)

- @IWGPROTECTCOLOR - The background color of protected cells. [default = RGB (255,255,255)]
- @IWGUNPROTECTCOLOR - The background color of un-protected cells. [default =RGB (255,255,255)]
- @IWGHILIGHTCOLOR - The background color of highlighted cells. [default = RGB(0,0,128)]
- @IWGHILIGHTTEXTCOLOR - The text color of highlighted cells [default =RGB (255,255,255)]

(The above are covered in the Getting Started > Grid Body > Color section.)

IWG_SetCellColor
Used to set an individual cell's foreground(text) and background colors.
The default foreground and background color is RGB(0,0,0)

IWG_SetCellImgMissingColor
Used to set an individual @IWGIMAGE type cell's foreground(text) and background colors when the image is missing.
The default foreground and background color is RGB(0,0,0)

Non-color commands that control options which may affect a cell's color

IWG_SetCellProt

Used to switch a cell between being protected and unprotected. At the same time it is used to switch between @IWGPROTECTCOLOR and @IWGUNPROTECTCOLOR for the cells background. (The above is covered in the Getting Started > Grid Body > Color section.)

IWG_SetHiLightRow
Used to set an internal flag that causes the @IWGHILIGHTCOLOR and @IWGHILIGHTTEXTCOLOR to be applied to all cells in a row when any cell in the row has been selected as the current cell.
(The above are covered in the Getting Started > Grid Body > Color section.)

Order of precedence of cell color application

1. If the IWG_SetHiLightRow command has been used to turn that feature on
    a. All cells in the row that contains the currently selected cell will be colored as follows:
        1) If the grid has focus, the cells text color will be @IWGHILIGHTTEXTCOLOR and the background color will be @IWGHILIGHTTEXTCOLOR.
        2) If the grid does not have focus, the cells text color will be RGB(0,0,0) and the background color will be RGB(200,200,200).
    b. All other cells will follow the rules in paragraph 2 below.

2. If the IWG_SetHiLightRow command has been used to turn that feature off (or it was never turned on)
    a. If the cell type is @IWGIMAGE and the image is present
        1) Follow the rules in paragraph 2c.
    b. If the cell type is @IWGIMAGE and the image is missing
        1) If the colors set by IWG_SetCellImgMissingColor are not black on black (default) then those colors will be used; otherwise the rules in paragraph 2c will be followed.
    c. If the colors set by IWG_SetCellColor are not black on black (default) then those colors will be used; otherwise the rules in paragraph 2d will be followed.
    d. The text will be black.
        1) If the cell is "protected" via IWG_SetCellProt the background will be @IWGPROTECTCOLOR; otherwise the background will be @IWGUNPROTECTCOLOR

3.5.6 Font

Every cell in the grid body (except for the row number column, uses the same font parameters. A complete discussion is covered in the Getting Started > Grid Body > Font section.

3.5.7 Cell Types

The IWGrid control supports the following fifteen cell types. Each is discussed in its own section.

@IWGANY
@IWGBUTTON
@IWGCHECK
@IWGCOMBO
@IWBHEX
@IWGIMAGE
@IWGIPADDRESS
@IWGLABEL
@IWGMASK
@IWGMONEY
@IWGNUMF
@IWGNUMI
@IWGRADIOBUTTON
@IWGSUBBUTTON
@IWGTXT
@IWGTXTNUM

3.5.7.1 @IWGANY

General

The @IWGANY type cell is used when the End User needs to able to enter any printable ASCII character. The intended use for this cell type is to allow free-form text, paragraphs, and data base MEMO field data to be stored in a cell. (Limited to 254 characters.)

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGANY, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Limited to 254 characters. Any ASCII character between 0x20 and 0xFF is allowed.
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being read.
text	-	The current text in the cell.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
text	-	The new text to place in the cell. Limited to 254 characters.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.

r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

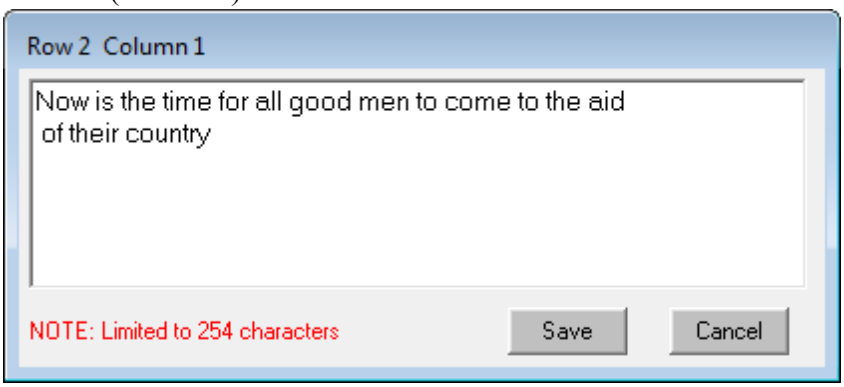
Editing

The End User may edit the contents of an @IWGANY type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGANY cell entry.

	A	B
1		
2	Now is the	
3		

If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



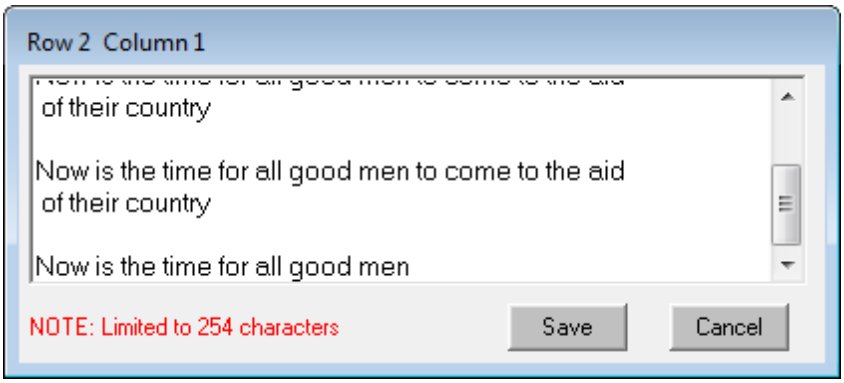
The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text either by typing or using *CTRL-V* to paste text from the clipboard.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X*/*CTRL-V* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

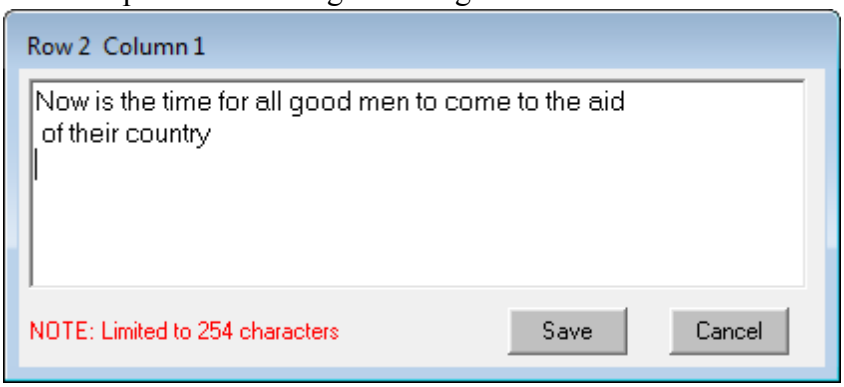
Notes:

1. The content of the cell can never exceed 254 characters under any circumstances.
2. Pressing the *Enter* key will create a LF/CR which accounts for 2 characters.

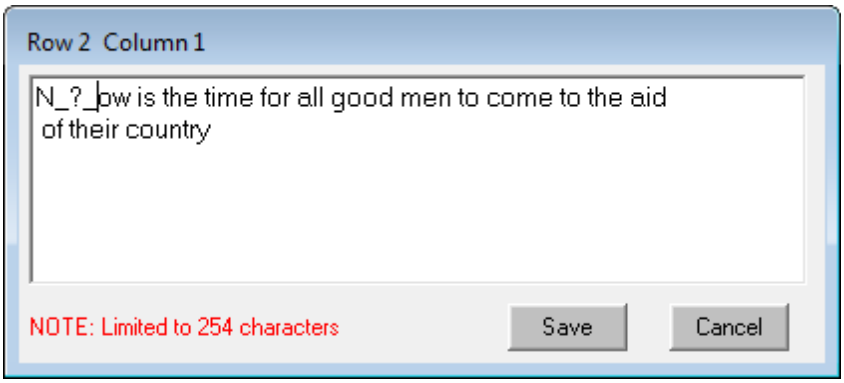
The following was created buy using *CTRL-C* to copy the original contents show above and repeatedly using *CTRL-V* to append the text to the cell. By scrolling down, the last line can be seen to be truncated. That is because the 254 character limit was reached at that point.



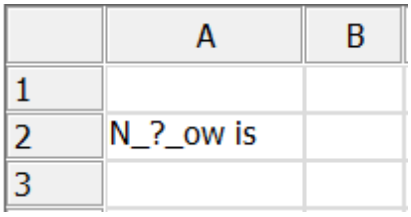
This example shows the original text again.



The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.



	A	B
1		
2	N_?_ow is	
3		

3.5.7.2 @IWGBUTTON

General

The @IWGBUTTON type cell is used when the End User needs to be able to click a button much like an IWBasic @BUTTON control.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGBUTTON, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Any ASCII character between 0x20 and 0xFF is allowed.
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).

Configuration

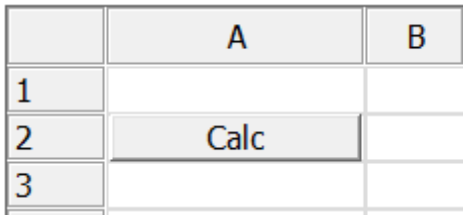
Without further configuration the initialized button will appear the same size as the cell. i.e. The button will fill the cell.
The @IWGBUTTON size can be configured with

IWG_SetCellButtonSize(*win*, *id*, *r*, *c*, *w*, *h*)

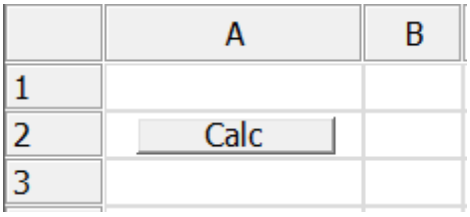
where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
w	-	If the value of w = 0 or w > cell width then w = cell width and is centered in the cell.
h	-	If the value of h = 0 or h > cell height then h = cell height and is centered in the cell.

The following shows an initialized @IWGBUTTON:



and this is the same, but the IWG_SetCellButtonSize command has been used to make the button smaller than the cell size.



Reading

The button label can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current button label text in the cell.

The button size can be read with

text = IWG_GetCellButtonSize(*win*, *id*, *r*, *c*, *w*, *h*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
w	-	Predefined variable to hold button width
h	-	Predefined variable to hold button height.

Updating

The button label can be changed with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text for the button label.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

Editing

There is no editing available to the End User. Left-clicking the button will cause an @IWGN_BUTTONCLICKED message to be sent to the grid parent.

3.5.7.3 @IWGCHECK

General

The @IWGCHECK type cell is used when the End User needs to able to select an option, much like an IWBasic @CHECKBOX control..

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *state*, @IWGCHECK, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
state	-	The initial state of the checkbox. • "1" - Checked • "0" - UnChecked
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).

Reading

The current state of the cell can be read with

state = IWG_GetCheckState(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.

and the current state is.

- 1 - Checked
- 0 - UnChecked

Updating

The current state of the cell can be changed with

IWG_SetCheckState(*win*, *id*, *r*, *c* , *s*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
s	-	The desired state. • 1 - Checked • 0 - UnChecked

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

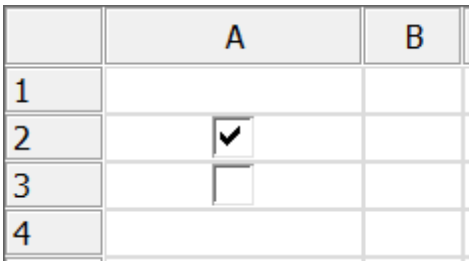
where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

Editing

The End User may change the state of the cell by left-clicking the cell. Each click will toggle the state of the currently selected cell. The following shows both a checked and un-checked @IWGCHECK cell.



3.5.7.4 @IWGCOMBO

General

The @IWGCOMBO type cell is used when the End User needs to able to select an option from a dropdown menu, much like an IWBasic @COMBOBOX control..

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGCOMBO, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell which must match one of the dropdown list options.. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Configuration

The options contained in the dropdown list portion of the @IWGCOMBO cell can be configured with

IWG_SetCellCombo(win, id, r, c, text)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.

text	-	The string containing all the options in the dropdown list. Each option is separated by a "|". Total maximum number of characters allowed for any given cell is 10,000. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type
h	-	If the value of h = 0 or h > cell height then h = cell height and is centered in the cell.

The following is an example of configuring the dropdown list options:

```
istring g[10000]=""
int xx
for xx=1 to 250
   g+="Test Entry#"+str$(xx)+"|"
next xx
IWG_SetCellCombo(win, ID_GRID1, 2, 1, g)
```

Reading

The cell current selection can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell.

Updating

The current cell text selection can be changed with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. Text must be one of the dropdown list options. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

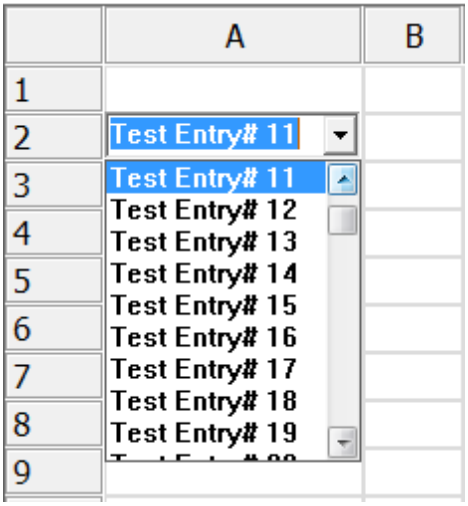
and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

Editing

The End User may change the current cell selection by typing a valid option in the edit control or selecting an option from the dropdown list. The following shows an @IWGCOMBO cell, first when it is not the currently selected cell

	A	B
1		
2	Test Entry# 11	
3		

and then with the dropdown list visible.



3.5.7.5 @IWGHEX

General

The @IWGHEX type cell is used when the End User input needs to be restricted to hexadecimal characters.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGHEX, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Allowable input is '0-9', 'a-f', and 'A-F'
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.

r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type. See above.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

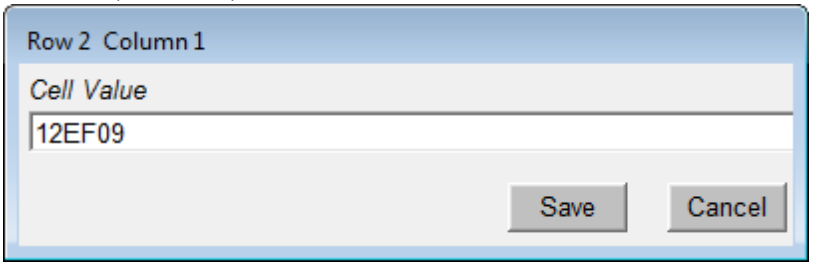
Editing

The End User may edit the contents of an @IWGHEX type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGHEX cell entry.

	A	B
1		
2	12EF09	
3		

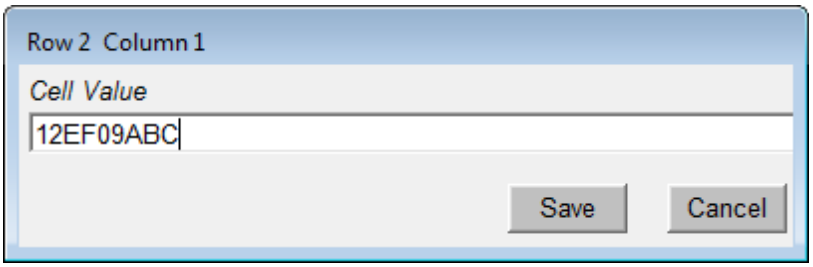
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text by typing valid text. Using *CTRL-V* to paste text from the clipboard is not allowed.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell,

shown below.

	A	B
1		
2	12EF09ABC	
3		

3.5.7.6 @IWGIMAGE

General

The @IWGIMAGE type cell is used to display a scalable bmp, jpg, or gif image.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGIMAGE, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	One of the following to identify the image to place in the cell. 1. The full path to the desired image. i.e. "C:\myrog\imgages\bug1.bmp" or, 2. A relative path to the desired image. i.e "images\bug1.bmp" or 3. The string name of a resource. i.e. "bmp1" or 4. The numeric id of a resource preceded by a '#'. i.e. "#2222"
flags	-	The optional styles to be applied to the cell's contents. See the table below.

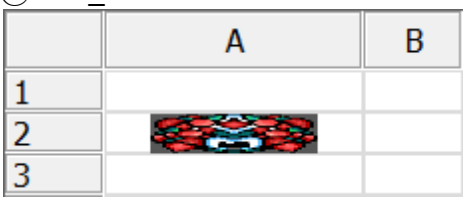
Valid flags:

@IWG_SCALE	-	If either, or both, dimensions of the actual image exceeds the corresponding dimension of the cell, that dimension of the image is scaled down to fit inside the cell space. Example below.
@IWG_NOSCALE	-	Each dimension is expanded/shrunk to fill the cell exactly. Example below.
@IWG_RATIO	-	The dimensions of the image are expanded/shrunk as necessary so that one image axis is equal to the corresponding cell axis and the other axis is less than or equal to its corresponding cell axis while maintaining the image ratio of x/y. Example below.

The following three examples are based up the following 84 by 100 image:

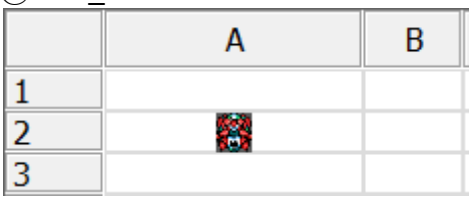
and the cell in each example is 130 x 21.

@IWG_SCALE



@IWG_NOSCALE

@IWG_RATIO



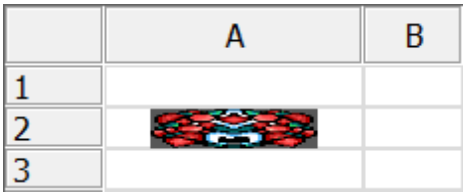
Configuration

There are several options for configuring an @IWGIMAGE type cell.

In the following discussions of the various configuration options it is assumed that the cell was initialized with

```
IWG_InitCellDat((main,  501, 2, 1, "bmp1", @IWGIMAGE, @IWG_SCALE)
```

and, with no configuration options selected, the grid cell appears like the following:



IWG_SetCellImageMissingText(*win*, *id*, *r*, *c*, *dat*)

can be used to set the "missing image" text for the selected cell.

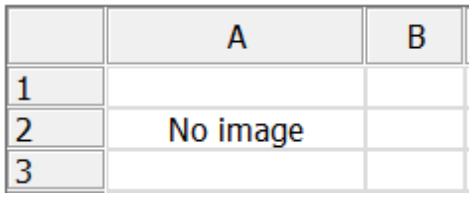
where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The text to be placed in the designated cell when the image is missing.

Example

```
IWG_SetCellImageMissingText(main,  501, 2, 1, "No image")
```

which displays the following if the image is not found for any reason



Note: If this command is not used then there will be no text displayed when an image is missing.

IWG_SetCellImgMissingColor(*win*, *id*, *r*, *c*, *fg, bg*)

can be used to set the text and background colors for the specified @IWGIMAGE type cell for when the image is missing.

where

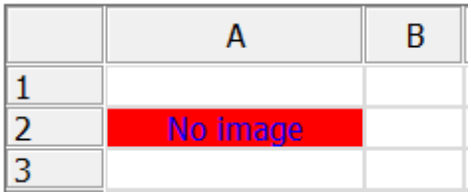
win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
fg	-	Foreground (text) color default = 0x000000
bg	-	Background color default = 0x000000

Remarks
See Grid Cells > Color

Example

```
IWG_SetCellImgMissingColor(main,  501, 2, 1, 0xff0000, 0x0000ff)
```

which displays the following if the image is not found for any reason



IWG_SetCellImageSize(*win*, *id*, *r*, *c*, *w, h)*

can be used to set the width and height of the image in the specified cell.

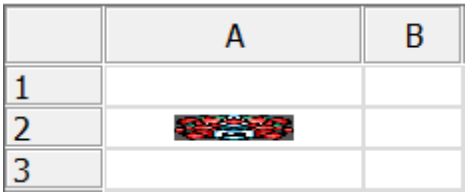
win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
w	-	Desired image width - If value is greater than cell width it will be adjusted to cell width.
h	-	Desired image height - If value is greater than cell height it will be adjusted to cell height.

Remarks
If this command is not used (or both w and h are set to 0) the size of the image will be automatically adjusted according to the settings of the @IWGNOSCALE and @IWGRATIO flags..

Example

```
IWG_SetCellImageSize(main,  501, 2, 1, 60, 14)
```

which displays the following



Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

STRING = IWG_GetCellImageMissingText(*win*, *id*, *r*, *c*)

Used to read the "missing image" text of the specified cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value

The "missing image"string contents of the cell at the specified position.

Example

```
STRING text = IWG_GetCellImageMissingText(main,  501, 2, 1)
```

UINT = IWG_GetCellImgMissingBGColor(win, id, r, c)

can be used to get the background color of the specified @IWGIMAGE type cell when the image is missing.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The background color the @IWGIMAGE type cell when the image is missing..

Example
```
uint mibgc = IWG_GetCellImgMissingBGColor(main,  501, 2, 1)
```

UINT = IWG_GetCellImgMissingFGColor(win, id, r, c)

can be used to get the foreground color of the specified @IWGIMAGE type cell when the image is missing.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The foreground color the @IWGIMAGE type cell when the image is missing..

Example
```
uint mibgc = IWG_GetCellImgMissingFGColor(main,  501, 2, 1)
```

IWG_GetCellImageSize(win, id, r, c, w, h)

can be used to set the width and height of the image in the specified cell.

where

win		The parent of the grid control.
id		The id of this instance of the grid control.
r		The row position of the desired cell.
c		The column position of the desired cell
w		Predefined variable to hold image width

h		Predefined variable to hold image height

Return value
Width and height of the selected image. If both values are 0 the image size is the same as the cell size

Example

```
INT w,h
IWG_GetCellImageSize(main,  501, 2, 1, w, h)
```

Updating

The cell can not be updated. To change to a new image the User must delete the current image with IWG_DeleteCell before using IWG_InitCellDat to insert the new image.

Cell Protection

The cell, by design is read-only and is used to display static text.

Editing

Since the cell is for static text only the End User has no editing options.

3.5.7.7 @IWGIPADDRESS

General

The @IWGIPADDRESS type cell is used when the End User needs to be able to enter an IP address.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGIPADDRESS, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Four numbers (0-255) separated by ".".
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: Four numbers (0-255) separated by ".".

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: Four numbers (0-255) separated by ".".

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

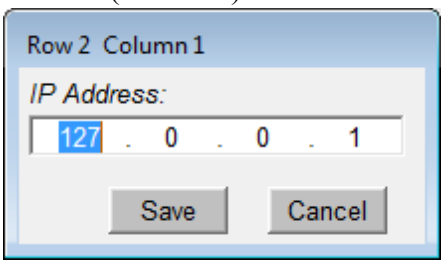
Editing

The End User may edit the contents of an @IWGIPADDRESS type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGIPADDRESS cell entry.

	A	B
1		
2	127.0.0.1	
3		

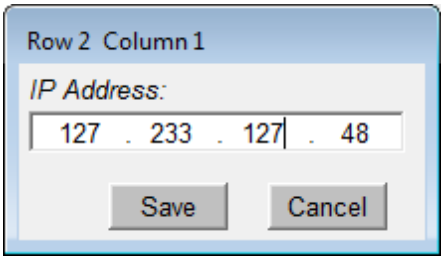
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text either by typing or ssing *CTRL-V* to paste text from the clipboard.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X*/*CTRL-V* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	127.233.127.48	
3		

3.5.7.8 @IWGLABEL

General

The @IWGLABEL type cell is used to display a non-editable, static text string.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGLABEL, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Any ASCII character between 0x20 and 0xFF is allowed.
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell, by design is read-only and is used to display static text.

Editing

Since the cell is for static text only the End User has no editing options.

3.5.7.9 @IWGMASK

General

The @IWGMASK type cell is used when the End User input needs to be restricted to the following:

1. A specific number of characters
2. A certain type of character
3. Static characters.
4. Any or all of the above.

Initialization

The cell is initialized with

IWGM_InitMaskCellDat(*win, id, r, c, text, mask, prompt, flags, intype*)

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
text	-	The initial text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the mask formatting or, are valid raw data characters depending on the setting of ***intype*** (below). **NOTE**: See critical instructions for text format following Valid Flag table below.
mask	-	The mask used to format raw data input. See Creating a Mask below.
prompt	-	The character used to indicate a blank entry in the input location in the mask.
flags	-	The optional styles to be applied to the cell's contents.
intype	-	Indicates the type of text in the text field above: 0 = raw, unformatted data 1 = formatted data that matched mask formatting

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are

		entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

When creating an @IWGMASK cell the *text* parameter may represent either raw data or formatted data depending upon the *intype* parameter setting. In both cases if the *text* parameter is null ("") or improperly formatted nothing will be displayed in the grid.

In both cases, *text* must account for all the formatted characters.
Let's say that *text* contains 123, *mask* = "(000) 000-0000", and *intype* is set for raw data. This would result in nothing being displayed in the grid. This is because it is treated as an error. The *mask* says it has 10 formatted characters. Therefore *text* has to contain 10 characters; either valid characters, prompt characters or a combination of both. So, in this example text would need to be '123_____' which would result in '(123) ___-____' being displayed in the grid.

If *intype* is set for formatted data we have to account for the same 10 formatted characters but also the static characters. This results in *text* looking like this: '(123) ___-____'

Improperly applying the above rules can result in improperly stored or lost data.

Reading

STRING = IWGM_GetCellFormatDat(*win, id, r, c*)

can be used to read the formatted contents of the specified @IWGMASK type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The formatted string contents of the cell at the specified position.

Example
```
STRING text = IWGM_GetCellFormatDat(main,  501, 22, 3)
```

STRING = IWGM_GetCellRawDat(*win, id, r, c*)

can be used to read the raw contents of the specified @IWGMASK type cell..

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The raw string contents of the cell at the specified position.

Example
```
STRING text = IWGM_GetCellRawDat(main,  501, 22, 3)
```

Updating

IWGM_SetCellFormatDat(*win, id, r, c, dat*)

can be used to load new formatted text into the @IWGMASK type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The new formatted text to be placed in the designated cell.

Example
```
IWGM_SetCellFormatDat(main,  501, 11, 2, "(123) 456-7890")
IWGM_SetCellFormatDat(main,  501, 11, 2, "(123) ___-___")
```

IWGM_SetCellRawDat(*win, id, r, c, dat*)

can be used to load new raw text into the @IWGMASK type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The new raw text to be placed in the designated cell.

Example

```
IWGM_SetCellRawDat(main,  501, 11, 2, "1234567890")
IWGM_SetCellRawDat(main,  501, 11, 2, "123_______")
```

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

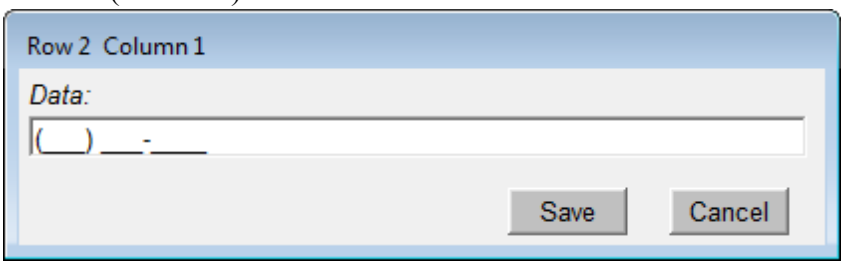
Editing

The End User may edit the contents of an @IWGMASK type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGMASK cell entry.

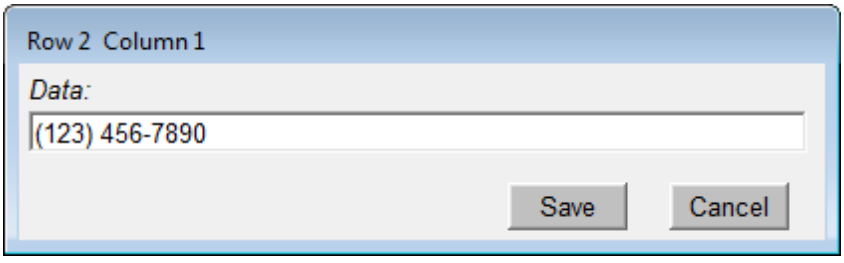
	A	B
1		
2	(___) ___-____	
3		

If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User can now fill in the data or click the *Cancel* button..

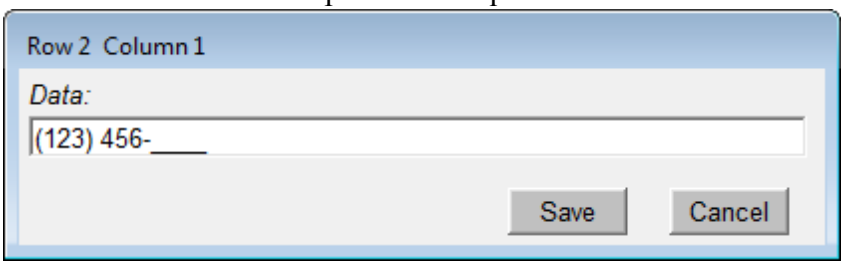
The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	(123) 456-7890	
3		

Had the End User entered part of the required entries like below



and *Saved* the partial edit then the displayed data would appear as below.

	A	B
1		
2	(123) 456-____	
3		

Creating a Mask

As previously stated the @IWGMASK type cell is used when the End User input needs to be restricted to a predefined format in the following ways:

1. A specific number of characters
2. A certain type of character
3. Static characters.
4. Any or all of the above.

The following table describes each of the mask formatting characters:

Character	Required Entry	Description
0	Y	The "0" (zero) character. 0-9 is allowed
9	N	The "9" (nine) character. 0-9 or a space is allowed
L	Y	The "L" character. A-Z or a-z is allowed.
$	N	The "L" character. A-Z or a-z or space is allowed
A	Y	The "A" character. A - Z or a-z or 0 - 9 is allowed
a	N	The "a" character. A - Z or a-z or 0 - 9 or a space is allowed
&	Y	The "&" character. Any character from ASC 33 to ASC 255, inclusive
C	N	The "C" character. Any character from ASC 32 to ASC 255, inclusive
" "	N	A space. A static blank space neither requiring or allowing any input.

">"	N/A	The ">" is used to modify an alpha mask character. When placed before a L,$,A,a,&, or C mask character the End User's input will be converted to upper case.
"<"	N/A	The "<" is used to modify an alpha mask character. When placed before a L,$,A,a,&, or C mask character the End User's input will be converted to lower case.
"\"	N/A	When the User desires to have one of the mask formatting characters above to appear as a static character instead of a formatting character the "\" character is placed before the formatting character.
See Description	N/A	Any other character then the above formatting characters that are inserted in a mask will be displayed as static characters

Following are some examples:

Mask: (000) 000-0000
Raw Data: 1234567890
Formatted Data: (123) 456-7890

Mask: 000-00-0000
Raw Data: 123456789
Formatted Data: 123-45-6789

Mask: \A\C>L-0-0
Raw Data: v34
Formatted Data: ACV-3-4

Mask: CCCCC**
Raw Data: 1 ^{>
Formatted Data: 1 ^{>**

Mask: \<00 00 00\>
Raw Data: 123456
Formatted Data: <12 34 56>

3.5.7.10 @IWGMONEY

General

The @IWGMONEY type cell is used when the End User input needs to be restricted to formatted currency data.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGMONEY, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Any ASCII character between 0x20 and 0xFF is allowed.
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Configuration

There are several options for configuring an @IWGMONEY type cell.

In the following discussions of the various configuration options it is assumed that the raw data is

```
-1234.56
```

and, with no configuration options selected the grid cell appears like the following:

	A	B
1		
2	$-1,234.56	
3		

IWG_SetCellDecPlaces(*win*, *id*, *r*, *c*, *dp*)

can be used to change the number of displayed decimal places in an @IWGMONEY type cell

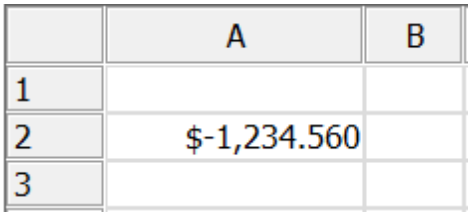
where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dp	-	Number of desired decimal places (0-9 max) Default = 2

Example

```
IWG_SetCellDecPlaces(main,  501, 2, 1,3)
```

results in the following



	A	B
1		
2	$-1,234.560	
3		

IWG_SetCellNegParenth(*win*, *id*, *r*, *c*, *pr*)

can be used to set how negative numbers are displayed in a @IWGMONEY type cell.

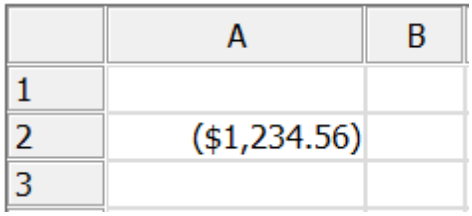
where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
pr	-	0 - Negative numbers are displayed as $-123.45. Default 1 - Negative numbers are displayed as ($123.45)

Example

```
IWG_SetCellNegParenth(main,  501, 2, 1,1)
```

results in the following



	A	B
1		
2	($1,234.56)	
3		

IWG_SetCellNoSymbol(*win*, *id*, *r*, *c*, *nosym*)

can be used to set whether the currency symbol is displayed or not in a @IWGMONEY type cell.

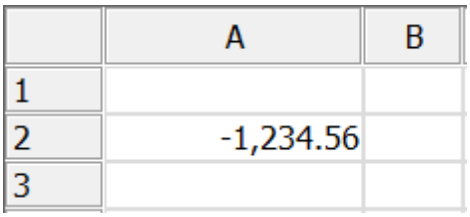
where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
nosym	-	0 - Currency symbol is displayed $123.45 Default 1 - Currency symbol is not displayed 123.45

Example

```
IWG_SetCellNoSymbol(main,  501, 2, 1, 1)
```

results in the following



	A	B
1		
2	-1,234.56	
3		

IWG_SetCellRoundDec(*win*, *id*, *r*, *c*, *round)*

can be used to set how excess decimal places are handled in a @IWGMONEY type cell.

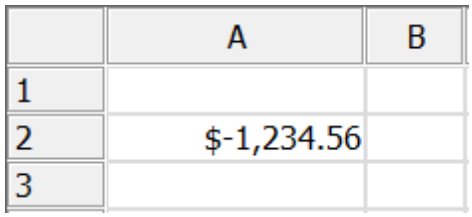
where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
round	-	Assume cell data is 1234.556 and decimal places is set to 2(default): 0 - Formatted data is truncated - results in 1234.55 (default) 1 - Formatted data is rounded - results in 1234.56

Example

```
IWG_SetCellRoundDec(main,  501, 2, 1, 1)
```

results in the following



	A	B
1		
2	$-1,234.56	
3		

IWG_SetSymbol(*win*, *id*, *r*, *c*, *sym*)

can be used to change the current symbol being used in @IWGMONEY type cells.

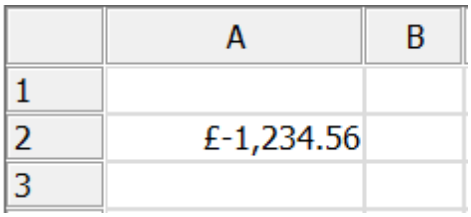
where

win		The parent of the grid control.
id		The id of this instance of the grid control.
r		The row position of the desired cell.
c		The column position of the desired cell
sym		A single ASCII character to be used. Default is "$"(US dollars)

Example

```
IWG_SetSymbol(main,  501, 2, 1,"£")
```

results in the following



	A	B
1		
2	£-1,234.56	
3		

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

IWG_GetSymbol(*win*, *id*, *r*, *c*)

can be used to get the current currency symbol configured for the specified @IWGMONEY type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value

The configured symbol for the specified cell. Default is "$" (US dollars)

Example

```
STRING sym = IWG_GetSymbol(main,  501, 2, 3)
```

IWG_GetCellRoundDec(*win*, *id*, *r*, *c*)

can be used to get the format of how excessive decimal digits are handled in the specified @IWGMONEY type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - Excessive decimal digits will be truncated. Default setting.
1 - Excessive decimal digits will be rounded..

Example
```
INT round = IWG_GetCellRoundDec(main,  501, 2, 3)
```

IWG_GetCellNoSymbol(*win*, *id*, *r*, *c*)

can be used to get the display format of the currency symbol for the specified @IWGMONEY type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - The currency symbol will be displayed Default setting.
1 - The currency symbol will not be displayed.

Example
```
INT no_symbol = IWG_GetCellNoSymbol(main,  501, 2, 3)
```

IWG_GetCellNegParenth(*win*, *id*, *r*, *c*)

can be used to get the negative number display format for the specified @IWGMONEY type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - Negative numbers are displayed as $-123.45 Default setting.
1 - Negative numbers are displayed as ($123.45)

Example
```
INT neg_format = IWG_GetCellNegParenth(main,  501, 2, 3)
```

INT = IWG_GetCellDecPlaces(*win*, *id*, *r*, *c*)

can be used to get the number of decimal places displayed in an @IWGMONEY type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
Number of decimal places that are displayed in the cell. Default is 2.

Example
```
INT dec_plc = IWG_GetCellDecPlaces(main,  501, 2, 3)
```

Updating
The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.

text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

Editing

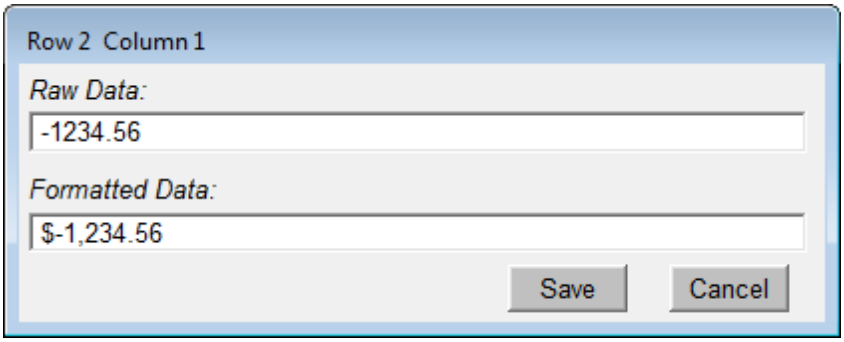
The End User may edit the contents of an @IWGMONEY type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGMONEY cell entry.

	A	B
1		
2	$-1,234.56	
3		

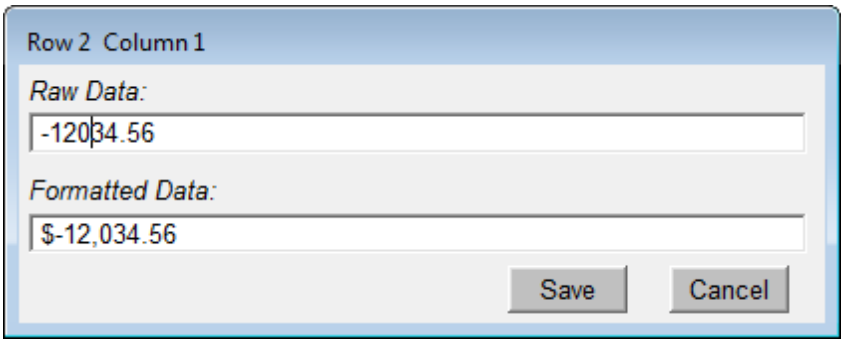
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).

The *Raw Data* is the actual numbers, stripped of any formatting. This field is editable by the End User.
The *Formatted Data* shows the currently displayed *Raw Data* with formatting applied. It will always reflect what will appear in the cell if the *Save* button is clicked. This field is read-only.



The End User has several options at this time:
1. Left-click anywhere in the *Raw Data* field at the desired edit point and insert text either by typing. Using *CTRL-V* to paste text from the clipboard is not allowed.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	$-12,034.56	
3		

3.5.7.11 @IWGNUMF

General

The @IWGNUMF type cell is used when the End User input needs to be restricted to floating point numbers.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGNUMF, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Allowable input is '0-9', '-' (provided it is at the start of the string), and '.' (provided it is the only one in the string and does not precede a "-")
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type. See above.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

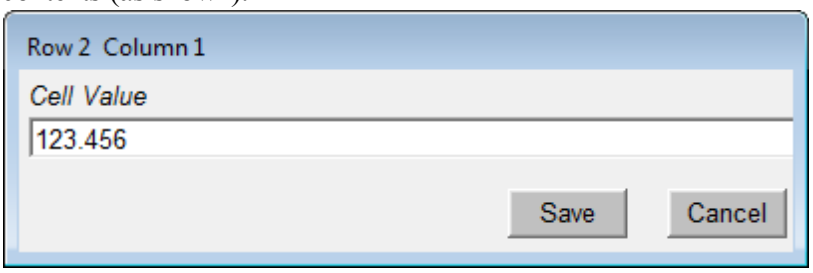
Editing

The End User may edit the contents of an @IWGNUMF type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGNUMF cell entry.

	A	B
1		
2	123.456	
3		

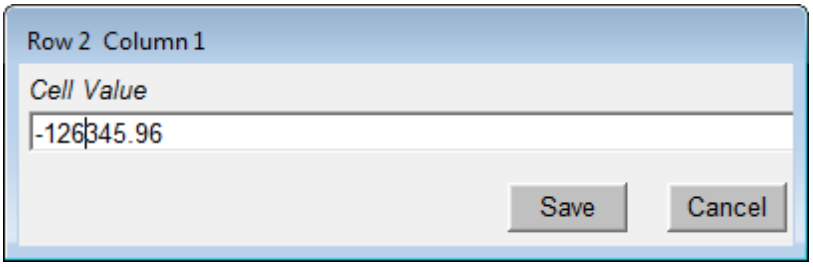
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text either by typing valid text. Using *CTRL-V* to paste text from the clipboard is not allowed.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	-126345.96	
3		

3.5.7.12 @IWGNUMI

General

The @IWGNUMI type cell is used when the End User input needs to be restricted to integer numbers.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGNUMI, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Allowable input is '0-9' and '-' (provided it is at the start of the string).
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

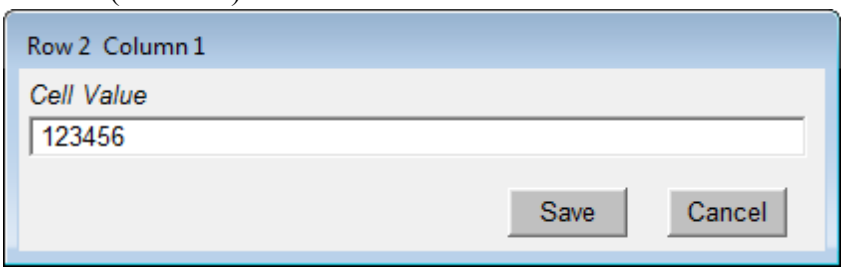
Editing

The End User may edit the contents of an @IWGNUMI type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGNUMI cell entry.

	A	B
1		
2	123456	
3		

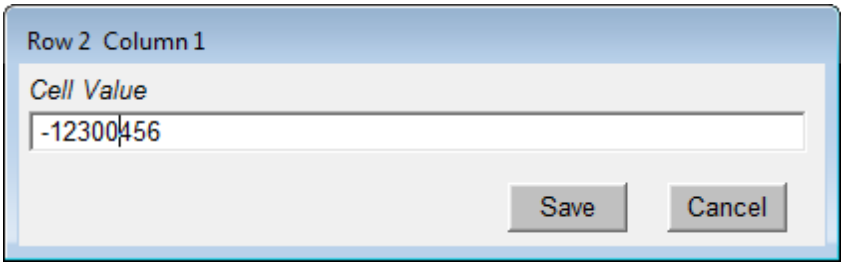
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text either by typing valid text. Using *CTRL-V* to paste text from the clipboard is not allowed.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	-12300456	
3		

3.5.7.13 @IWGRADIOBUTTON

General

The @IWGCHECK type cell is used when the End User needs to able to select an option, much like an IWBasic @CHECKBOX control..

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *state*, @IWGCHECK, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
state	-	The initial state of the checkbox. • "1" - Checked • "0" - UnChecked
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).

Example: The following two lines of code

```
IWG_InitCellDat(win,  ID_GRID1, 2, 1, "0",@IWGRADIOBUTTON,0)
IWG_InitCellDat(win,  ID_GRID1, 3, 1, "1",@IWGRADIOBUTTON,0)
```

results in the following grid entries:

	A	B
1		
2	○	
3	◉	
4		

Reading

The current state of the cell can be read with

state = IWG_GetCheckState(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.

and the current state is.

- 1 - Checked
- 0 - UnChecked

Updating

The current state of the cell can be changed with

IWG_SetCheckState(*win*, *id*, *r*, *c* , *s*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
s	-	The desired state. • 1 - Checked • 0 - UnChecked

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

Editing

The End User may change the state of the cell by left-clicking the cell. Each click will toggle the state of the currently selected cell. The following code shows how to group four @IWGRADIOBUTTON cells to operate the way a group of IWBasic @RadioButtons function.

```
    IWG_InitCellDat(win,  ID_GRID1, 2, 1, "1",@IWGRADIOBUTTON,0)
    IWG_InitCellDat(win,  ID_GRID1, 3, 1, "0",@IWGRADIOBUTTON,0)
    IWG_InitCellDat(win,  ID_GRID1, 4, 1, "0",@IWGRADIOBUTTON,0)
    IWG_InitCellDat(win,  ID_GRID1, 5, 1, "0",@IWGRADIOBUTTON,0)
......

SUB winHandler(),INT
   INT ROW,COL
   SELECT @MESSAGE
      CASE @IDCREATE
         centerwindow win
      case @idcontrol
         SELECT @controlid
            CASE ID_GRID1
               SELECT @NOTIFYCODE
                  CASE @IWGN_RADIOCHECKED
                     IWG_CellLocate(@lparam,row,col)
                     int k
                     for k=2 to 5
                        if k<>r
                           IWG_SetCheckState( win,  ID_GRID1,  k,  c ,  0)
                        else
                           IWG_SetCheckState( win,  ID_GRID1,  k,  c ,  1)
                        endif
                     next k
                  ENDSELECT
               ENDSELECT
            CASE @IDCLOSEWINDOW
               run = 0
      endselect
   return 0
endsub
```

3.5.7.14 @IWGSUBBUTTON

General

The @IWGSUBBUTTON type cell is used when the End User needs to enter some data and then press a dedicated button for some action to take place based upon their data entry. Each time the End User clicks the button in the cell a User defined subroutine is called, passing the cell contents to the subroutine. The subroutine then returns a string that is loaded into the cell.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGSUBBUTTON, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Any ASCII character between 0x20 and 0xFF is allowed.
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Configuration

IWG_SetCellButtonCB(*win*, *id*, *r*, *c*, *dat*)

is used to set the User subroutine to be called when the button is clicked in a @IWGSUBBUTTON type cell.

where

win	-	The parent of the grid control.

id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	Pointer to the User's subroutine

Remarks

The User's subroutine declaration must match the following template:

```
[GLOBAL] SUB mysub(STRING a), STRING
STRING ret
....
RETURN ret
ENDSUB
```

The returned string will become the new contents of the text portion of the cell.

Example

```
IWG_SetCellSub(main,  501, 10, 3, &test)

SUB test(STRING d),STRING
STRING ret
...
ret = "Made it!"
RETURN ret
ENDSUB
```

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

POINTER = IWG_GetCellButtonCB(*win*, *id*, *r*, *c*)

can be used to get the address of the callback subroutine associated with an @IWGSUBBUTTON type cell.

where

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
Address of callback subroutine.

Example

```
IWG_SetCellButtonCB(main,  501, 13, 4, &test2)
pointer ff = IWG_GetCellButtonCB(main,  501, 13, 4)
```

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.

r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

Editing

The End User may edit the contents of an @IWGSUBBUTTON type cell provided the grid is not protected AND the individual cell is not protected.

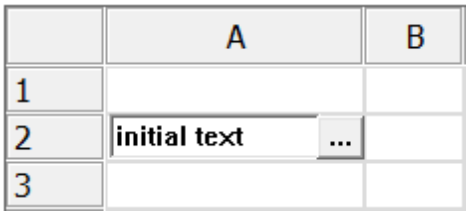
The following code is typical of a typical an @IWGSUBBUTTON cell entry.

```
   IWG_InitCellDat(win, ID_GRID1, 2, 1, "initial text",@IWGSUBBUTTON,
@IWG_CENTERALIGN)

   IWG_SetCellButtonCB(win, ID_GRID1, 2, 1, &test1)

sub test1(string d),string
   string ret
   '...
   ret = "["+ucase$(d)+"]"
   return ret
endsub
```

The following is the resulting grid entry

	A	B
1		
2	initial text	
3		

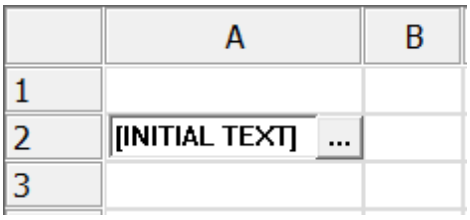
If the End User left-clicks the cell changes to an edit control and a button (as shown).



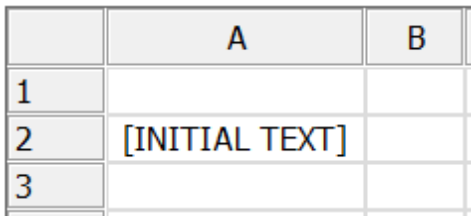
The End User has several options at this time:

Option 1

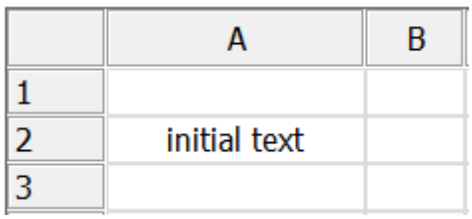
The End User can left-click the button. This results in the cell contents being passed to the User defined subroutine. The subroutine performs whatever actions are desired and then return a string which becomes the cells content(as shown below).



The End User can now accept the change by clicking on another cell in the grid which results in the cell now appearing as below



or, the End User can press *ESC* and the cell returns to its original state(shown below)



Option 2

The End User can do one of the following

1. Left-click anywhere in the edit control at the desired edit point and insert text either by typing or using CTRL-V to paste text from the clipboard.

2. Highlight a portion of the cell contents and use Delete/CTRL-C/CTRL-X/CTRL-V in their normal manner or simply type over the selected text.
3. Simply click another cell to end the editing.

The above would appear as follows

the original entry

	A	B
1		
2	initial text	
3		

left-clicking the cell then entering new text

	A	B
1		
2	new data ...	
3		

left-clicking the button results in

	A	B
1		
2	[NEW DATA] ...	
3		

and clicking another cell results in

	A	B
1		
2	[NEW DATA]	
3		

or, optionally cancelling the change with the ESC key resulting in

	A	B
1		
2	initial text	
3		

3.5.7.15 @IWGTXT

General

The @IWGTXT type cell is used when the End User input needs to be restricted to alpha characters..

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGTXT, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Allowable input is 'a-z', 'A-Z', " ", and ENTER key
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.

r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

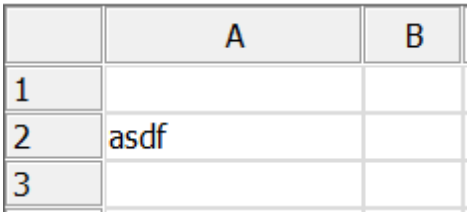
win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

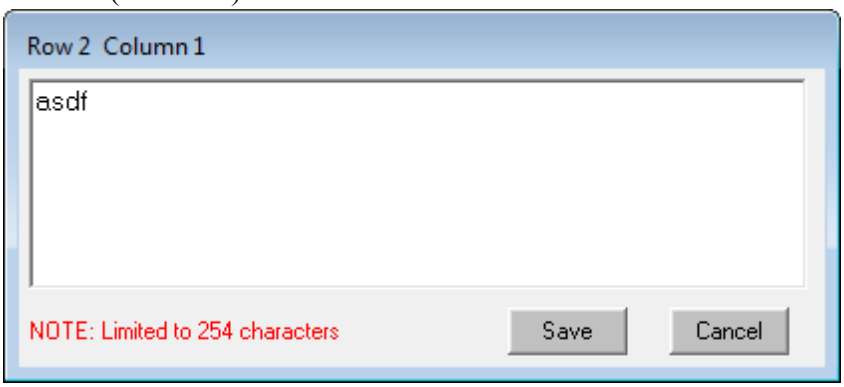
Editing

The End User may edit the contents of an @IWGTXT type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGTXT cell entry.



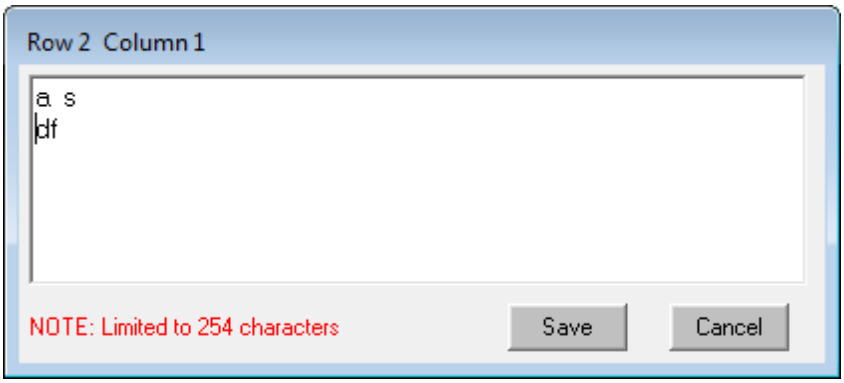
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text by typing. Using *CTRL-V* to paste text from the clipboard is not allowed.
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	a s	
3		

3.5.7.16 @IWGTXTNUM

General

The @IWGTXTNUM type cell is used when the End User input needs to be restricted to alphanumeric characters.

Initialization

The cell is initialized with

IWG_InitCellDat(*win*, *id*, *r*, *c*, *text*, @IWGTXTNUM, *flags*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The initial text to place in the cell. Allowable input is 'a-z', 'A-Z', '0-9', '-', " ", and ENTER key
flags	-	The optional styles to be applied to the cell's contents. See the table below.

Valid flags:

@IWG_PROTECTCELL	-	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	-	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	-	Aligns the displayed text to the right.
@IWG_CENTERALIGN	-	Aligns the displayed text to the center.
@IWG_CASE_ANY	-	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	-	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	-	All alpha characters are converted to lower case before displaying.

Reading

The cell contents can be read with

text = IWG_GetCellDat(*win*, *id*, *r*, *c*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The current text in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Updating

The cell is updated with

IWG_SetCellDat(*win*, *id*, *r*, *c*, *text*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being configured.
text	-	The new text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.

Cell Protection

The cell's protection can be can be turned on/off with

IWG_SetCellProt(*win*, *id*, *r*, *c*, *p*)

where

win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

The individual cell's current state of protection can be read with

state = IWG_GetCellProt(*win*, *id*, *r*, *c*)

where

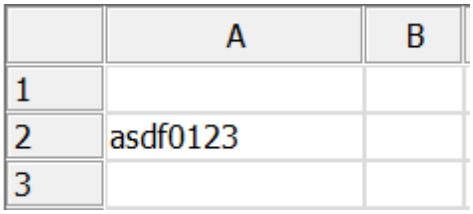
win,id	-	The parent and instance of the grid control; like any other IWBasic control.
r,c	-	The row and column position of the cell being modified.

and state = 1 for protected and 0 for unprotected. Result does not reflect current state of overall grid protection.

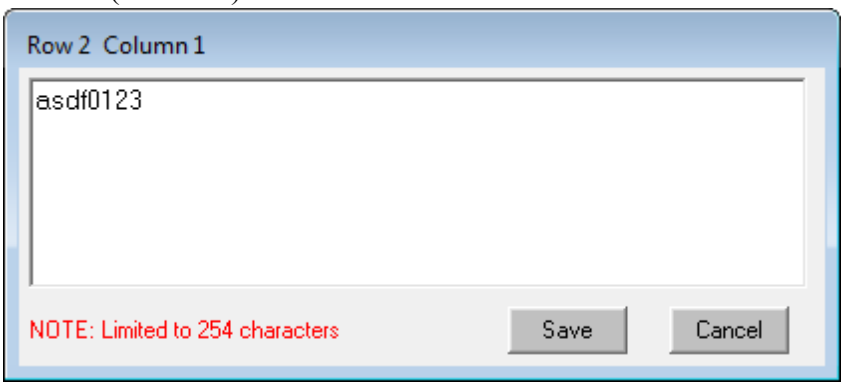
Editing

The End User may edit the contents of an @IWGTXTNUM type cell provided the grid is not protected AND the individual cell is not protected.

The following is an example of a typical @IWGTXTNUM cell entry.



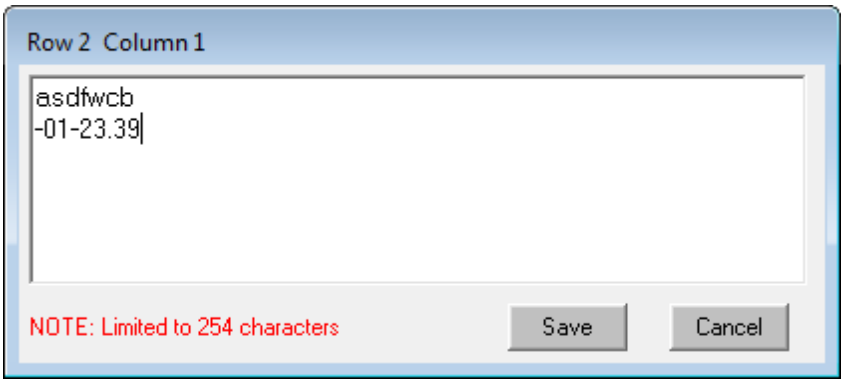
If the End User left-clicks the cell the following dialog will open displaying the cells entire contents (as shown).



The End User has several options at this time:

1. Left-click anywhere in the window at the desired edit point and insert text either by typing. Using *CTRL-V* to paste text from the clipboard. is not allowed
2. Highlight a portion of the cell contents and use *Delete*/*CTRL-C*/*CTRL-X* in their normal manner or simply type over the selected text.
3. Simply click the *Cancel* button.

The above has been edited as shown.



Clicking the *Save* button saves the edits, closes the dialog, and updates the displayed grid cell, shown below.

	A	B
1		
2	asdfwcb	
3		

Commands

Part IV

4 Commands

4.1 IWG_AutoAddRow

Syntax

IWG_AutoAddRow(WINDOW | DIALOG ***win***, UINT ***id***, INT ***ar***)

Description
Automatically increases the number of rows in the grid when a cell is added that is outside the current bounds.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
ar	-	1 - turns feature ON (default), 0 - turns feature oOFF

Return value
None

Remarks
See Also:

Example usage
```
IWG_AutoAddRow(main, 501, 1)
```

4.2 IWG_AutoNumberHeader

Syntax

IWG_AutoNumberHeader(WINDOW | DIALOG ***win***, UINT ***id***, INT ***num***)

Description
Controls the auto-numbering of the header row columns.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
num	-	1 - columns are numbered (default), 0 - User entered labels are used if any are entered. Both depend on the setting of an internal flag controlled by IWG_ShowHeader

Return value

None

Remarks
See Also: IWG_ShowHeader

Example usage
```
IWG_AutoNumberHeader(main,  501, 1)
```

4.3 IWG_CellLocate

Syntax

IWG_CellLocate(*(*UINT ***lparam,*** INT ***row*** *BYREF*, INT ***col*** *BYREF)*)

Description
Utility to convert the contents of @LPARAM to its row and column components

Parameters

lparam	-	The value passed in a notification message sent by IWGrid to the parent window.
row	-	The predefined variable to hold the decoded row value..
col	-	The predefined variable to hold the decoded column value..

Return value
The currently selected cell's row and column coordinates when the notification message was sent.

Remarks
See Also: Notification Messages

Example usage
```
SUB mainHandler(),INT
   int r,c
   SELECT @MESSAGE
      CASE @IDCONTROL
         SELECT @controlid
            CASE 501
               SELECT @notifycode
                  CASE @IWGN_CELLCLICKED
                     IWG_CellLocate(@LPARAM,r,c)
                     PRINT "@IWGN_CELLCLICKED ",r,c
               ENDSELECT
         ENDSELECT
   ...
```

4.4 IWG_ColResizing

Syntax

IWG_ColResizing(WINDOW | DIALOG ***win***, UINT ***id***, INT resize)

Description
Controls User's ability to resize column widths.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
resize	-	1 - columns can be resized by User 0 - columns can not be resized by User (default),

Return value
None

Remarks
See Also:

Example usage

```
IWG_ColResizing(main,  501, 1)
```

4.5 IWG_Create

Syntax

IWG_Create(WINDOW | DIALOG ***win***, STRING ***caption***, INT ***l***, INT ***t***, INT ***w***, INT ***h***, UINT ***id***)

Description
Creates an instance of the grid.

Parameters

win	-	The parent of the grid control.
caption	-	The displayed title of the grid. Multiple lines can be accomplished by inserting "\n". Each line will be centered in the grid window.
l	-	The left edge of the grid control in pixels relative to the left edge of the parent window's client area.
t	-	The top edge of the grid control in pixels relative to the top edge of the parent window's client area.
w	-	The width of the grid control in pixels.
h	-	The height of the grid control in pixels.

id	-	The id of this instance of the grid control.

Return value
None

Remarks
See Also:

Example usage
```
IWG_Create(main, "Custom Grid Control\nfor IWBasic", 20,30,1200,600, 501)
```

4.6 IWG_DeleteAllCells

Syntax

IWG_DeleteAllCells(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Clears all cell entries in the grid

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value
None

Remarks
See Also:

Example usage
```
IWG_DeleteAllCells(main,  501)
```

4.7 IWG_DeleteCell

Syntax

IWG_DeleteCell(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Deletes the specified cell

Parameters

win	-	The parent of the grid control.

id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell.

Return value
None

Remarks
See Also:

Example usage
```
IWG_DeleteCell(main,  501, 4, 11)
```

4.8 IWG_EnableEditDialog

Syntax

IWG_EnableEditDialog(WINDOW | DIALOG ***win***, UINT ***id***, INT en)

Description
Used to prevent all cells that have edit dialogs from opening the dialog..

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
en	-	0 - Disables the opening of edit dialogs for grid cells. 1 - Enables the opening of edit dialogs for grid cells.- DEFAULT

Return value
None

Remarks
See Also:

Example usage
```
IWG_EnableEditDialog(main,501,0).
```

4.9 IWG_GetCellBGColor

Syntax

UINT = **IWG_GetCellBGColor**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description

Used to get the background color of the specified cell..

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The cell background color.

Remarks
None

Example usage
```
uint bgc = IWG_GetCellBGColor(main,  501, 2, 2)
```

4.10 IWG_GetCellButtonCB

Syntax

POINTER = **IWG_GetCellButtonCB**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description
Used to get the address of the callback subroutine associated with @IWGSUBBUTTON type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
Address of callback subroutine.

Remarks
See Also:

Example usage
```
IWG_SetCellButtonCB(main,  501, 13, 4, &test2)
pointer ff = IWG_GetCellButtonCB(main,  501, 13, 4)
```

4.11 IWG_GetCellButtonSize

Syntax

IWG_GetCellButtonSize(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, INT ***w*** BYREF **,** INT ***h*** BYREF)

Description
Sets the width and height of the button in the specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
w	-	Predefined variable to hold button width
h	-	Predefined variable to hold button height

Return value
Width and height of the selected button. If both values are 0 the button size is the same as the cell size

Remarks
Applies only to @IWGBUTTON type cells.

Example usage

```
INT w,h
IWG_GetCellButtonSize(main,  501, 2, 2, w, h)
```

4.12 IWG_GetCellDat

Syntax

STRING = **IWG_GetCellDat**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to read the contents of the specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The string contents of the cell at the specified position.

Remarks
Use IWG_GetCheckState for @IWGCHECK type cells
Use IWGM_GetCellFormatDat or IWGM_GetCellRawDat for @IWGMASK type cells

Example usage
```
STRING text = IWG_GetCellDat(main,  501, 22, 3)
```

4.13 IWG_GetCellDecPlaces

Syntax

INT = **IWG_GetCellDecPlaces**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description
Used to get the number of decimal places displayed in an @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
Number of decimal places that are displayed in the cell. Default is 2.

Remarks
See Also:

Example usage
```
INT dec_plc = IWG_GetCellDecPlaces(main,  501, 2, 3)
```

4.14 IWG_GetCellFGColor

Syntax

UINT = **IWG_GetCellFGColor**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to get the foreground color of the specified cell..

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The cell foreground color.

Remarks
None

Example usage
```
uint bgc = IWG_GetCellFGColor(main,  501, 2, 2)
```

4.15 IWG_GetCellImageMissingText

Syntax

STRING = **IWG_GetCellImageMissingText**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to read the "missing image" text of the specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The "missing image"string contents of the cell at the specified position.

Remarks
Applies only to @IWGIMAGE type cells.

Example usage
```
STRING text = IWG_GetCellImageMissingText(main,  501, 22, 3)
```

4.16 IWG_GetCellImgMissingBGColor

Syntax

UINT = **IWG_GetCellImgMissingBGColor**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to get the background color of the specified @IWGIMAGE type cell when the image is missing.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The background color the @IWGIMAGE type cell when the image is missing..

Remarks
None

Example usage
```
uint mibgc = IWG_GetCellImgMissingBGColor(main,  501, 2, 2)
```

4.17 IWG_GetCellImgMissingFGColor

Syntax

UINT = **IWG_GetCellImgMissingFGColor**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to get the foreground color of the specified @IWGIMAGE type cell when the image is missing.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The foreground color the @IWGIMAGE type cell when the image is missing..

Remarks
None

Example usage
```
uint mifgc = IWG_GetCellImgMissingFGColor(main,  501, 2, 2)
```

4.18 IWG_GetCellImageSize

Syntax

IWG_GetCellImageSize(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, INT ***w*** BYREF**,** INT ***h*** BYREF)

Description
Sets the width and height of the image in the specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
w	-	Predefined variable to hold image width
h	-	Predefined variable to hold image height

Return value
Width and height of the selected image. If both values are 0 the image size is the same as the cell size

Remarks
Applies only to @IWGIMAGE type cells.

Example usage
```
INT w,h
IWG_GetCellImageSize(main,  501, 2, 2, w, h)
```

4.19 IWG_GetCellNegParenth

Syntax

INT = **IWG_GetCellNegParenth**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description

Used to get the negative number display format for the specified @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - Negative numbers are displayed as $-123.45 Default setting.
1 - Negative numbers are displayed as ($123.45)

Remarks
See Also:

Example usage
```
INT neg_format = IWG_GetCellNegParenth(main,  501, 2, 3)
```

4.20 IWG_GetCellNoSymbol

Syntax

INT = **IWG_GetCellNoSymbol**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description
Used to get the display format of the currency symbol for the specified @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - The currency symbol will be displayed Default setting.
1 - The currency symbol will not be displayed.

Remarks
See Also:

Example usage
```
INT no_symbol = IWG_GetCellNoSymbol(main,  501, 2, 3)
```

4.21 IWG_GetCellProt

Syntax

INT = **IWG_GetCellProt**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to determine if the specified cell.can be edited.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected.
1 - Cell contents can not be edited.

Remarks
See Also: IWG_SetGridProt

Example usage

```
INT prot = IWG_GetCellProt(main,501, 2, 3)
```

4.22 IWG_GetCellRoundDec

Syntax

INT = **IWG_GetCellRoundDec**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description
Used to get the format of how excessive decimal digits are handled in the specified @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 - Excessive decimal digits will be truncated. Default setting.
1 - Excessive decimal digits will be rounded..

Remarks
See Also:

Example usage
```
INT round = IWG_GetCellRoundDec(main,  501, 2, 3)
```

4.23 IWG_GetCellType

Syntax

INT = **IWG_GetCellType**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description
Used to get the type cell.at the specified location

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
0 or one of the followingcell type constants
@IWGANY
@IWGTXTNUM
@IWGTXT
@IWGNUMI
@IWGNUMF
@IWGHEX
@IWGCOMBO
@IWGCHECK
@IWGSUBBUTTON
@IWGMASK
@IWGMONEY
@IWGLABEL
@IWGIMAGE
@IWGIPADDRESS

Remarks
See Also:

Example usage
```
INT typ = IWG_GetCellType(main,  501, 2, 3)
```

4.24 IWG_GetCheckState

Syntax

INT = **IWG_GetCheckState**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to read the state of the specified @IWGCECK cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The current state of the cell at the specified position.

- 1 - Checked
- 0 - UnChecked

Remarks
See Also:

Example usage

```
INT state = IWG_GetCheckState(main,  501, 22, 3)
```

4.25 IWG_GetColumns

Syntax

INT = **IWG_GetColumns**(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to get the number of columns in the grid.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value
Number of columns

Remarks
See Also:

Example usage
```
INT num_cols = IWG_GetColumns(main,  501)
```

4.26 IWG_GetColumnWidth

Syntax

INT = **IWG_GetColumnWidth**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***c***)

Description
Used to get the width of the specified column..

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
c	-	The column position of the desired cell

Return value
Width of column in pixels

Remarks
See Also:

Example usage
```
INT col_width = IWG_GetColumnWidth(main,  501, 3)
```

4.27 IWG_GetCurrentCol

Syntax

INT = **IWG_GetCurrentCol**(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to get the current column.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value
Current column.

Remarks
See Also:

Example usage

```
INT current_col = IWG_GetCurrentCol(main,  501)
```

4.28 IWG_GetCurrentRow

Syntax

INT = **IWG_GetCurrentRow**(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to get the current row

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value
Current row

Remarks
See Also:

Example usage

```
INT current_row = IWG_GetCurrentRow(main,  501)
```

4.29 IWG_GetDim

Syntax

IWG_GetDim(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r*** byref, INT ***c*** byref)

Description
Used to get the number of rows and columns currently in the grid.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The returned number of rows in the grid
c	-	The returned number of columns in the grid

Return value
The number of rows and columns currently in the grid

Remarks

See Also:

Example usage

```
int rows,cols
IWG_GetDim(main,  501, rows, cols)
```

4.30 IWG_GetHeaderRowHeight

Syntax

INT = **IWG_GetHeaderRowHeight**(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to get the height of the header row.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value
The height of the header row in pixels.

Remarks
See Also:

Example usage

```
int header_height = IWG_GetHeaderRowHeight(main,  501)
```

4.31 IWG_GetImageSize

Syntax

IWG_GetImageSize(***(***STRING ***filename,*** INT ***w*** *BYREF*, INT ***h*** *BYREF)*)

Description
Utility to obtain the dimensions of a bmp, jpg, or gif image.

Parameters

filename	-	The value passed in a notification message sent by IWGrid to the parent window.
w	-	The predefined variable to hold the width value..
h	-	The predefined variable to hold the height value..

Return value

The dimensions of the specified image in pixels.

Remarks
None

Example usage
```
int w,h
string text = "image1.jpg"
IWG_GetImageSize(text, w, h)
```

4.32 IWG_GetRowHeight

Syntax

INT = **IWG_GetRowHeight**(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to get the height of the rows in the grid.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value
The height of the rows in pixels

Remarks
See Also:

Example usage
```
int rows_height = IWG_GetRowHeight(main, 501)
```

4.33 IWG_GetRows

Syntax

INT = **IWG_GetRows**(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to get the number of rows in the grid.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

Return value

The number of rows in the grid

Remarks
See Also:

Example usage
```
int num_rows = IWG_GetRows(main, 501)
```

4.34 IWG_GetSymbol

Syntax

STRING = **IWG_GetSymbol**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c***)

Description
Used to get the current currency symbol configured for the specified @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The configured symbol for the specified cell. Default is "$" (US dollars)

Remarks
See Also:

Example usage
```
STRING sym = IWG_GetSymbol(main,  501, 2, 3)
```

4.35 IWG_InitCellData

Syntax

IWG_InitCellDat(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***text***, OPT INT ***celltype***=1, OPT INT ***flags***=0)

Description
Used to set the initial basic configuration for a specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

r	-	The row position of the desired cell.
c	-	The column position of the desired cell
text	-	The initial text to place in the cell. For a @IWGIMAGE, the text will be in one of two forms: 1) The full path name of the image file 2) If the image is being loaded from the resource file the entry will be a string containing a "#" followed by the string equivalent of the resource ID number. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.
celltype	-	The optional designation for the type of permissible input to the cell. The default is @IWGANY (See Remarks below)
flags	-	The optional styles to be applied to the cell's contents.

Return value
None

Remarks
Valid cell types:

@IWGLABEL	-	Static text that can not be selected nor edited.
@IWGANY	-	Any ASCII character between 0x20 and 0xFF is allowed
@IWGTXTNUM	-	A-Z, a-z, 0-9, '-' , and '.'
@IWGTXT	-	A-Z, a-z
@IWGNUMI	-	0-9 and '-'
@IWGNUMF	-	0-9, '-', and '.'
@IWGHEX	-	0-9, A-F, a-f
@IWGCOMBO	-	Any ASCII character between 0x20 and 0xFF
@IWGCHECK	-	A checked or unchecked checkbox
@IWGSUBBUTTON	-	A text field that accepts any ASCII character between 0x20 and 0xFF and a button. Clicking the button passes the edit field to a User defined subroutine. The output of the subroutine returns a string which is placed in the edit field.
@IWGMASK	-	Allows a User defined mask to format the input data. NOTE: The IWG_InitMaskCellDat command should be used for initializing this type of cell.
@IWGMONEY	-	0-9, '-', and '.'. The input is automatically formatted to optionally include a currency symbol. Also thousand separators(,) are inserted at the proper locations and decimals are rounded/truncated to the proper number of decimal places.
@IWGIMAGE	-	A scalable .bmp, .jpg, or .gif image

@IWGBUTTON	-	A button that looks and acts much like a normal IWBasic @BUTTON control.
@IWGIPADDRESS	-	An IP address.

Valid flags:

@IWG_PROTECTCELL	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	Aligns the displayed text to the left. Default. Does not apply to @IWGCOMBO, @IWGCHECK, and @IWGSUBBUTTON cell types.
@IWG_RIGHTALIGN	Aligns the displayed text to the right. Does not apply to @IWGCOMBO, @IWGCHECK, and @IWGSUBBUTTON cell types.
@IWG_CENTERALIGN	Aligns the displayed text to the center. Does not apply to @IWGCOMBO, @IWGCHECK, and @IWGSUBBUTTON cell types.
@IWG_CASE_ANY	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	All alpha characters are converted to lower case before displaying.
@IWG_NOSCALE	Used only with @IWGIMAGE type cells When the flag is not used the image will be displayed full size. If the image is smaller than the cell size the image will be centered in the cell. If either dimension exceeds the cell size the image with be reduce in either or both directions to fit inside the cell. When the flag is used the image will be stretched/compressed in both directions to fill the cell. Note: The. @IWG_RATIO flag can impact the above.
@IWG_RATIO	Used only with @IWGIMAGE type cells When the flag is not used the size of the displayed image is controlled entirely by the @IWG_NOSCALE flag. When the flag is used the image will be adjusted to insure the aspect ration of the image is maintained after the adjustments caused by the state of the @IWG_NOSCALE have been applied.

Example usage

```
IWG_InitCellDat(main,  501, 1, 1, getstartpath+"bug1.bmp",@IWGIMAGE,@IWG_NOSCALE)
IWG_InitCellDat(main,  501, 3, 1, getstartpath+"back.jpg",@IWGIMAGE,@IWG_RATIO)


IWG_InitCellDat(main,  501, 2, 2, "Any character",@IWGLABEL,@IWG_RIGHTALIGN)
IWG_InitCellDat(main,  501, 2, 3, "A/a/(d)",@IWGANY,@IWG_CASE_UPPER|@IWG_CENTERALIGN)
```

```
IWG_InitCellDat(main,  501, 3, 2, "Alpha-Numeric",@IWGLABEL,@IWG_CENTERALIGN)
IWG_InitCellDat(main,  501, 3, 3, "ABC123",@IWGTXTNUM,@IWG_CASE_LOWER|@IWG_LEFTALIGN)

IWG_InitCellDat(main,  501, 4, 2, "Alpha",@IWGLABEL,@IWG_LEFTALIGN)
IWG_InitCellDat(main,  501, 4, 3, "ABC",@IWGTXT,@IWG_CASE_LOWER|@IWG_RIGHTALIGN)

IWG_InitCellDat(main,  501, 5, 2, "Numbers w/decimal",@IWGLABEL)
IWG_InitCellDat(main,  501, 5, 3, "101.3",@IWGNUMF)

IWG_InitCellDat(main,  501, 6, 2, "Integer Numbers only",@IWGLABEL)
IWG_InitCellDat(main,  501, 6, 3, "234",@IWGNUMI)


IWG_InitCellDat(main,  501, 7, 2, "Combobox",@IWGLABEL)
IWG_InitCellDat(main,  501, 7, 3, "Test Entry# 11",@IWGCOMBO)

IWG_InitCellDat(main,  501, 8, 2, "Checkbox",@IWGLABEL)
IWG_InitCellDat(main,  501, 8, 3, "1",@IWGCHECK)

IWG_InitCellDat(main,  501, 10, 2, "Edit/Button combo",@IWGLABEL)
IWG_InitCellDat(main,  501, 10, 3, "file name",@IWGSUBBUTTON)

IWG_InitCellDat(main,  501, 11, 2, "Hex Numbers",@IWGLABEL)
IWG_InitCellDat(main,  501, 11, 3, "101FF",@IWGHEX, @IWG_CASE_UPPER|@IWG_CENTERALIGN)


IWG_InitCellDat(main,  501, 12, 2, "Currency Numbers",@IWGANY,@IWG_PROTECTCELL)
IWG_InitCellDat(main,  501, 12, 3, "$4,321,234,101.899",@IWGMONEY)
```

4.36 IWG_SetCellButtonCB

Syntax

IWG_SetCellButtonCB(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, POINTER ***dat***)

Description

Sets the User subroutine to be called when the button is clicked in a @IWGSUBBUTTON type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	Pointer to the User's subroutine

Return value

None

Remarks

The User's subroutine declaration must match the following template:

```
[GLOBAL] SUB mysub(STRING a), STRING
STRING ret
....
RETURN ret
ENDSUB
```

The returned string will become the new contents of the text portion of the cell.

Example usage

```
IWG_SetCellSub(main,  501, 10, 3, &test)

SUB test(STRING d),STRING
STRING ret
...
ret = "Made it!"
RETURN ret
ENDSUB
```

4.37 IWG_SetCellButtonSize

Syntax

IWG_SetCellButtonSize(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, UINT ***w,*** UINT ***h***)

Description
Sets the width and height of the button in the specified in an @IWGBUTTON type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
w	-	Desired button width
h	-	Desired button height

Return value
None

Remarks
Applies only to @IWGBUTTON type cells.
If this command is not used (or both w and h are set to 0) the size of the button will be automatically adjusted to fill the cell.
If desired width or height exceeds the actual cell size the size, when applied, it will be limited to cell size.

Example usage

```
IWG_SetCellButtonSize(main,  501, 2, 2, 80, 20)
```

4.38 IWG_SetCellColor

Syntax

IWG_SetCellColor(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, UINT ***fg,*** UINT ***bg***)

Description
Sets the text and background colors for the specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
fg	-	Foreground (text) color default = RGB(0,0,0)
bg	-	Background color default = RGB(0,0,0)

Return value
None

Remarks
See Grid Cells > Color

Example usage

```
IWG_SetCellColor(main,  501, 2, 2, RGB(255,0,0), RGB(0,0,255))
```

4.39 IWG_SetCellCombo

Syntax

IWG_SetCellCombo(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***dat***)

Description
Used to load the list portion of a combobox.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	String containing all the entries to be added to the combobox with each item separated

		by a "|".

Return value
None

Remarks
The maximum allowable size of the passed string is 10,000 characters. If a string larger than that is passed the extra text will be truncated.

Example usage
```
istring g[10000]=""
int xx
for xx=1 to 700
   g += "Test Entry#"+str$(xx)+"|"
next xx
IWG_SetCellCombo(main,  501, 3, 4, g)
```

4.40 IWG_SetCellDat

Syntax

IWG_SetCellDat(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***dat***)

Description
Loads new text into the cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The new text to be placed in the designated cell.

Return value
None

Remarks
Can not be used with a @IWGIMAGE type cell.
Use IWG_SetCheckState for @IWGCHECK type cells
Use IWGM_SetCellFormatDat or IWGM_SetCellRawDat for @IWGMASK type cells

Example usage
```
IWG_SetCellDat(main,  501, 11, 2, "New Text")
```

4.41 IWG_SetCellDecPlaces

Syntax

IWG_SetCellDecPlaces(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c,*** INT ***dp***)

Description
Used to set the number of decimal places in a @IWGMONEY type cell to a new value.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dp	-	Number of desired decimal places (0-9 max)

Return value
None

Remarks
See Also:

Example usage

```
IWG_SetCellDecPlaces(main,  501, 2, 3,3)
```

4.42 IWG_SetCellImageMissingText

Syntax

IWG_SetCellImageMissingText(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***dat***)

Description
Sets the "missing image" text for the selected cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The text to be placed in the designated cell when the image is missing.

Return value
None

Remarks
Applies only to @IWGIMAGE type cells.

Example usage
```
IWG_SetCellImageMissingText(main,  501, 11, 2, "No image Available")
```

4.43 IWG_SetCellImgMissingColor

Syntax

IWG_SetCellImgMissingColor(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, UINT ***fg,*** UINT ***bg***)

Description
Sets the text and background colors for the specified @IWGIMAGE type cell for when the image is missing.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
fg	-	Foreground (text) color default = 0x000000
bg	-	Background color default = 0x000000

Return value
None

Remarks
See Grid Cells > Color

Example usage
```
IWG_SetCellImgMissingColor(main,  501, 2, 2, 0xff0000, 0x0000ff)
```

4.44 IWG_SetCellImageSize

Syntax

IWG_SetCellImageSize(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, UINT ***w,*** UINT ***h***)

Description

Sets the width and height of the image in the specified cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
w	-	Desired image width
h	-	Desired image height

Return value
None

Remarks
Applies only to @IWGIMAGE type cells.
If this command is not used (or both w and h are set to 0) the size of the image will be automatically adjusted according to the settings of the @IWGNOSCALE and @IWGRATIO flags..

Example usage

```
IWG_SetCellImageSize(main,  501, 2, 2, 80, 20)
```

4.45 IWG_SetCellNegParenth

Syntax

IWG_SetCellNegParenth(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c,*** INT ***pr***)

Description
Used to set how negative numbers are displayed in a @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
pr	-	0 - Negative numbers are displayed as $-123.45 1 - Negative numbers are displayed as ($123.45)

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetCellNegParenth(main,  501, 2, 3,1)
```

4.46 IWG_SetCellNoSymbol

Syntax

IWG_SetCellNoSymbol(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c,*** INT ***nosym***)

Description
Used to set whether the currency symbol is displayed or not in a @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
nosym	-	0 - Currency symbol is displayed $123.45 1 - Currency symbol is not displayed 123.45

Return value
None

Remarks
See Also: IWG_SetSymbol

Example usage
```
IWG_SetCellNoSymbol(main,  501, 2, 3, 1)
```

4.47 IWG_SetCellProt

Syntax

IWG_SetCellProt(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, INT ***p***)

Description
Controls the editability of the specified cell.

Parameters

win	-	The parent of the grid control.

id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
p	-	0 - Cell contents can be edited. NOTE: Overridden if the entire grid is protected. 1 - Cell contents can not be edited.

Return value
None

Remarks
See Also: IWG_SetGridProt

Example usage
```
IWG_SetCellProt(main,501, 2, 3, 1)
```

4.48 IWG_SetCellRoundDec

Syntax

IWG_SetCellRoundDec(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c,*** INT ***round***)

Description
Used to set how excess decimal places are handled in a @IWGMONEY type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
round	-	Assume cell data is 123.456 and decimal places is set to 2(default): 0 - Formatted data is truncated - results in 123.45 1 - Formatted data is rounded -results in 123.46

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetCellRoundDec(main,  501, 2, 3,1)
```

4.49 IWG_SetCheckState

Syntax

IWG_SetCheckState(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, INT ***state***)

Description
Sets the state of the specified @IWGCHECK cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
state	-	• 1 - Set state to Checked • 0 - Set state to Un-Checked

Return value
None

Remarks
None

Example usage
```
IWG_SetCheckState(main,  501, 11, 2, 1)
```

4.50 IWG_SetColumnAutoWidth

Syntax

IWG_SetColumnAutoWidth(WINDOW | DIALOG ***win***, UINT ***id***, INT ***aw***)

Description
Automatically sets the column width to accommodate the longest text in the column

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
aw	-	1 - Turns the feature ON 0 - Turns the feature OFF (default)

Return value
None

Remarks

Must be used before any IWG_InitCellDat commands are executed.
When ON the header row text is shown on one line, with ellipses if required, even if the text contains "\n".
When OFF, the header row text will be displayed in multiple rows if it contains "\n" . Column width is determined by the header text width.

Example usage

```
IWG_SetColumnAutoWidth(main,501, 0)
```

4.51 IWG_SetColWidth

Syntax

IWG_SetColWidth(WINDOW | DIALOG ***win***, UINT ***id***, INT ***c***, INT ***w***)

Description
Sets the default width of an individual column.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
c	-	The column to be changed
w	-	The desired width in pixels. Default is 50

Return value
None

Remarks
See Also:

Example usage

```
IWG_SetColWidth(main,501, 2, 250)
```

4.52 IWG_SetCursorPosition

Syntax

IWG_SetCursorPosition(WINDOW | DIALOG ***win***, UINT ***id***, INT r, INT c)

Description
Sets the grids current cell position and draws the cursor box.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
None

Remarks
If the grid does not have focus there will be no indication unless the IWG_SetHiLightRow feature is turned ON.
If the grid does have focus the designated cell will be hilighted.
Has no impact on mouse cursor.

Example usage

```
IWG_SetCursorPosition(main, 501, 3, 4)
SETFOCUS (main, 501)
```

4.53 IWG_SetDim

Syntax

IWG_SetDim(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Sets the desired number of rows and columns in the grid.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The number of desired rows. - Limited only by memory
c	-	The number of desired columns.- Max of 512 including row numbers

Return value
None

Remarks
None

Example usage

```
IWG_SetDim(main, 501, 50, 5)
```

4.54 IWG_SetFont

Syntax

IWG_SetFont(WINDOW | DIALOG ***win***, UINT ***id***, INT ***texttype***, STRING ***typeface***, INT ***height***, INT ***weight***, OPT UINT ***flags***=0)

Description

Used to change the font of the grid's title, header row, and/or the cell body.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
texttype	-	The area whose font is being changed. Valid entries are: • @IWGTITLEFONT - the title • @IWGHEADERFONT - the header row • @IWGCELLFONT - all cells in the grid's cell body • @IWGROWNUMFONT - row number column
typeface	-	Name of the new font.
height	-	Size of font in points
weight	-	Weight of font. Ranges from 0 to 1000 with 700 being standard for bold fonts and 400 for normal fonts.
flags	-	Style flags for font and character set. Valid entries are any combination of the following: • @SFITALIC • @SFUNDERLINE • @SFSTRIKEOUT • Any one of the character set constants listed in the SETFONT section of the IWBasic© *User's Guide*.

Return value

None

Remarks

Since font is used in calculating column widths they must be set before any cells are configured.

Example usage

```
IWG_SetFont(main, 501, @IWGTITLEFONT, "Tahoma", 28, 800, @SFITALIC)
IWG_SetFont(main, 501, @IWGHEADERFONT, "Times Roman", 8, 800, @SFUNDERLINE)
IWG_SetFont(main, 501, @IWGCELLFONT, "Courier", 10, 400,0)
IWG_SetFont(main, 501, @IWGROWNUMFONT, "Comic Sans MS", 10, 400,0)
```

4.55 IWG_SetGridColor

Syntax

IWG_SetGridColor(WINDOW | DIALOG ***win***, UINT ***id***, INT ***typ***, UINT ***clr1***, OPT UINT ***clr2*** = 0)

Description
Sets the color(s) of various elements of the grid

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
typ	-	The grid element being changed. Valid entries are: • @IWGHEADERCOLOR - The header row fg and bg colors • @IWGTITLECOLOR - The grid title fg and bg colors.Black on COLOR_3DFACE • @IWGCURSORCOLOR - The cell cursor box color. [default = RGB (255,255,255)] • @IWGPROTECTCOLOR - The bg color of protected cells. [default = RGB (255,255,255)] • @IWGUNPROTECTCOLOR - The bg color of un-protected cells. [default = RGB(255,255,255)] • @IWGHILIGHTCOLOR - The bg color of highlighted cells. [default = RGB (0,0,128)] • @IWGHILIGHTTEXTCOLOR - The fg (text) color of highlighted cells. [default = RGB(255,255,255)] • @IWGGRIDLINECOLOR - The color of the grid's gridlines. [default = RGB (220,220,220)] • @IWGTITLESHADOWCLR - The color of the shadow under the title's text. [default = RGB(128,128,128)] • @IWGHEADERSHADOWCLR - The color of the shadow under the header's text. [default =RGB(128,128,128)]
clr1	-	The color to be applied. Where two colors are applied this is the fg color.
clr2	-	Where two colors are applied this is the bg color.

Return value
None

Remarks
The @IWGCURSORCOLOR is XOR'd with the @IWGGRIDLINECOLOR
See the Color sub-section in each of the sections under Getting Started.

Example usage

```
IWG_SetGridColor(main, 501, @IWGTITLECOLOR, RGB(255,0,0), RGB(255,255,0))
IWG_SetGridColor(main, 501, @IWGHEADERCOLOR, RGB(255,0,0), RGB(0,255,255))
IWG_SetGridColor(main, 501, @IWGGRIDLINECOLOR, RGB(255,0,0))
```

4.56 IWG_SetGridProt

Syntax

IWG_SetGridProt(WINDOW | DIALOG ***win***, UINT ***id***, INT ***p***)

Description
Used to prevent any cell in the entire grid from being edited.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
p	-	0 - Allows individual cell protection to control each cell's editing 1 - Stops all cell editing in the entire grid.

Return value
None

Remarks
See Also:

Example usage

```
IWG_SetGridProt(main,501,1)
```

4.57 IWG_SetHeaderAutoHeight

Syntax

IWG_SetHeaderAutoHeight(WINDOW | DIALOG ***win***, UINT ***id***, INT a***h***)

Description
Used to allow header cell contents to control header row height.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
ah	-	1 - Contents automatically adjusts the header row height 0 - Row height is fixed (default)

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetHeaderAutoHeight(main, 501,1)
```

4.58 IWG_SetHeaderHeight

Syntax

IWG_SetHeaderHeight(WINDOW | DIALOG ***win***, UINT ***id***, INT ***h***)

Description
Used to adjust the header row height.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
h	-	The desired header row height. default = 21

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetHeaderHeight(main, 501,50)
```

4.59 IWG_SetHiLightRow

Syntax

IWG_SetHiLightRow(WINDOW | DIALOG ***win***, UINT ***id***)

Description
Used to control the highlighting of the entire row containing the current cell

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

hl	-	1 - turns highlighting ON, 0 - turns it OFF (default)

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetHiLightRow(main,501, 1)
```

4.60 IWG_SetRowAutoHeight

Syntax

IWG_SetRowAutoHeight(WINDOW | DIALOG ***win***, UINT ***id***, INT a***h***)

Description
Used to allow cell contents to control overall row height

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
ah	-	1 - Contents automatically adjusts row height 0 - Row height is fixed (default). In order to insure that text with embedded "\n"s is displayed properly(with the default setting) the cell height needs to be set manually to accommodate all the rows.

Return value
None

Remarks
Automatic height adjustments are made only on the basis of cell text that contains embedded "\n". No height adjustments are made solely on the basis of normal word-wrap activities.

Example usage
```
IWG_SetRowAutoHeight(main, 501,1)
```

4.61 IWG_SetRowHeight

Syntax

IWG_SetRowHeight(WINDOW | DIALOG ***win***, UINT ***id***, INT ***h***)

Description
Sets the height of all rows in the grid

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
h	-	The height of all rows in the grid in pixels. Default is 21.

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetRowHeight(main,501, 30)
```

4.62 IWG_SetShadowOffset

Syntax

IWG_SetShadowOffset(WINDOW | DIALOG ***win***, UINT ***id***, INT ***typ***, INT offset)

Description
Sets the amount of shadow offset from the associate text.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
typ	-	The grid element being changed. Valid entries are: • @IWGHEADER - The header row shaddow • @IWGTITLE - The grid title shadow.
offset	-	The amount of offset in both the x and y directions in pixels. Negative numbers may e used. Default = 1

Return value
None

Remarks
None

Example usage

```
IWG_SetShadowOffset(main, 501, @IWGTITLE, 3)
IWG_SetShadowOffset(main, 501, @IWGHEADER, -4)
```

4.63 IWG_SetSymbol

Syntax

IWG_SetSymbol(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r,*** INT ***c,*** STRING ***sym***)

Description
Used to change the current symbol being used in @IWGMONEY type cells.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
sym	-	A single ASCII character to be used. Default is "$"(US dollars)

Return value
None

Remarks
See Also:

Example usage

```
IWG_SetSymbol(main,  501, 2, 3,"£")
```

4.64 IWG_SetTitle

Syntax

IWG_SetTitle(WINDOW | DIALOG ***win***, UINT ***id***, STRING ***title***)

Description
Used to change the grid title.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.

title	-	Text for grid title. May contain one or more "\n" to generate multi-line title.

Return value
None

Remarks
See Also:

Example usage
```
IWG_SetTitle(main, 501, "A\n different\n title\nfor\nthe\ngrid")
```

4.65 IWG_SetTitleHeight

Syntax

IWG_SetTitleHeight(WINDOW | DIALOG ***win***, UINT ***id***, INT ***height***)

Description
Used to change the current title header height.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
height	-	The desired title height

Return value
None

Remarks
Overridden by subsequent IWG_SetTitle commands

Example usage
```
IWG_SetTitleHeight(main, 501, 100)
```

4.66 IWG_ShowHeader

Syntax

IWG_ShowHeader(WINDOW | DIALOG ***win***, UINT ***id***, INT ***show***)

Description
Used to show/hide header row.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
show	-	1 - Show header row (default) 0- Hide header row.

Return value
None

Remarks
See Also:

Example usage
```
IWG_ShowHeader(main,  501, 0)
```

4.67 IWG_ShowHeaderShadow

Syntax

IWG_ShowHeaderShadow(WINDOW | DIALOG ***win***, UINT ***id***, INT ***show***)

Description
Used to show/hide a shadow under the header text

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
show	-	1 - Show header text shadow 0 - Hide header text shadow (default).

Return value
None

Remarks
See Also:

Example usage
```
IWG_ShowHeaderShadow(main, 501, 1)
```

4.68 IWG_ShowRowNumbers

Syntax

IWG_ShowRowNumbers(WINDOW | DIALOG ***win***, UINT ***id***, INT ***show***)

Description
Used to show/hide row numbers.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
show	-	1 - Show row numbers (default) 0 - Hide row numbers.

Return value
None

Remarks
See Also:

Example usage
```
IWG_ShowRowNumbers(main, 501, 0)
```

4.69 IWG_ShowTitleShadow

Syntax

IWG_ShowTitleShadow(WINDOW | DIALOG ***win***, UINT ***id***, INT ***show***)

Description
Used to show/hide a shadow under the title text

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
show	-	1 - Show title text shadow 0 - Hide title text shadow (default).

Return value
None

Remarks
See Also:

Example usage
```
IWG_ShowTitleShadow(main, 501, 1)
```

4.70 IWGM_InitMaskCellDat

Syntax

IWGM_InitMaskCellDat(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***text***, STRING ***mask***, STRING ***prompt***, OPT INT ***flags****=0,* OPT INT ***intype****=0)*

Description
Used to set the initial basic configuration for a specified @IWGMASK type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
text	-	The initial text to place in the cell. NOTE: The User is responsible for insuring that the text complies with the normal input restrictions for the specified cell type.
mask	-	The mask used to format raw data input.
prompt	-	The character used to indicate a blank entry in the input location in the mask.
flags	-	The optional styles to be applied to the cell's contents.
intype	-	Indicates the type of text in the text field above: 0 = raw, unformatted data 1 = formatted data that matched mask formatting

Return value
None

Remarks
See @IWGMASK for details of mask formatting.

Valid flags:

@IWG_PROTECTCELL	Sets the initial state of the cell to protected (non-editable).
@IWG_LEFTALIGN	Aligns the displayed text to the left. Default.
@IWG_RIGHTALIGN	Aligns the displayed text to the right.
@IWG_CENTERALIGN	Aligns the displayed text to the center.

@IWG_CASE_ANY	Alpha characters are displayed the same as they are entered. Default.
@IWG_CASE_UPPER	All alpha characters are converted to upper case before displaying.
@IWG_CASE_LOWER	All alpha characters are converted to lower case before displaying.

Example usage

```
IWGM_InitMaskCellDat(win, ID_GRID1, 4, 3, "1234567890", "(999) 999-9999", "_",
@IWG_CENTERALIGN, 0)
IWGM_InitMaskCellDat(win, ID_GRID1, 5, 6, "(123) 456-7890", "(999) 999-9999", "_",
@IWG_CENTERALIGN, 1)
```

4.71 IWGM_GetCellFormatDat

Syntax

STRING = **IWGM_GetCellFormatDat**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to read the formatted contents of the specified @IWGMASK type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The formatted string contents of the cell at the specified position.

Remarks
Can only be used with a @IWGMASK type cell.

Example usage

```
STRING text = IWGM_GetCellFormatDat(main,  501, 22, 3)
```

4.72 IWGM_GetCellRawDat

Syntax

STRING = **IWGM_GetCellRawDat**(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***)

Description
Used to read the raw contents of the specified @IWGMASK type cell..

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell

Return value
The raw string contents of the cell at the specified position.

Remarks
Can only be used with a @IWGMASK type cell.

Example usage
```
STRING text = IWGM_GetCellRawDat(main,  501, 22, 3)
```

4.73 IWGM_SetCellFormatDat

Syntax

IWGM_SetCellFormatDat(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***dat***)

Description
Loads new formatted text into the @IWGMASK type cell..

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The new formatted text to be placed in the designated cell.

Return value
None

Remarks
Can only be used with a @IWGMASK type cell.

Example usage
```
IWGM_SetCellFormatDat(main,  501, 11, 2, "(123) 456-7890")
```

4.74 IWGM_SetCellRawDat

Syntax

IWGM_SetCellRawDat(WINDOW | DIALOG ***win***, UINT ***id***, INT ***r***, INT ***c***, STRING ***dat***)

Description
Loads new raw text into the @IWGMASK type cell.

Parameters

win	-	The parent of the grid control.
id	-	The id of this instance of the grid control.
r	-	The row position of the desired cell.
c	-	The column position of the desired cell
dat	-	The new raw text to be placed in the designated cell.

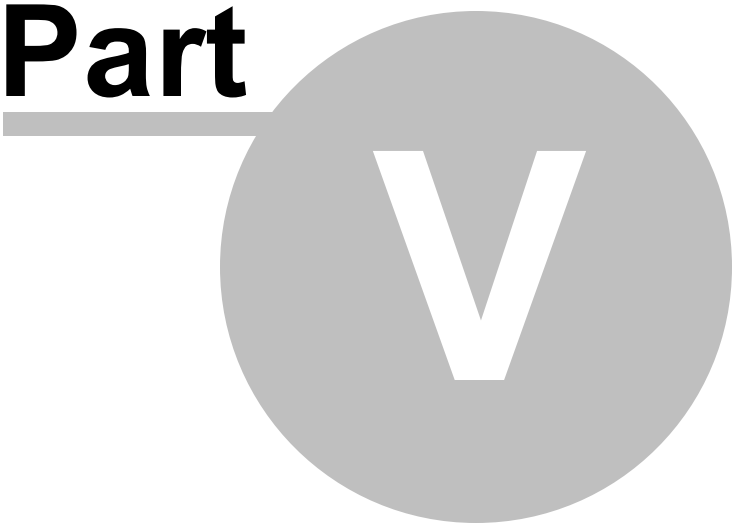
Return value
None

Remarks
Can only be used with a @IWGMASK type cell.

Example usage

```
IWGM_SetCellRawDat(main,  501, 11, 2, "1234567890")
```

Notification Messages



5 Notification Messages

An IWGrid control sends notification messages to the parent window or dialog in the @NOTIFYCODE variable. The ID of the control is found in @CONTROLID. The notification messages in the following table are supported:

Message ID	Meaning
@IWGN_ROWCHANGED	The User has clicked on a cell that is in a different row than the previously selected cell.
@IWGN_COLCHANGED	The User has clicked on a cell that is in a different column than the previously selected cell.
@IWGN_EDITBEGIN	The editing of a cell has begun.
@IWGN_EDITEND	The editing of a cell has ended.
@IWGN_BTNSUBCLICKED	The button in an @IWGSUBBUTTON type cell has been clicked.
@IWGN_CBSELCHANGED	The selection in a @IWGCOMBO type cell has changed.
@IWGN_CKBOXCHECKED	An @IWGCHECK type cell has transitioned to the Checked state
@IWGN_CKBOXUNCHECKED	An @IWGCHECK type cell has transitioned to the UnChecked state
@IWGN_RADIOCHECKED	An @IWGRADIOBUTTON type cell has transitioned to the Checked state
@IWGN_RADIOUNCHECKED	An @IWGRADIOBUTTON type cell has transitioned to the UnChecked state
@IWGN_IMAGECLICKED	An @IWGIMAGE type cell has been clicked.
@IWGN_CELLCLICKED	A cell of any type has been clicked
@IWGN_BUTTONCLICKED	The button in an @IWGBUTTON type cell has been clicked.
@IWGN_F1	The indicated function key has been pressed.
@IWGN_F2	The indicated function key has been pressed.
@IWGN_F3	The indicated function key has been pressed.
@IWGN_F4	The indicated function key has been pressed.
@IWGN_F5	The indicated function key has been pressed.
@IWGN_F6	The indicated function key has been pressed.
@IWGN_F7	The indicated function key has been pressed.

@IWGN_F8	The indicated function key has been pressed.
@IWGN_F9	The indicated function key has been pressed.
@IWGN_F10	The indicated function key has been pressed. NOTE: Unless this key has been assigned with the ADDACCELERATOR function by the parent window, pressing this key will set focus on the parent windows menu bar.
@IWGN_F11	The indicated function key has been pressed.
@IWGN_F12	The indicated function key has been pressed.

NOTE: With each notification message the identification of the current cell can be determined using the IWG_CellLocate utility function provided in the IWGrid library.

```
SUB mainHandler(),INT
   int r,c
   SELECT @MESSAGE
      CASE @IDCONTROL
         SELECT @controlid
            CASE 501
               SELECT @notifycode
                  CASE @IWGN_CELLCLICKED
                     IWG_CellLocate(@LPARAM,r,c)
                     PRINT "@IWGN_CELLCLICKED ",r,c
               ENDSELECT
         ENDSELECT
   ...
```

Revision History

Part VI

6 Revision History

Date	Version	Remarks
2015-Feb-26	v1.03	fixed - when grid was resized via dragging parent last rows of grid were not properly displayed when scrolling to bottom of grid.
2014-Nov-23	v1.02	Add @IWGROWNUMFONT option flag to the IWG_SETFONT command so row number column font can be set different from cell font.
2014-Oct-17	v1.01	Reinstated the IWG_EnableEditDialog command and revised its functionality
2014-Oct-03	v1.0	Official Release Added the following commands: • IWGM_InitMaskCellDat • IWGM_GetCellFormatDat • IWGM_GetCellRawDat • IWGM_SetCellFormatDat • IWGM_SetCellRawDat Removed the following commands • IWG_GetCellMask • IWG_GetCellMaskPrompt • IWG_SetCellMask • IWG_SetCellMaskPrompt added @IWGRADIOBUTTON type cell added IWG_SetShadowOffset command fixed - visual operation of @IWGBUTTON cell was imprpoer fixed - converted @IWGCOMBO to pointer to hold long string fixed - @IWGCOMBO could be update with invalid entry. fixed - @IWGINT allowed "-" at other than 1st character fixed - @IWGNUMF allowed more than 1 decimal character fixed - @IWGMONEY allowed more than 1 decimal character fixed - @IWGNUMF allowed a number to be inserted before the '-' sign. fixed - @IWGMONEY allowed a number to be inserted before the '-' sign. fixed - @IWGNUMF allowed a '.' to be inserted before the '-' sign. fixed - @IWGMONEY allowed a '.' to be inserted before the '-' sign. fixed - could paste improper characters into a cell - disabled pasting in all but @IWGHEX, @IWGNUMF,@IWGNUMI, and @IWGMONEY type cells. fixed - IWG_SetCellDecPlaces command was not updating grid upon initial configuration fixed - IWG_SetCellNegParenth command was not updating grid upon

		initial configuration fixed - IWG_SetCellNoSymbol command was not updating grid upon initial configuration fixed - IWG_SetCellRoundDec command was not updating grid upon initial configuration fixed - IWG_SetSymbol command was not updating grid upon initial configuration fixed - IWGIMAGE cell images were not loaded correctly in all cases fixed - @IWGSUBBUTTON cell contents not updating properly
2014-Mar-09	v0.11	Added the following commands: • IWG_ColResizing • IWG_ShowRowNumbers fixed - font size was not being applied consistently when adding/ modifying title text and title font fixed - title header height was not being calculated consistently fixed - column header auto-numbers not properly centered fixed - changing font did not change column header height when autoheaderheight was active.
2014-Jan-21	v0.10	Added the following commands: • IWG_GetCellMask • IWG_SetCellMask • IWG_GetCellMaskPrompt • IWG_SetCellMaskPrompt • IWG_GetCellType command Added multiple dialogs for editing cell contents Added code to support input formatting mask Added code to support @IWGIPADDRESS type cells Fixed - If the same image was loaded into multiple cells each one created a new copy of the image in memory. Now, only one copy will exists in memory regardless of how many times it is used in a grid
2014-Jan-15	v0.09	Added error message if attempt is made to use IWG_SetDat with @IWGIMAGE type cell Added ability to load IWGIMAGE cells from a resource file.
2013-Dec-10	v0.08	Fixed - @IWGIMAGE type cells were not being initialize correctly when there was no file name. Fixed - Home and End keys were not scrolling the grid. Added-if line hilite feature is on and column 0 is clicked the hilited row will change to the clicked on row
2013-Dec-09	v0.07	Corrected several errors in how different colors are applied to cells. Added the following commands: • IWG_SetCellImgMissingColor • IWG_GetCellImgMissingBGColor

		• IWG_GetCellImgMissingFGColor
2013-Dec-07	v0.06	Fixed- misalignment of text with certain flag combinations. Modified @IWGNOSCALE and @IWGRATIO flag operation and how they interact with each other. Added the following commands: • IWG_GetCellButtonSize • IWG_SetCellButtonSize • IWG_GetCellImageSize • IWG_SetCellImageSize • IWG_GetCellImageMissingText • IWG_SetCellImageMissingText
2013-Dec-04	v0.05	Added @IWGBUTTON cell type Added edit dialogs to handle multiline cells better. Corrected vertical centering issue. Corrected issue with grid resizing itself Added the following commands: • IWG_CellLocate • IWG_EnableEditDialog • IWG_GetImageSize
2013-Oct-18	v0.04	Deleted IWG_SetEllipsis command Added: • IWG_GetCheckState • IWG_SetCheckState • IWG_SetHeaderAutoHeight • IWG_SetRowAutoHeight Fixed several issues with grid sizing
2013-Jul-31	v0.03	Fixed several problems and added new functions
2013-Jul-28	v0.02	Added numerous functions and updated demo
2013-Jul-01	v0.01	Initial Alpha Release to public